ISBN 978 -987-633-574-4

Tecnología CASE para Modelado Específico de Dominio en Sistemas de Información Sanitaria basado en Estándar de Interoperabilidad Clínica

Juan Cesaretti[1], Lucas Paganini[1], Arián Calabrese[1], Martín Lunasco[1], Leandro Rocca [1], Leopoldo Nahuel[1], Roxana Giandini[1,2],

[1] GIDAS, Grupo de I&D Aplicado a Sistemas informáticos y computacionales, UTN - FRLP, La Plata, Argentina
[2] CIC, Centro de Investigación, LIFIA, UNLP - Facultad de Informática, La Plata, Argentina
{jcesaretti, lpaganini, acalabrese, mlunasco, leorocca, lnahuel, rgiandini}@frlp.utn.edu.ar

Abstract. Los sistemas de información sanitaria plantean dos grandes retos: por un lado, deben adaptarse a las constantes actualizaciones tecnológicas, y por otro, deben posibilitar la integración de toda la información y su disponibilidad en cada punto en que se necesite acceder a ella. La primera dificultad se abordó con el enfoque del Modelado Específico de Dominio (DSM). El poder de abstracción que provee el DSM permite a los ingenieros de software manejar la complejidad creciente de una manera rápida y clara. Por eso resulta beneficioso disponer de un Lenguaje Específico de Dominio (DSL) como el que aquí se propone: SIS_Static, complementado por un DSL dinámico: SIS_Dynamic. Para solucionar el segundo problema (comunicación entre distintos sistemas), se utilizó como referencia un estándar de interoperabilidad clínica: FHIR. Así, se implementó una herramienta de software basada en DSM que permite crear especificaciones gráficas de alto nivel y producir código fuente de manera automatizada, en distintos lenguajes de programación.

Keywords: Modelado específico de dominio (DSM), lenguaje específico de dominio (DSL), Fast healthcare interoperability resources (FHIR), desarrollo dirigido por modelos (MDD).

1 Introducción

La Ingeniería Dirigida por Modelos (MDE: Model-Driven Engineering) se enfoca fuertemente en los modelos, a los que considera los elementos centrales de la Ingeniería de Software.

Un modelo es una representación simplificada de la realidad, desacoplada de los detalles de implementación. Esto provee una gran adaptabilidad frente a la evolución de las tecnologías utilizadas. En particular, el Desarrollo de Software Dirigido por Modelos [1], [2] abarca todas las propuestas y mecanismos para producir aplicaciones a partir de la transformación de modelos, proporcionando un alto nivel de abstracción.

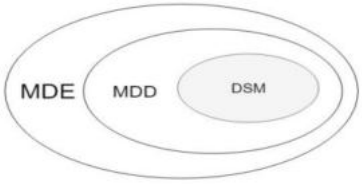

Fig. 1. Especialización de enfoques de la MDE

ISBN 978 -987-633-574-4

El Modelado Específico del Dominio o DSM (del inglés: Domain Specific Modeling), es una disciplina que, en el contexto del MDD, trabaja con lenguajes propios, restringidos a cada dominio o ámbito de interés. Son los llamados Lenguajes Específicos de Dominio, también conocidos por su acrónimo: DSL (Domain-Specific Language). Esta especialización permite una mayor automatización, que no podría lograrse usando un lenguaje de modelado de propósito general, como UML [17]. En la Fig. 1 se representa con un diagrama de Venn la relación de inclusión del enfoque DSM en el MDD, en el encuadre más general de la MDE.

Un modelo puede constituirse con diagramas, utilizando símbolos gráficos definidos en un lenguaje de modelado. En un DSL, cada bloque de construcción representa algún concepto propio del dominio considerado. Luego, este tipo de lenguaje es más claro y manejable para los usuarios, y las particularidades tecnológicas son transparentes para los mismos [1].

La sintaxis abstracta de un lenguaje gráfico de modelado se define en un metamodelo que especifica los elementos de modelado, las relaciones entre ellos y las reglas de buena formación de los modelos. Dado que un metamodelo es también un modelo, debe estar expresado en un lenguaje bien definido, conocido como metalenguaje. Este determina qué elementos pueden ser usados y cómo pueden vincularse. Algunos metalenguajes conocidos son: MOF (Meta-Object Facility) [3], ECORE [4] y GOPPRR [5].

La sintaxis concreta de un lenguaje gráfico determina el aspecto visual del mismo, estableciendo una colección de símbolos que pueden utilizarse para construir los diagramas.

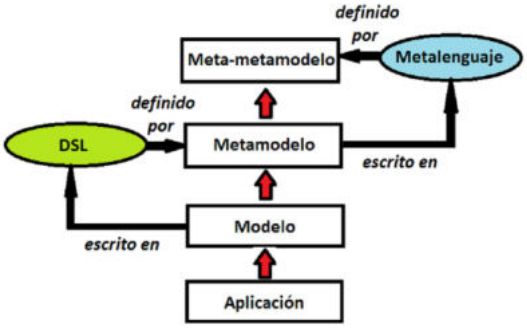

Fig. 2. Arquitectura de cuatro niveles de abstracción

La organización OMG propone una arquitectura de cuatro niveles, como muestra la Fig. 2:

Nivel M0 (Aplicación): Consta de los objetos instanciados por la aplicación.

Nivel M1 (Modelo): En el que se dibujan los diagramas para modelar el sistema.

Nivel M2 (Metamodelo): Es donde se definen los bloques de construcción (elementos y relaciones) de los lenguajes de modelado, utilizando un metalenguaje.

Nivel M3 (Meta-metamodelo): En el que se define la sintaxis abstracta de cada metalenguaje.

La solución que planteamos parte de dos vistas, una estática y otra dinámica. Se desarrolló un DSL para construir cada una de ellas: SIS_Static y SIS_Dynamic, respectivamente. Los metamodelos (nivel M2) en los que se sustentan estos lenguajes se expresaron en GOPPRR, que es el metalenguaje que emplea MetaEdit+ [6], la herramienta Meta CASE que utilizamos. La expresividad de los lenguajes que presentamos fue validada con distintos modelos (nivel M1), como el que se expone en la sección 4, a modo de ejemplo de aplicación.

Un dominio con una complejidad creciente es el de los Sistemas de Información Sanitarios (SIS). En los países avanzados, la esperanza de vida va en aumento. Y las personas mayores padecen distintas enfermedades crónicas, debido al sedentarismo, la obesidad y otros problemas causados por su estilo de vida. Se estima que el 17% de los pacientes en EE.UU. tienen más de seis afecciones crónicas. Así, una misma persona

ISBN 978 -987-633-574-4

consulta a distintos especialistas, y se plantea la necesidad de integrar toda la información registrada por ellos. Y esta información de atención médica debe ser accesible desde distintas organizaciones y puntos geográficos, en virtud de la movilidad de los pacientes [7]. Además, los SIS necesitan adecuarse constantemente a las nuevas tecnologías. Por lo tanto, el abordaje del DSM resulta muy conveniente en este caso.

Para intercambiar la información entre distintos sistemas de gestión sanitaria, existen estándares de interoperabilidad que definen la estructura de los datos, para que puedan compartirse [8]. Diversas organizaciones se ocupan de uniformar criterios de interoperabilidad, como ser: HL7 Internacional, HIMMS o NEMA.

El último estándar de interoperabilidad clínica de HL7 es FHIR (Fast Healthcare Interoperability Resources) [9]. Es de código abierto, y amalgama lo mejor de los estándares más utilizados en la actualidad. El elemento fundamental de FHIR es el "recurso", definido como la unidad básica de interoperabilidad. Cada recurso especifica un concepto del dominio sanitario: paciente, médico, problema de salud, entre otros.

Este trabajo presenta una solución DSM basada en FHIR. A través de ella, un ingeniero de software, aún sin tener un conocimiento previo de FHIR, puede crear diagramas compatibles con dicho estándar. Y esas especificaciones gráficas con alto nivel de abstracción, pueden ser transformadas en código de manera automatizada.

El presente artículo se organiza de la siguiente manera: en la Sección 2 se presentan la definición de los DSLs y cómo se construyó un editor basado en los mismos, en la Sección 3 se detalla la aplicación de la herramienta en un caso de estudio, en la Sección 4 se describen los antecedentes y los avances, y finalmente, en la Sección 5, se abordan las conclusiones y líneas de trabajo futuro.

2 Desarrollo

En esta sección se abordará el detalle de los dos DSLs diseñados cada una de las vistas construidas (estática y dinámica), el editor gráfico que se construyó como herramienta de modelado y los mecanismos utilizados para lograr generar código fuente a partir de modelos construidos con los DSLs.

2.1 Vista Estática

En primer lugar, se definió un DSL para modelar los aspectos estructurales de los sistemas de información sanitaria, al que denominamos SIS_Static. Sentamos sus bases en un estándar específico del dominio: FHIR. Del mismo, se seleccionó un subconjunto relevante de recursos. Cada uno de estos recursos fue tomado como un bloque de construcción del lenguaje de modelado SIS_Static, a saber:

Patient: Paciente, sujeto que recibe atención sanitaria.

Practitioner: Personal sanitario que provee servicios de atención médica, de enfermería u otras áreas afines.

Person: Abstracción que agrupa la información demográfica, tanto de pacientes como del personal sanitario.

Organization: Organización dentro de la que se proveen servicios sanitarios (hospital, clínica, sala de atención primaria, etc.).

Encounter: Encuentro entre un paciente y personal sanitario, por ejemplo: una práctica de salud (curación, rehabilitación, colocación de férulas y yesos, entre otras), una consulta ambulatoria, entre otras.

Episode of care: Episodio de atención, o asociación temporal entre una organización sanitaria responsable y un paciente, durante la cual pueden ocurrir uno o más encuentros.

ISBN 978 -987-633-574-4

Las propiedades de estos elementos también fueron tomadas de FHIR, al igual que la única relación válida entre dichos elementos: “Reference”. Asimismo, los nombres de los roles con que cada elemento participa en una relación se extrajeron del mismo estándar.

Este subconjunto de elementos escogidos permite modelar sistemas básicos de información sanitaria, para gestionar atenciones ambulatorias de demanda espontánea, servicios de guardia e internaciones no programadas. Y puede ser extendido fácilmente para aumentar el alcance de los sistemas modelados.

La Fig. 3 muestra el metamodelo en GOPPRR que define la sintaxis abstracta del lenguaje SIS_Static. La semántica de cada elemento es la misma que la especificada para los recursos correspondientes de FHIR.

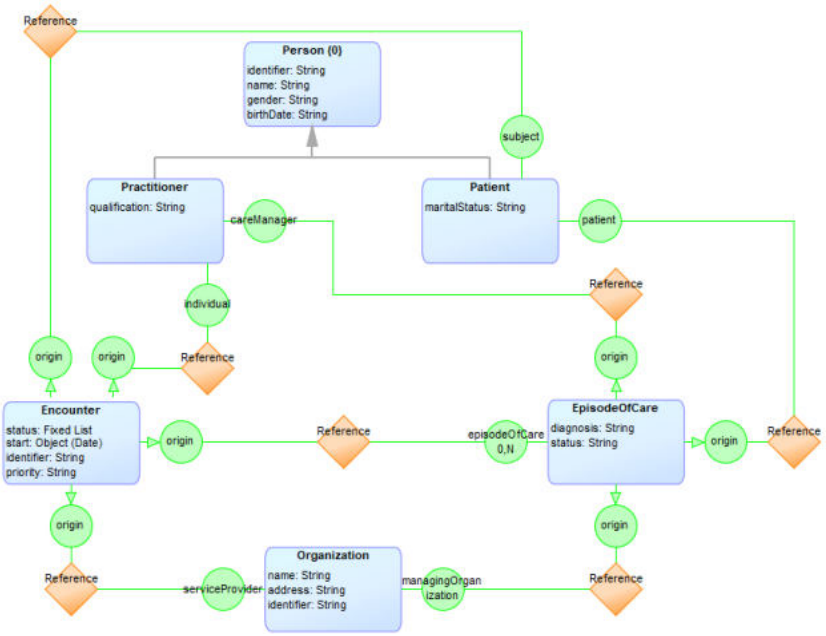

Fig. 3. Metamodelo en GOPPRR del DSL SIS_Static

2.2 Herramienta DSL

Para poder crear, visualizar y editar modelos, usando el lenguaje SIS_Static, se construyó un editor gráfico. Para ello, se utilizó el metaeditor MetaEdit+.

La sintaxis concreta del DSL se definió a través del editor de símbolos que está integrado a dicho metaeditor. La Fig. 4 presenta la paleta de elementos y relaciones del DSL SIS_Static. Los elementos pueden arrastrarse hasta el espacio de trabajo del diagrama, y pueden vincularse entre sí. La herramienta detecta e impide que se formen relaciones ilegales. Esto se debe a que las reglas de correctitud del dominio se materializaron por medio de “bindings”, que establecen las conexiones válidas entre elementos. El metaeditor también permite almacenar otras restricciones de conectividad, ocurrencia, unicidad, etc.

Fig. 4. Paleta de elementos y relaciones construídas para SIS_Static

ISBN 978 -987-633-574-4

2.3 Generación automática de código

Para transformar a texto (código fuente de un lenguaje de programación) los diagramas construidos con el DSL presentado, se utilizó el editor generado que MetaEdit+ posee integrado. El mismo cuenta con un lenguaje propio: MERL (MetaEdit Report Language). Este permite recorrer los diagramas, y producir una salida de texto, a partir de la información extraída de los elementos, sus propiedades, sus roles y relaciones por las que va navegando.

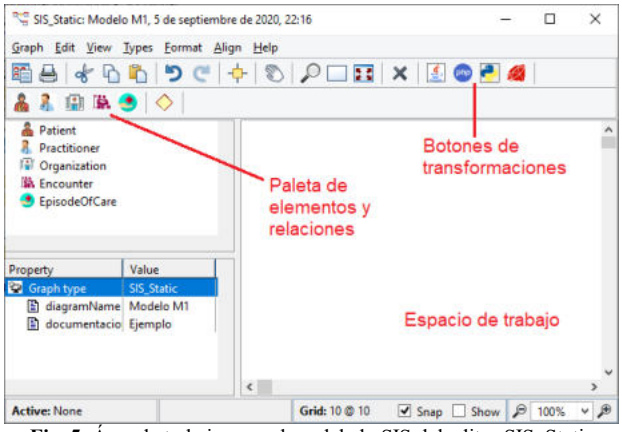

Fig. 5. Área de trabajo para el modelado SIS del editor SIS_Static

En la Fig. 5 se muestra la ventana principal del editor, con la paleta de elementos y relaciones, y los botones añadidos a la barra de herramientas para generar código automáticamente en distintos lenguajes de programación, como: Java, Php, Python y Ruby.

Aprovechando el generador integrado que posee MetaEdit+, se realizaron transformaciones de modelo a texto (ver Fig. 6) para producir código automáticamente, con sólo presionar un botón de la barra de herramientas (Botones de transformaciones en Fig. 5). Esto fue posible gracias a un lenguaje propio del metaeditor: MERL (MetaEdit Report Language), que permite navegar por los elementos de los diagramas, generando salidas con distintos formatos.

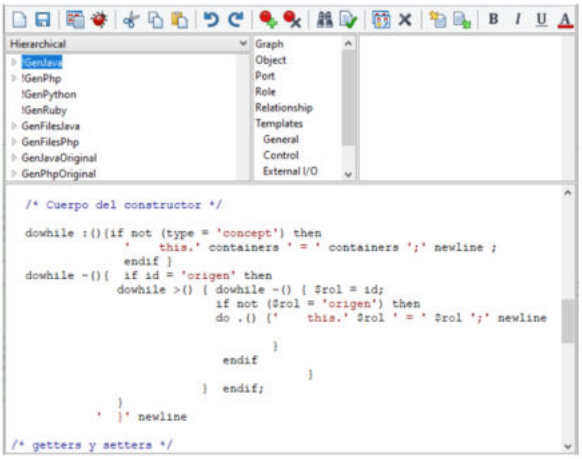

Fig. 6. Generador de código de MetaEdit+ a partir de modelos construidos con el editor SIS_Static

ISBN 978 -987-633-574-4

En la Fig. 7 se muestra un archivo de salida, obtenido a partir de un diagrama.

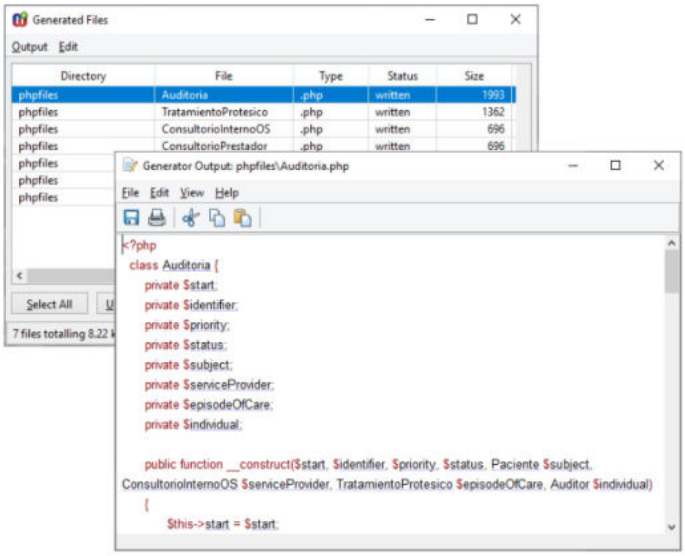

Fig. 7. Código fuente generado a partir de modelos gráficos construidos con el editor SIS_Static

2.4 Vista dinámica

Por último, se definió un DSL para capturar aspectos de comportamiento del dominio de los sistemas de información sanitaria. Lo denominamos SIS_Dynamic, y está basado en el enfoque de las máquinas de estado de UML [17]. Permite representar los diferentes estados por los que puede transitar un "Encounter" (Encuentro entre un paciente y personal sanitario), que es el elemento neurálgico de los modelos de este dominio acotado. Dichos estados se restringieron a una lista de valores posibles, que se halla especificada en FHIR.

Los elementos del lenguaje SIS_Dynamic son:

State [Encounter]: Se trata de cada uno de los estados por los que puede pasar un encuentro (planned, arrived, triaged, in-progress, onleave, cancelled, finished, entered-in-error y unknown). La herramienta solamente permite seleccionar uno de esos valores, definidos en FHIR. Y se agregó una propiedad "name", para describir el estado en lenguaje natural, para que sea más intuitivo.

InitialState: Es el estado temporal de inicio. Al igual que en UML, se restringe a una sola instancia por diagrama.

FinalState: Es el estado final, que cierra el ciclo de vida.

Action: Es una acción desencadenada al producirse la transición a la que se asocia. De ella puede extraerse información para agregar funcionalidad al código generado a partir de SIS_Static.

ISBN 978 -987-633-574-4

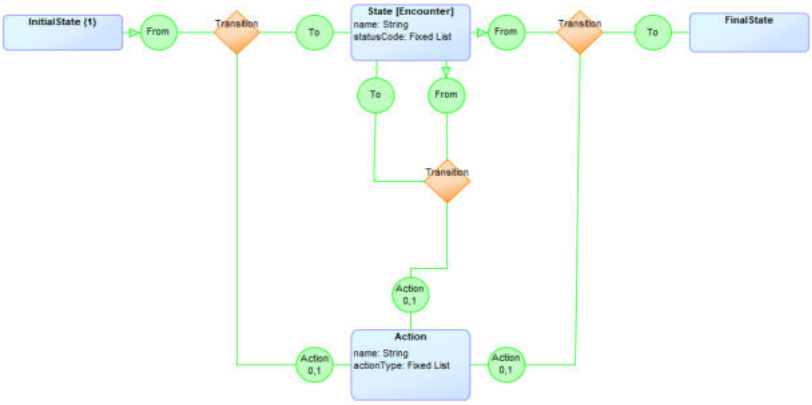

Fig. 8. Metamodelo en GOPPRR del DSL SIS_Dynamic

La única relación válida entre estos elementos es: "Transition" (transición). Esta representa el pasaje de un estado a otro.

La Fig. 8 muestra el metamodelo de SIS_Dynamic en GOPPRR, que define su sintaxis abstracta.

Del mismo modo que se hizo con la vista estática, se construyó un editor gráfico en MetaEdit+. La Fig. 9 presenta la paleta correspondiente de elementos y relaciones.

Fig. 9. Paleta de elementos y relaciones de SIS_Dynamic

Los metamodelos de los DSL SIS_Static y SIS_Dynamic se vincularon mediante una estructura de explosión. La Fig. 10 muestra cómo un tipo de objeto Encounter, componente de un diagrama estático en SIS_Static, puede ser refinado en otro diagrama más específico: un diagrama dinámico en SIS_Dynamic, que detalla sus cambios de estado y su transición.

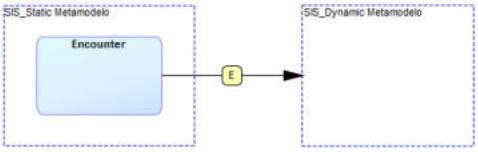

Fig. 10. Relación entre la vista estática y la vista dinámica

3 Resultados obtenidos a partir de un caso de estudio

Para probar el poder expresivo y la suficiencia de la solución DSM propuesta, se modeló la atención de pacientes en una Unidad Febril de Urgencia (UFU). Estas unidades son espacios anexados por el Ministerio de Salud de la ciudad de Buenos Aires a gran parte de los hospitales de agudos y pediátricos, en el contexto de la pandemia de COVID-19 [19].

Cuando una persona, presuntamente contagiada de COVID-19, se presenta en una UFU, primero es atendido por un enfermero. Si este determina que se trata de una emergencia, el paciente se remite directamente a la guardia. Si no, luego de llevar a cabo el protocolo de triage, el enfermero lo deriva a un médico. Este último lo evalúa de nuevo, y si resuelve que es una emergencia, lo envía a la guardia. Si no, le realiza un examen médico y epidemiológico, y registra los resultados en un informe. Luego es derivado a una Unidad Transitoria de Aislamiento (UTA), donde aguarda de forma segura hasta ser trasladado a un

ISBN 978 -987-633-574-4

hotel, una sala de hospital, o a una Unidad de Tratamientos Intensivos (UTI), según la gravedad del cuadro de la persona.

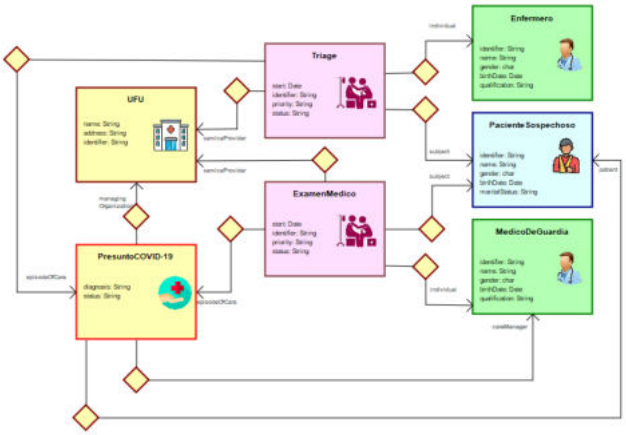

Fig. 11. Diagrama estático de Atención en UFU realizado con el DSL SIS_Static

La Fig. 11 muestra el diagrama estático, realizado con el lenguaje SIS_Static. El triage llevado a cabo por un enfermero, y el examen realizado por un médico, se representan con dos elementos de tipo Encounter. Ambos están referidos a un mismo Episode of care, que es la presunción de contagio del virus.

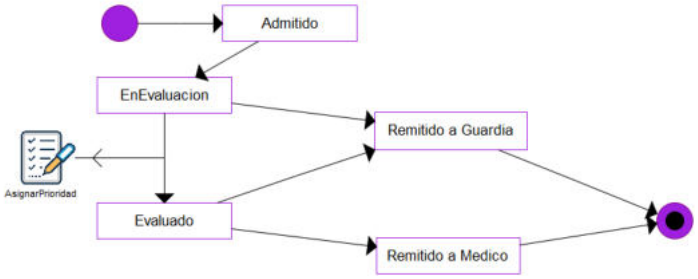

Fig. 12. Diagrama dinámico del Triage en una UFU realizado con el DSL

En la Fig. 12 se presenta un diagrama dinámico correspondiente al Encounter del Triage efectuado por un enfermero.

Los diagramas y archivos de código fuente generados automáticamente, para éste y otros casos de estudio, se encuentran disponibles en un repositorio [20]. Para conocer más detalles de la solución y tecnología utilizada, se ofrece un video demostrativo [21].

4 Discusión

Existe un metamodelo de HL7, llamado MIF (Model Interchange Format) [10], el cual es muy extenso y complejo, y cuyo aprendizaje significa un costo muy alto para los ingenieros de software. Además, MIF tiene múltiples versiones coexistentes, y esto puede causar problemas de incompatibilidad [11]. Por otra parte, algunos investigadores de HL7 Internacional han publicado un perfil de UML ajustado a MIF [12]. No obstante, el propósito de este trabajo es presentar una solución DSM, con un metamodelo más simple, fundado en un subconjunto relevante de recursos de FHIR. Se trata de una vista estática complementada con otra dinámica, que permite adicionar funcionalidad a los sistemas de información modelados.

Se han publicado experiencias previas en el uso de HL7 en el marco de MDE [13], [14]. Pero en ninguna de ellas se produjo una herramienta para automatizar la generación de código fuente. Esto se debe a que su propósito fue extender o especializar un lenguaje de propósito general como UML, o lograr transformaciones de modelos entre HL7 y UML. Con nuestra solución DSM, es posible esta automatización, siendo uno de los aspectos más significativos de esta propuesta.

En trabajos previos [15], [16], se desarrolló un DSL acotado para este dominio, a través de un metamodelo construido con el metalenguaje ECORE, reutilizando algunos elementos del metamodelo de UML [17]. Se implementó un editor gráfico basado en el DSL propuesto. Para ello se utilizó Sirius [18], un framework de código abierto, que aprovecha las tecnologías EMF y GMF de la plataforma Eclipse. Este editor solamente permitía crear, visualizar y modificar diagramas, respetando la sintaxis del DSL, pero no contaba con la posibilidad de realizar transformaciones de modelo a código fuente.

5 Conclusiones

En primer lugar, destacamos las fortalezas de la solución DSM propuesta. Se probó la expresividad del editor construido, modelando algunos casos concretos de situaciones reales y de interés actual por la pandemia Covid-19. Uno de esos casos (atención de pacientes en Unidad Febril de Urgencia en contexto de evaluación de contagios Covid-19) fue presentado en la sección anterior, mostrando vistas: estática y dinámica. Además, se realizaron transformaciones de modelo a texto, para obtener código en diferentes lenguajes de programación, de manera automatizada.

En cuanto al DSL SIS_Static, podemos observar que ofrece un mecanismo de abstracción muy adecuado, para manejar la complejidad del dominio. Es muy útil para simplificar el trabajo de los analistas de negocios y diseñadores de sistemas, en las primeras fases del proceso de desarrollo del software. Cada elemento de la paleta de edición se identifica con algún objeto reconocible del dominio, y es además una unidad de interoperabilidad.

Otro aspecto importante es que el editor verifica automáticamente los modelos construidos y detecta tempranamente posibles errores en el proceso de modelado, evitando la creación de especificaciones ilegales o no deseadas (según las reglas de negocio del dominio). Esto último constituye un aporte significativo en automatizar las reglas de buena formación de relaciones válidas entre los elementos del dominio.

El DSL SIS_Dynamic, por su parte, se adicionó para capturar el comportamiento funcional de los sistemas de información, permitiendo refinar elementos definidos en la vista estática, aumentando así las capacidades de la herramienta propuesta.

También es importante subrayar las ventajas observadas en el metalenguaje y el metaeditor utilizados. GOPPRR, al ser diseñado específicamente para describir lenguajes de modelado, aporta los mismos beneficios que usar DSM en cualquier dominio, contribuyendo principalmente en la sencillez y precisión. Y en MetaEdit+ se observan ventajas comparativas, con respecto a otros modeladores. Comparándolo con las herramientas utilizadas anteriormente en la plataforma Eclipse, se puede afirmar que el tiempo invertido para construir el editor es notablemente menor en MetaEdit+. Además, su entorno es mucho más intuitivo y simple. Ofrece un buen soporte a la evolución del lenguaje, facilitando la propagación de cambios del metamodelo a sus instancias. Y posee un editor de símbolos y un generador de código, ambos integrados en la herramienta, lo que resulta de suma utilidad para el diseño e implementación de un DSM.

Existen antecedentes de proyectos que abordaron el mismo dominio desde la perspectiva de MDE, tomando como base algún estándar de HL7. Sin embargo, ninguno de ellos se realizó en el marco del DSM, para explotar los beneficios que ofrece frente a un proceso tradicional de desarrollo de sistemas informáticos: ocultamiento de la complejidad y automatización.

Como trabajo futuro, dando continuidad a este proyecto, se plantea adicionar a las vistas estática y dinámica, una vista de interfaz gráfica de usuario (DSL SIS_GUI). También se procura agregar eventos a los diagramas dinámicos, y perfeccionar la definición de los elementos de tipo Action. Al mismo tiempo, se espera realizar un test de usabilidad y calidad interna del entorno construido: herramienta de software DSM para SIS.

Referencias

1. J. García Molina, F. O. García Rubio, V. Pelechano, A. Vallecillo, J. M. Vara y C. Vicente-Chicote, (2013). "Desarrollo de software dirigido por modelos: Conceptos, métodos y herramientas". Madrid, España: Ra-Ma
2. Kelly, S., Tolvanen, J., (2008). Domain-Specific Modeling: Enable Full Code Generation. Hobaken, Estados Unidos: Wiley-IEEE Computer Society
3. Especificación de MOF (MetaObject Facility). Recuperado de https://www.omg.org/mof/
4. Definición del Package ECORE. Recuperado de http://download.eclipse.org/modeling/emf/emf/javadoc/2.6.0/org/eclipse/emf/ecore/package-summary.html
5. Especificaciones del lenguaje de modelado de GOPPRR. Recuperado de https://www.metacase.com/support/45/manuals/mwb/Mw-1_1.html
6. MetaEdit+ Workbench – Build your own modeling tool. Recuperado de https://www.metacase.com/mwb/
7. M. L. Braunstein., (2018). "Health Care in the Age of Interoperability: The Potential and Challenges", IEEE Pulse, vol(9), 34-36. doi: 10.1109/MPUL.2018.2856941
8. Foro de la OMS sobre la Estandarización y la Interoperabilidad de los Datos Sanitarios. (2012). Organización Mundial de la Salud. Ginebra, Siuza.
9. Resourcelist – FHIR v4.0. Recuperado de https://hl7.org/fhir/resourcelist.html
10. Spronk, R., Ringholm, C. (2010). The HL7 MIF - Model Interchange Format. Recuperado de http://www.ringholm.com/docs/03060_en_HL7_MIF.htm
11. Villegas, A., Olivé, A. (2013). UML Profile for MIF Static Models. Version 1.0. Recuperado de http://www.vico.org/HL7_Tooling/Submission/MIF.pdf
12. Model Interchange Format, HL7. Recuperado de https://wiki.hl7.org/index.php?title=Model_Interchange_Format
13. E. R. Pfaff et al. (2019). "Fast Healthcare Interoperability Resources (FHIR) as a Meta Model to Integrate Common Data Models: Development of a Tool and Quantitative Validation Study". JMIR medical informatics, vol. (7) doi:10.2196/15199
14. M. A. Olivero, F. J. Domínguez-Mayo, C. L. Parra-Calderón, M. J. Escalona, y A. Martínez-García. (2020). "Facilitating the design of HL7 domain models through a model-driven solution", BMC medical informatics and decision making, vol. (20). doi:10.1186/s12911-020-1093-4
15. Cesaretti, J., Paganini, L., Rocca, L., Caputti, M., Zugnoni, I. (2019). Herramienta basada en Lenguaje Específico de Dominio para Sistemas elementales de Información Sanitaria. JAIIO 48. Salta, Argentina.
16. Rocca, L., Caputti, M., Zugnoni, I., Paganini, L., Cesaretti, J., Nahuel, L., Giandini, R. (2018). Marco de trabajo para el diseño y desarrollo de Herramientas de modelado conceptual basado en DSL utilizando tecnologías GMF. CIITI. Buenos Aires, Argentina.
17. OMG Unified Modeling Language (OMG UML). Version 2.5.1. December 2017.Recuperado de https://www.omg.org/spec/UML/2.5.1/PDF
18. Sirius – The easiest way to get your own Modeling Tool. Recuperado de https://www.eclipse.org/sirius/
19. Unidades Febriles de Urgencia (UFU). Recuperado de https://www.buenosaires.gob.ar/coronavirus/unidades-febriles-de-urgencia-ufu
20. Repositorio de modelos y código generado para casos de estudio. Recuperado de https://drive.google.com/drive/folders/1_9pxCOqDRH8qtxNVAKuyLlir_-R8zpGG
21. DSL-SIS Un Lenguaje de Modelado Especifico del Dominio de Sistemas de Información Sanitaria. Recuperado de https://youtu.be/WTs6f_ZNqf8