Marco de trabajo para el diseño y desarrollo de Herramientas de modelado conceptual basado en DSL utilizando tecnologías GMF

Leandro Rocca[1], Matías N. Caputti[1], Iván Zugnoni[1], Lucas Paganini[1], Juan Cesaretti[1],
Leopoldo Nahuel[1,2], Giandini Roxana[1,2,3]

[1]GIDAS – Grupo de Investigación del Departamento de Sistemas – Facultad Regional La Plata Universidad Tecnológica Nacional, La Plata, Buenos Aires, Argentina

[2]LIFIA – Facultad de Informática – Universidad Nacional de La Plata
La Plata, Buenos Aires, Argentina

[3]CIC – Comisión de Investigaciones Científicas de la Provincia de Buenos Aires

{lrocca, macaputti, izugnoni, jcesaretti, lpaganini, lnahuel, rgiandini} @frlp.utn.edu.ar

Resumen. *Innovando en la forma de poner en práctica Ingeniería de Requerimientos Dirigida por Modelos IRDM en el contexto del Desarrollo Dirigido por Modelos MDD, nuestro trabajo consiste en el diseño y desarrollo de una herramienta CARE (Computer-Aided Requirements Engineering) que asista en el uso de nuestro Lenguaje Específico del Dominio (DSL) para Salud, al que llamamos DSL_SALUD [1]. Esta herramienta permite crear modelos escritos en el lenguaje DSL_SALUD y transformarlos automáticamente a otros modelos escritos en linguaje UML, por ejemplo: diagrama de clases UML.*

Palabras clave. *Ingeniería de Requerimientos Dirigida por Modelos IRDM, Lenguaje Específico del Dominio DSL, Metalenguajes, Desarrollo Dirigido por Modelos MDD, Herramienta CARE, Proceso de Desarrollo de Software, Agilidad, Trazabilidad de Requerimientos, Usabilidad del Software.*

1. Introducción

La Ingeniería de Requerimientos Basada en Modelos (IRBM), es una de las más importantes ya que integra un conjunto de conocimientos que se aplican durante la primera etapa del desarrollo, apuntando a obtener un relevamiento del dominio y el problema en sí, lo más acertado posible, de manera de entender qué es lo que se requiere hacer. Esta etapa es indispensable si se pretende lograr una buena solución tecnológica acorde a las necesidades de los usuarios, y es por esto, que la IRBM toma fuerte relevancia en el campo de la ingeniería de software.

A su vez, somos testigos del surgimiento de un nuevo paradigma en la ingeniería de software, que se propone metas muchos más altas que las que el enfoque tradicional, basado en modelos, proponía. Hablamos de Desarrollo Dirigido por Modelos. (MDD por sus siglas en inglés). MDD propone que sean los modelos los que dirijan el desarrollo entero durante todas sus etapas. De esta manera partimos de los modelos con alto nivel de abstracción llegando a los más concretos, logrando finalmente la generación del código fuente, a través de transformaciones sucesivas de modelos.

Con esta idea surgen varias técnicas de análisis y diseño de sistemas que dan soporte al relevamiento y el entendimiento del dominio, en conjunto con lenguajes de modelado estándar que permiten la esquematización y la documentación de sistemas complejos.

En este contexto y bajo la premisa de que los requerimientos completos, no ambiguos y rastreables, promueven la eficiente administración del cambio, evolución y mantenimiento del producto, surge la Ingeniería de Requerimientos.

2. Marco teórico

En esta sección introducimos los principales conceptos, paradigmas y estándares que dan base al propósito del presente trabajo.

2.1. Ingeniería de Requerimientos

La Ingeniería de Requerimientos (IR) es considerada hoy un tema clave y fundamental de la Ingeniería de Software, esto se evidencia en el hecho de contar con su propio campo de trabajo y su avance continuo como disciplina. El proceso de IR es un proceso iterativo que consta a su vez de tres grandes subprocesos: elicitación, especificación y validación de requerimientos. La IR logró expandirse para no sólo abordarse en etapas tempranas del ciclo de vida del software, sino en prácticamente todas las etapas. Al parecer, el principal motivo del avance de esta disciplina tiene que ver con el hecho de que requerimientos bien especificados y rastreables, entre los artefactos o productos de desarrollo, ayudan a una eficiente gestión del cambio, evolución y mantenimiento del producto software, llegando así a productos de mejor calidad y escalables.

2.2. MDD: una metodología de trabajo para la producción de aplicaciones software

La Ingeniería de Software [4] tiene la problemática de construir software de calidad, que pueda ser capaz de sobrevivir a la evolución de sus requisitos funcionales, y que sea flexible a los cambios en la tecnología que lo sustenta, es actualmente contemplado en los principios del paradigma de desarrollo de software conocido como MDD (por sus siglas en inglés: Model Driven Software Development) [1, 2, 3]. Este paradigma ofrece mejorar los procesos de construcción de software. La idea troncal de MDD, es obtener mediante transformaciones automáticas, modelos más específicos o concretos, a partir de otros más abstractos.

Los postulados básicos de MDD son los siguientes: los modelos asumen un rol protagónico en el proceso de desarrollo del software; los modelos pasan de ser entidades contemplativas para convertirse en entidades productivas a partir de las cuales se deriva la implementación en forma automática.

La iniciativa MDD promueve:

- el uso de un mayor nivel de abstracción tanto en la especificación del problema a resolver como de la solución correspondiente, en relación con los métodos tradicionales de desarrollo de software.
- el aumento de confianza en la automatización asistida por computadora para soportar el análisis, el diseño y la ejecución.
- el uso de estándares industriales como medio para facilitar las comunicaciones, la interacción entre diferentes aplicaciones y productos, y la especialización tecnológica.

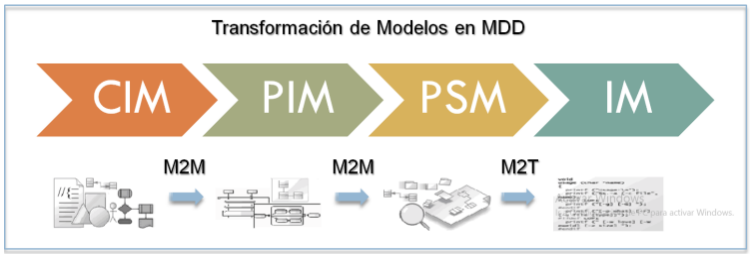

Figura 1: Transformación de modelos en MDD

Una propuesta concreta utilizada en el ámbito MDD es la idea de crear modelos para un dominio específico a través de lenguajes DSLs (por su nombre en inglés: Domain-Specific Language), focalizado y especializado para dicho dominio. Estos lenguajes permiten especificar la solución usando directamente conceptos del dominio del problema. Los productos finales son luego generados automáticamente desde estas especificaciones de alto nivel. La técnica más usada de para especificar un DSL es el metamodelado donde se define qué elementos pueden existir en el modelo. Por ejemplo, en el metamodelo DSL_SALUD encontraremos a "Paciente", "Encuentro", "Profesional de Salud", etc. que luego aparecerán instanciadas en un modelo UML y también en un modelo relacional.

2.3. Metamodelos y Metalenguajes

La definición de un lenguaje de modelado establece qué elementos pueden existir en un modelo. Usando un lenguaje de modelado, podemos crear modelos que especifican qué elementos pueden existir en un sistema.

Se puede describir un lenguaje por medio de un modelo, llamado "metamodelo" del lenguaje, que describe qué elementos pueden ser usados en el lenguaje y cómo deben ser conectados.

Un metamodelo es también un modelo y por lo tanto debe estar escrito en un lenguaje bien definido que llamamos "metalenguaje".

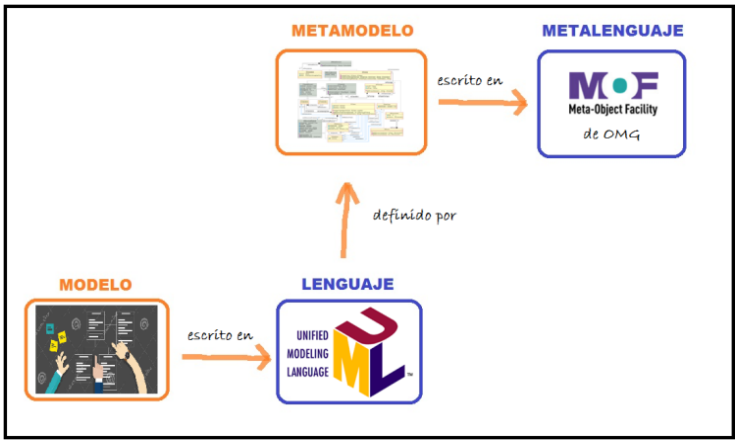

Figura 2: Modelo, Lenguaje, Metamodelos y Metalenguajes

Siguiendo la Figura 2, para el caso del lenguaje de modelado UML, tenemos por un lado modelos escritos en UML construidos en el contexto de un desarrollo de software. Por otro lado, tenemos el lenguaje UML que está definido por su metamodelo, el metamodelo de UML está escrito en el metalenguaje Meta-Object Facility MOF. El metamodelo de UML, y sus sucesivas versiones, son publicados como estándar por OMG [11].

El metamodelo describe la sintaxis abstracta del lenguaje. Esta sintaxis es la base para el procesamiento automatizado (basado en herramientas) de los modelos, de lo que destacamos como relevante, las transformaciones automáticas entre modelos. Por otra parte, la sintaxis concreta es definida mediante otros mecanismos, es la interfaz para el modelador e influye fuertemente en el grado de legibilidad de los modelos.

La arquitectura de cuatro capas de modelado de OMG cuenta con estos niveles:

- Nivel M0: Instancias. Se trata de todas las instancias reales del sistema, es decir, los objetos de la aplicación.
- Nivel M1: Modelo del sistema. Representa el modelo de un sistema de software. Los conceptos del nivel M1 representan categorías de las instancias de M0.
- Nivel M2: Metamodelo. En este nivel aparecen conceptos claves como Clase, Atributo y Operación. Por ejemplo: metamodelo de UML.
- Nivel M3: Meta-metamodelo. Es el nivel más abstracto, que permite definir metamodelos concretos. Dentro del OMG, MOF es el lenguaje estándar de la capa M3.

2.4. Modelos de Información para Historia de Salud Electrónica

Existen estándares internacionales que ofrecen modelos de información para Historia de Salud Electrónica que tienen una gran divulgación y muchos casos de éxito en desarrollos reales para la industria de software específica para Salud. Los más destacados son HL7 y OpenEHR. Estas organizaciones surgieron por la gran necesidad de intercambio de información entre sistemas informáticos que existe en el área de salud, esto es, interoperabilidad entre sistemas.

HL7 Internacional es una organización fundada en 1987 que se dedica al desarrollo de estándares para el ámbito de salud. El objetivo es minimizar las incompatibilidades entre sistemas de información en salud, permitiendo el intercambio de datos entre aplicaciones heterogéneas, independientemente de su plataforma tecnológica. OpenEHR es un estándar abierto que describe la administración y almacenamiento de información sanitaria en forma de informes de historia clínica electrónica (HCE).

3. Desarrollo del Lenguaje Específico del Dominio DSL_SALUD

Comenzamos con el Desarrollo de un Lenguaje Específico del Dominio (DSL) para Salud, al que llamaremos DSL_SALUD, a partir de estándares de Interoperabilidad y Modelado de Información para Historia Clínica Electrónica: FHIR de HL7 y OpenEHR, considerando aspectos estáticos y dinámicos del dominio.

Desde el punto de vista del estudio profundo de los conceptos propios del dominio, se analizaron los modelos de información que brindan los estándares FHIR de HL7 y OpenEHR. Se hizo un recorte del modelo de información de FHIR considerando recursos de información fundamentales: Resource, DomainResource, Patient, Practitioner, Encounter y EpisodeOfCare. Con estas clases se podrá armar un DSL_SALUD incial que luego iremos ampliando en futuras iteraciones del proceso de construcción del DSL.

Desde el punto de vista tecnológico de implementación del DSL, se evaluaron distintos entornos de trabajo para la construcción de DSLs, entre los cuales destacamos: Domain-Specific Language Tools de Microsoft, Sirius y DSL Toolkit, siendo las dos últimas partes del proyecto Eclipse Modeling Project. Elegimos como entorno a la herramienta

DSL Toolkit por ser más versátil en cuanto a las posibilidades de modelado gráfico y textual, y porque cuenta con la posibilidad de utilizar los lenguajes de transformación de modelos ATL y QVT en forma integrada en el mismo entorno.

El paso siguiente será llevar a cabo el propio desarrollo tecnológico del metalenguaje DSL_SALUD utilizando Eclipse Modeling Framework, EMF, dentro de DSL Toolkit. EMF nos permitirá modelar y desarrollar el núcleo de nuestro DSL_SALUD. Utilizando EMF desarrollaremos la sintaxis abstracta de nuestro DSL_SALUD. Además, permitirá aplicar al mismo distintas restricciones OCL y validar los modelos. En este paso se construirá el metamodelo DSL_SALUD como una estructura ECORE.

Como puede verse en la *Figura 3*, nuestro DSL_SALUD o Metamodelo de Dominio Salud resultante estará compuesto por:

- La Definición del Diagrama con las interacciones entre las clases que componen nuestro dominio.
- Las Reglas de Transformación M2M.
- Las Reglas de Transformación M2T.
- Las Definiciones Textuales de la Sintaxis.

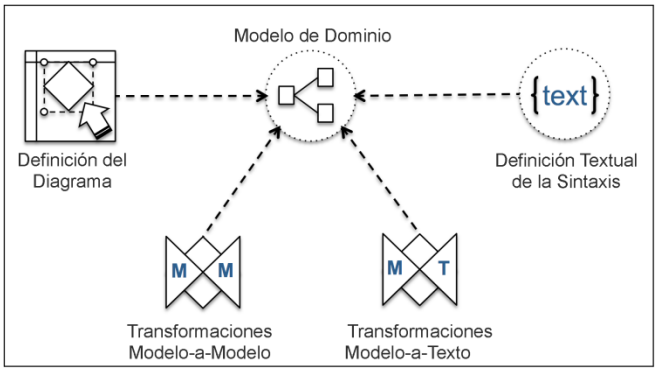

Figura 3: Metamodelo del Dominio Salud, DSL_SALUD

4. Diseño y desarrollo de Herramienta CARE utilizando tecnologías GMF

En esta etapa ya contamos con el lenguaje específico de dominio definido y se podrá comenzar con la etapa de diseño y desarrollo de la herramienta modeladora CARE, Computer-Aided Requirements Engineering.

Nuestro DSL_SALUD se continuará con el desarrollo tecnológico de una herramienta CARE (Computer-Aided Requirements Engineering).

La herramienta CARE será construída para que asista al ingeniero de software en las siguientes cuestiones:

- construcción de modelos escritos en DSL_SALUD (que a su vez está basado en estándares internacionales de modelado de información para SALUD) mediante interfaz gráfica para modelado.
- ejecutar transformaciones automáticas desde estos modelos escritos en lenguaje DSL_SALUD hacia modelos escritos otros lenguajes, es decir, automáticamente transformar estos modelos independientes de la computación CIM hacia

modelos independientes de la plataforma PIM escritos en UML, por ejemplo: diagrama de clases UML, y también transformar estos a modelos PSM como el modelo relacional.

Todo esto se hace con la intención de que los Ingenieros de Software, especializados en soluciones IT para Salud, puedan hacer uso tanto del metalenguaje DSL_SALUD como de la herramienta CARE para llevar a cabo la construcción de modelos que representen los requerimientos propios para sus desarrollos de aplicaciones software para Salud, es decir, de esta forma poner en práctica Ingeniería de Requerimientos Dirigida por Modelos IRDM en el contexto del Desarrollo Dirigido por Modelos MDD.

4.1 Creación de la herramienta modeladora con GMF

El proceso de creación de la herramienta CARE será realizado utilizando GMF, Graphical Modelling Framework.

GMF posee 6 etapas cuyo producto final está representado por un plugin de Eclipse en estado puro, o en su defecto una aplicación Eclipse RCP. Las etapas pueden resumirse de la siguiente forma:

- Creación del metamodelo: utilizado para la creación de nuevos modelos de parte del usuario.
- Creación del set de herramientas del plugin: proporciona la forma de uso del plugin final, a través de los elementos gráficos disponibles al usuario.
- Creación del modelo de mapeo gráficos: necesario para el mapeo de nuevos modelos definidos por el usuario a través de las herramientas gráficas.

En la Figura 4 se muestra el dashboard utilizado durante la creación de un proyecto GMF. En los siguientes apartados se explicará cada una de las etapas.

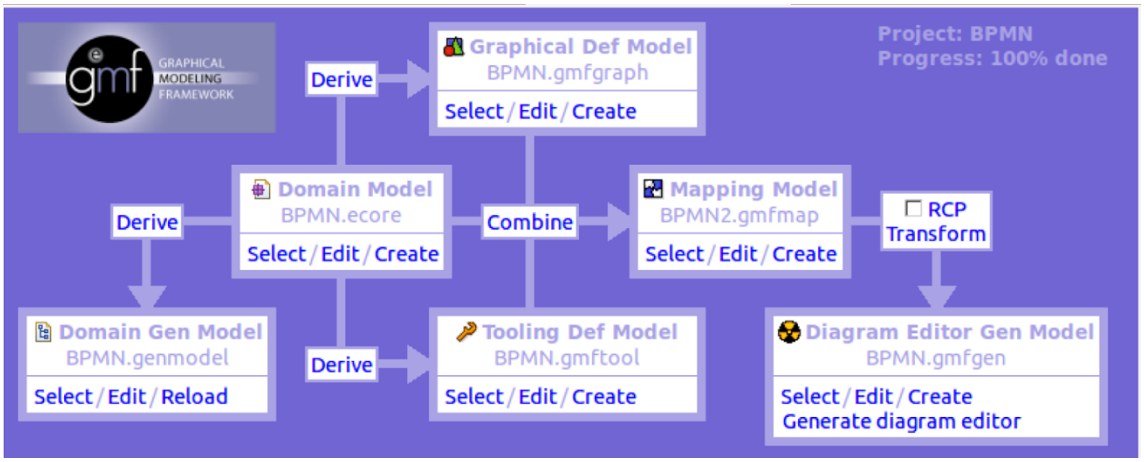

Figura 4: Dashboard del desarrollo de un proyecto GMF

4.1.1 Definición del metamodelo (Ecore metamodel)

EMF[17] está basado en 2 metamodelos[18]: el Ecore y el Genmodel. El metamodelo Ecore contiene la información sobre las clases definidas. El modelo Ecore permite definir diferentes elementos:

- EClass: representa un atributo que contiene el nombre y el tipo.
- EAtribute: representa un atributo que tiene un nombre y un tipo.

- EReference: representa un extremo de la asociación entre dos clases. Tiene un flag para indicar si es un contenedor y una referencia a la clase que apunta.
- EDataType: representa el tipo de un atributo.

El modelo Ecore posee una estructura de árbol. Dentro de este modelo existe un objeto raíz que representa al modelo Ecore. Este modelo tiene hijos que representan paquetes, y sus hijos representan las clases, mientras que los hijos de las clases representan los atributos de estas clases. El otro metamodelo utilizado GenModel, contiene información adicional para la generación de código, por ejemplo el directorio y la información del archivo. Produce las clases de implementación java para un modelo o instancia del metamodelo Ecore. El GenModel también contiene los parámetros de control sobre cómo debe ser generado el código. Define los mapeos necesarios entre el metamodelo y definiciones para configurar dos plugins llamados edit y editor que contienen operaciones de bajo nivel que necesita el plugin que se generará para funcionar.

El metamodelo de Dominio Salud, DSL_SALUD, puede observarse en la Figura 5.

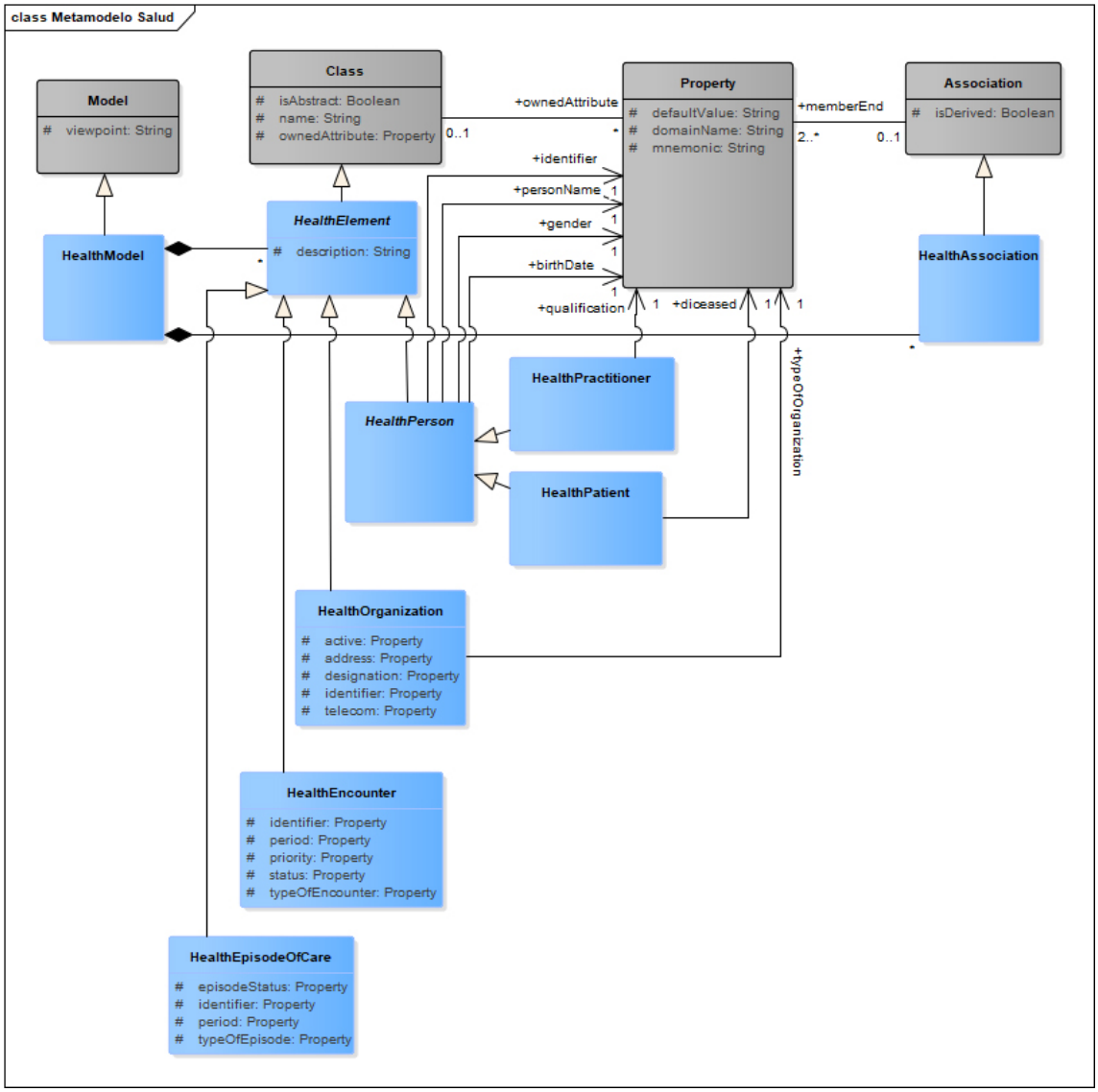

Figura 5: Metamodelo de Dominio Salud – DSL_SALUD

4.1.2 Definición gráfica del metamodelo (GMFGraph)

GMF permite generar la infraestructura y componentes necesarios en tiempo de ejecución para desarrollar editores gráficos. Al estar basado en Ecore EMF y GEF, es utilizado para producir un editor gráfico que genera instancias de metamodelos[19]. GMFGraph es utilizado para definir los elementos gráficos de nuestro dominio. En este caso, vinculados con cada uno de los elementos del modelo de dominio o metamodelo ecore DSL_SALUD.

4.1.3 Configuración de las barras de herramientas (GMFTool)

Este archivo es utilizado para definir la paleta de herramientas que puede usarse en el editor gráfico. En cada una de las barras de herramientas definidas, se podrá hacer uso de los elementos que deban estar disponibles al usuario. Por ejemplo, en el metamodelo DSL_SALUD definido existen ciertas clases como es el caso de ProcessObject, las cuales no deben poder ser insertadas en el modelo creado por el usuario, de esta forma, al momento de crear las barras de herramientas con los elementos disponibles, se deberán considerar cuáles de los mismos son necesarios y cuales no.

4.1.4 Mapeo de configuraciones con el modelo GMFMap

En este archivo se vincula o mapean los modelos originados por las anteriores configuraciones: el modelo de dominio Ecore, el modelo gráfico GMFGraph y el modelo de herramientas GMFTool. GMFMap es quien permite la generación de código del editor.

Como se observa en la Figura 6, la forma de identificar visualmente los objetos de la clase Process es a través de rectángulos redondeados con borde sólido. Así, cada una de las representaciones gráficas del metamodelo utilizado se encontraran en el archivo .gmfmap.

Figura 6: Notación gráfica de un elemento de tipo Process

4.1.5 Generación del plugin a través de GMFGen

Este archivo contiene la información necesaria para poder generar el plugin que hará posible la edición gráfica de modelos definidos a través del metamodelo desarrollado. GMFGen se compila mediante GMF, obteniendo como producto final el plugin desarrollado. De esta forma se logra un editor gráfico en base a GMF y la utilización de EMF (GenModel).

4.2 Despliegue de la herramienta CARE

Teniendo nuestro lenguaje específico de dominio DSL_SALUD y la herramienta CARE desarrollada, tal como está desarrollado en secciones anteriores. Ahora podremos instanciar un nuevo diagrama tipo dsl_salud que estará reglamentado a partir de nuestro metamodelo DSL_SALUD.

En la Figura 7 se pueden observar dos archivos que se crean al iniciar un nuevo proyecto DSL_SALUD: uno que contendrá nuestro modelo (de tipo dslsalud), y otro que contendrá un diagrama gráfico representando a nuestro modelo (de tipo dslsalud_diagram).

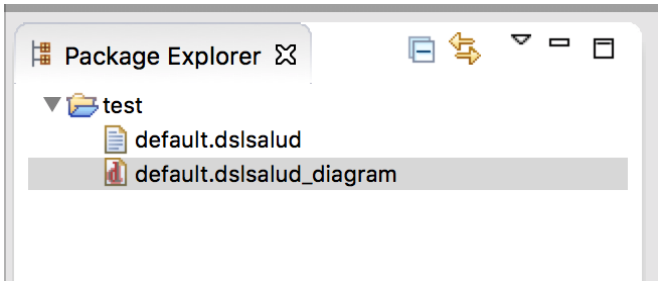

Figura 7: Archivos de un proyecto DSL_SALUD

En la Figura 8 se puede observar el toolkit final, el cual nos permitirá instanciar los distintos elementos de nuestro DSL.

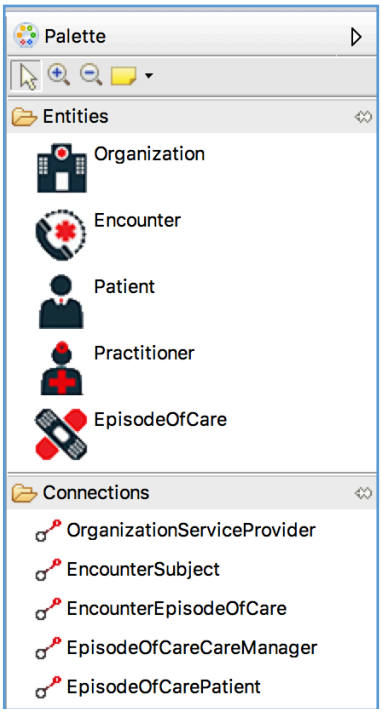

Figura 8: Vista del toolbox de la herramienta CARE DSL_SALUD

5. Marco Metodológico

Crear un marco de trabajo conceptual y metodológico en el que se lleve a cabo Ingeniería de Requerimientos Dirigida por Modelos IRDM en el contexto del Desarrollo de Software Dirigido por Modelos MDD.

El aporte metodológico no es exclusivo para el dominio de la Salud, dado que, de la misma forma podría implementarse en otro dominio. Transfiriendo el conocimiento, la metodología de trabajo y la herramienta CARE a equipos de desarrollo reales, se podrá analizar el aporte realizado por la construcción y uso del metalenguaje a través de dicha herramienta, teniendo en cuenta aspectos claves del Proceso de Desarrollo de Software como la Agilidad del proceso mismo, la Trazabilidad de requerimientos a lo largo de todos los productos del desarrollo, la Usabilidad del producto software final, entre otros.

Llevar a cabo un estudio experimental en un Proyecto de Desarrollo de Software puntual, que permita evaluar la aplicación del marco conceptual y metodológico, junto con el uso de la herramienta CARE, con estos sub-objetivos:

- Analizar cómo la incorporación de Transformación de Modelos al Proceso Ingeniería de Requerimientos aporta específicamente a la trazabilidad de requerimientos y brinda agilidad al Proceso Desarrollo de Software.
- Analizar incidencia de la incorporación de Transformación de Modelos al Proceso Ingeniería de Requerimientos en aspectos comunicacionales y aceptación del producto software (Usabilidad).
- Conocer el grado en que la incorporación de Transformación de Modelos al Proceso Ingeniería de Requerimientos adelanta las solicitudes de cambios de requerimiento a etapas tempranas del Proceso.
- Desarrollo de Software optimizando así la gestión de recursos del proyecto de desarrollo.

Finalmente se realizará un estudio experimental del uso de esta herramienta CARE en el contexto de un proyecto de desarrollo de software real en el ámbito de la Salud. Así se podrán analizar las consecuencias del uso de esta herramienta.

Para llevar a cabo este estudio experimental, por un lado, en una primera etapa, se asistirá en la implementación y uso de la herramienta CARE, donde se llevarán a cabo entrevistas no estructuradas con preguntas abiertas a los stakeholders y a los miembros del equipo de desarrollo, y por otro lado en una segunda etapa, se realizarán cuestionarios estructurados para completar el estudio.

6. Conclusiones y Trabajo Futuro

Esta publicación surge del trabajo en conjunto del equipo de investigadores del PI&D: "Herramientas de soporte a la Ingeniería de Requisitos Dirigida por Modelos: desde las necesidades de negocio hacia los requisitos de software", proyecto parte del Grupo GIDAS del Departamento de Sistemas de Información de la Facultad Regional La Plata - UTN. El PID cuenta con 9 miembros entre coordinadores, investigadores graduados y alumnos, y está homologado por Rectorado UTN.

Las líneas de Investigación del PID son: Ingeniería de Requisitos Dirigida por Modelos, Procesos de Negocio, Lenguajes Específicos de Dominio, Herramientas CARE.

Este proyecto es la continuidad de un PID anterior llamado "Modelado Ágil para la Producción de Software MAPS" por el que se realizaron publicaciones [12, 13, 14, 15, 16] siempre con la temática principal: Desarrollo Dirigido por Modelos.

En lo que va de proyecto ya se ha demostrado que es factible su realización desde todo punto de vista, por lo que se comenzará a desarrollar el DSL_SALUD y la herramienta CARE que van a ser distribuidos como un plugin para Eclipse que permite construir modelos a partir de nuestro metamodelo DSL_SALUD, extendido del modelo OpenEHR/FHIR, disponer de la funcionalidad de transformaciones automáticas M2M y M2T (escritas en QVT y Xpand respectivamente).

Dicho plugin permitirá llevar a cabo una correcta modelización y transformación automática desde un modelo construido e instanciado visualmente a partir de nuestro DSL, hacia distintos diagramas UML como por ejemplo Diagrama de Actividades, contribuyendo así a la generación de un modelo PIM necesario para la etapa de inicio del proceso de desarrollo de sistemas orientados a objetos.

Referencias

[1] Ariste, C., Rocca, L., Caputti, M., Zugnoni, I. “Diseño de un Lenguaje Específico del Dominio para sistemas de Salud basado en estándares de interoperabilidad FHIR de HL7 y OpenEHR)”. Congreso Argentino de Informática y Salud (2017).

[2] Pons, C., Giandini, R. y Pérez, G. “Desarrollo de Software Dirigido por Modelos: conceptos teóricos y su aplicación práctica”. 1°Edición 2010 . EDULP & McGraw-Hill.

[3] J. García, F. O. García, V. Pelechano, A. Vallecillo, J.M. Vara, C. Vicente-Chicote. “Desarrollo de Software Dirigido por Modelos”. ISBN 978-84-9964-215-4 (2013).

[4] MDA, “Model Driven Architecture Guide” (OMG). V.2.0, 2014. Disponible en http://www.omg.org/cgi-bin/doc?ormsc/14-06-01

[5] I. Sommerville, “Ingeniería de Software”, 7ma. edición, Pearson, 2005. ISBN: 84-7829-074-5.

[6] “Domain-Specific Modeling,, Enabling Full Code Generation”, STEVEN KELLY, JUHA-PEKKA TOLVANEN, IEEE COMPUTER SOCIETY - Ed. 2008

[7] A Domain-Specific Language Toolkit, Richard C. Gronback - Adison Wesley - Ed. 2009.

[8] Especificación de FHIR de HL7, Realease 3. Disponible en: https://www.hl7.org/fhir

[9] OpenEHR Information Model, 2017. Disponible en: http://www.openehr.org/releases/RM/latest/docs/ehr/ehr.html

[10] Línea de Investigación del CEIS - Centro de Estudios de Ingeniería de Software http://www.ceisufro.cl/index.php?id=45

[11] UML, “Unified Modeling Language Infrastructure” (OMG). Versión 2.4.1, 2011. Disponible en http://www.omg.org/spec/UML/2.4/

[12] Ariste Cecilia, Ponisio Julieta, Nahuel Leopoldo, Giandini Roxana. JAIIO - ASSE (2015). “Diseñando Transformaciones de Modelos CIM / PIM: desde un enfoque de negocio hacia un enfoque de sistema”.

[13] Informe técnico “Especificación de la Transformación de Proceso BPD en BPMN a Diagrama de Actividades UML” PID MAPS 2015. Disponible en: http://maps.frlp.utn.edu.ar/

[14] Giandini, Roxana, Nahuel, Leopoldo, Roca, Leandro, Caputi, Matías, Zugnoni, Ivan - CoNaIISI (2014). “Implementando Transformación de Modelos utilizando MOSKitt Tool en adhesión al Paradigma MDD”. Congreso Nacional de Ingeniería en Informática/Sistemas de Información.

[15] L. Nahuel, E. Santanera, M. C. Ariste, L. Rocca, R. Giandini. Integración Metodológica para el Desarrollo de Tecnologías Software Dirigidas por Modelos y Basadas en Procesos de Negocio. CIINDET 2014.

[16] L. Nahuel, E. Santanera, L. Rocca, C. Ariste, R. Giandini. Aportes de las Tecnologías para Gestión de Procesos de Negocio al Desarrollo de Software Dirigido por Modelos. HCITISI 2013, (ISBN 978.88.96.471.25.8).

[17] Dave Steinberg, Frank Budinsky, Marcelo Paternostro y Ed Merks."EMF: Eclipse Modeling Framework". Segunda Edición (2009). Addison-Wesley Professional. ISBN-10: 0-321-33188-5

[18] Tutorial de EMF. http://www.vogella.com/articles/EclipseEMF/article.html

[19] Tutorial de GMF. http://www.kermeta.org/docs/fr.irisa.triskell.kermeta.samples.fsm.documentation/build/html.chunked/KerMeta-Create-FSMGraphical-Editor-With-GMF/ch02.html