Herramienta basada en Lenguaje Específico de Dominio para Sistemas elementales de Información Sanitaria

Juan Cesaretti[1] , Lucas José Paganini[1], Leandro Rocca[1], Matías Caputti[1], Iván Zugnoni[1]

[1] GIDAS - Grupo de Investigación del Departamento de Sistemas de Información.
UTN Facultad Regional La Plata.
Calle 126 e/ 58 y 60. CP(1924) Berisso. Te./Fax: (0221) 412-4350 / 412-4315,

{ juangces, lucasjpaganini, leorocca,
matias.caputti, zugnoni.ivan }@gmail.com

Abstract. Un DSL (por su nombre en inglés: Domain-Specific Language) permite modelar soluciones usando directamente conceptos propios del ámbito del problema. En este nivel de abstracción, se ignoran los detalles de implementación, simplificando la tarea del analista/diseñador de software. Se presenta el lenguaje DSLSalud, aplicable al modelado de sistemas básicos de información sanitaria, a través de bloques de construcción que se identifican de forma clara y sencilla con elementos del dominio. Para definir esos elementos, se tomó como base la especificación de FHIR (Fast Healthcare Interoperability Resource), último estándar abierto de interoperabilidad clínica desarrollado por la organización internacional HL7 (Health Level Seven). Esta característica de nuestro DSL pretende facilitar el intercambio de datos y la comunicación entre distintos sistemas de información sanitaria. Primeramente se definió la sintaxis abstracta del lenguaje DSLSalud con un metamodelo y luego se implementó un editor gráfico en la plataforma Eclipse. Dicho editor fue creado utilizando Sirius, un framework de código abierto que aprovecha las tecnologías EMF (Eclipse Modeling Framework) y GMF (Graphical Modeling Framework). Con esta herramienta, se puede generar, visualizar y modificar diferentes modelos, sobre los que pueden aplicarse transformaciones de modelo a texto para obtener automáticamente código ejecutable.

1 Introducción

La Ingeniería Dirigida por Modelos, cuyo acrónimo es MDE (Model-Driven Engineering), es una disciplina dentro de la Ingeniería de Software que propugna el uso sistemático de modelos como artefactos fundamentales para mejorar la producción, el mantenimiento y otros aspectos de la calidad del software. Las técnicas que propone para construir nuevas aplicaciones (ingeniería directa) se engloban en lo que llamamos MDD (Model-Driven Development) o Desarrollo de Software Dirigido por modelos. Históricamente, los modelos se han usado como parte de la documentación,

pero desde la perspectiva del MDD tienen un rol central. El foco de atención no está en el código sino en los modelos [1], [2], [3].

Un modelo es una representación simplificada de la realidad. Es una abstracción que capta solamente los aspectos esenciales del sistema en estudio. Como suprime los detalles irrelevantes, permite reducir y manejar la complejidad. Cuando se modela, únicamente quedan visibles los elementos y relaciones significativos, y lo demás se oculta. Es así que, frente a la continua evolución de las tecnologías, la iniciativa MDD ofrece una gran ventaja: la adaptabilidad. Como los modelos son independientes de los detalles de implementación, aportan la flexibilidad necesaria para ajustarse a los cambios que pueda experimentar la plataforma tecnológica del software desarrollado [3].

La complejidad de los sistemas de información sanitaria ha ido creciendo con el tiempo [4]. Las tecnologías digitales y los conocimientos genómicos están cambiando profundamente las forma en que la medicina trata a los pacientes y a las enfermedades. Resulta entonces un dominio muy interesante para trabajar, en el marco del MDD. Ya que este abordaje permite gestionar la complejidad por su alto nivel de abstracción.

Un modelo puede representarse a través de diagramas, desde distintos puntos de vista, utilizando símbolos gráficos definidos en un lenguaje de modelado. UML (Unified Modeling Language) es el lenguaje de modelado de propósito general más difundido. Se trata de un estándar definido por el consorcio internacional OMG (Object Management Group) [5]. Pero el formato de su especificación dificulta la comprensión de su semántica [3]. Esta falta de precisión condujo al desarrollo de lenguajes más particulares y mejor definidos: los lenguajes específicos de dominio, conocidos como DSL (Domain-Specific Language).

Un DSL es un lenguaje enfocado en un área particular y bien acotada, donde cada elemento está estrechamente identificado con algún concepto propio de dicho dominio. Este lenguaje, por lo tanto, es más cercano y comprensible para los usuarios, y está lejos de los detalles de implementación [1].

Para definir un DSL, hay que determinar: su sintaxis abstracta, su sintaxis concreta y su semántica.

La sintaxis abstracta define los elementos del lenguaje, las relaciones entre ellos y las reglas de buena formación de los modelos. Dado que, generalmente, los lenguajes de modelado están basados en gráficos y no en texto, su sintaxis abstracta se especifica con un metamodelo. Un metamodelo es también un modelo y por lo tanto debe estar expresado en un lenguaje bien definido, conocido como metalenguaje. Este establece qué elementos pueden ser usados en el lenguaje y cómo pueden ser conectados. Los metalenguajes más conocidos son: MOF (Meta-Object Facility) y su variante, ECORE [1].

La sintaxis concreta del lenguaje queda determinada por el conjunto de símbolos que pueden utilizarse para dibujar los diagramas. Esta determina la notación y el aspecto visual de dicho lenguaje.

En la Fig. 1 se muestra la arquitectura de cuatro capas de modelado propuesta por la organización OMG. La misma consta de los siguientes 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.

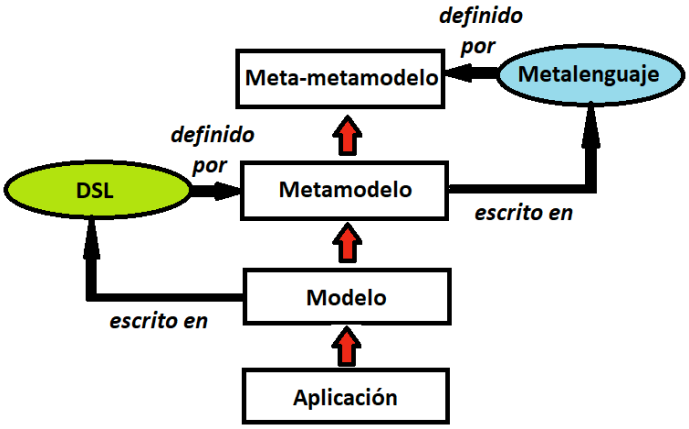

Fig. 1. Arquitectura de cuatro niveles del OMG

Volviendo al dominio de la gestión sanitaria, los datos de los pacientes deben estar disponibles a medida que este se mueve por el ecosistema de la salud digital. Para lograr la comunicación entre los distintos sistemas de salud se definen estándares de interoperabilidad que establecen el formato y estructura de los datos, para que puedan intercambiarse [6]. Hay distintas organizaciones que pretenden uniformar criterios de interoperabilidad: HL7 Internacional, HIMMS o NEMA.

FHIR (Fast Healthcare Interoperability Resources) [7] es el último estándar de HL7. Es de código abierto, y trata de combinar lo mejor de los estándares más utilizados en la actualidad. FHIR parte del concepto de *recurso*. Cada unidad básica de interoperabilidad es un recurso, y representa un concepto del dominio sanitario: paciente, médico, problema de salud, etc.

Los estándares de HL7 cuentan con un metamodelo común, llamado MIF (Model Interchange Format) [8]. Sin embargo, este es muy extenso y complejo, y su aprendizaje representa un costo muy alto para los ingenieros de software. También existe un Perfil de UML [9] adaptado a MIF, publicado por investigadores de HL7 Internacional. Pero el propósito de este trabajo es presentar un DSL básico, con un metamodelo más sencillo y amigable, basado en un subconjunto significativo de recursos de FHIR.

2 Lenguaje DSLSalud

En el presente proyecto, se restringió el dominio, de modo que puedan modelarse sistemas que denominamos elementales. Un sistema elemental de información sanitaria puede registrar atenciones médicas ambulatorias de demanda espontánea, guardia o emergencias e internaciones no programadas. Se excluyó la planificación de consultorios externos, internaciones e intervenciones quirúrgicas, al igual que el detalle completo de los datos clínicos de cada atención, diagnóstico por imagen o laboratorio, y las prescripciones de medicamentos.

Se desarrolló un DSL para este dominio acotado. Este DSL es de naturaleza gráfica. Su sintaxis abstracta se de definió con un metamodelo que se denominó Health, y fue construido con el metalenguaje ECORE [1], reutilizando algunos elementos del metamodelo de UML [5], como muestra la Fig. 2.

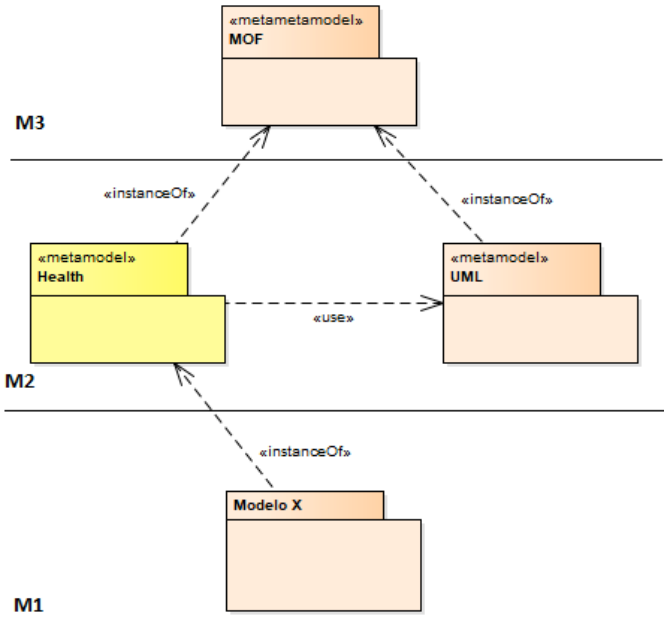

Fig. 2. Metamodelado. El modelo X instancia metaclases del metamodelo Health.

Cada elemento del metamodelo Health se corresponde con un recurso de la versión actual de FHIR, que es R4 (v4.0.0) [7]. A continuación se detalla el subconjunto de recursos seleccionados:

- *Patient* (Paciente): Sujeto receptor de los servicios de atención sanitaria.
- *Practitioner* (Profesional de la Salud): Cualquier persona que está directa o indirectamente involucrada en el aprovisionamiento de atención sanitaria. Puede ser médico, enfermero, etc.

- *Person* (Persona): Abstracción que reúne la información demográfica y administrativa de una persona, independiente del contexto sanitario.
- *Organization* (Organización): Establecimiento donde se proveen servicios sanitarios. Puede ser hospital, clínica, sala de atención primaria, etc.
- *Encounter* (Encuentro): Cualquier interacción entre un paciente y un proveedor de atención sanitaria. Puede ser una consulta ambulatoria, una práctica, etc.
- *Episode of Care* (Episodio de Atención): Asociación temporal entre una organización sanitaria responsable y un paciente, durante la cual pueden ocurrir encuentros. Se enfoca en una enfermedad o problema de salud particular. Varias organizaciones pueden estar involucradas en el proceso, pero cada una tendrá su propio Episodio de Atención, para realizar el seguimiento de su responsabilidad con el paciente. Esto permite separar los encuentros según el problema. Por ejemplo, si un paciente está siendo atendido por una enfermedad viral y por un traumatismo de rodilla, se mantendrán dos Episodios de Atención diferentes.

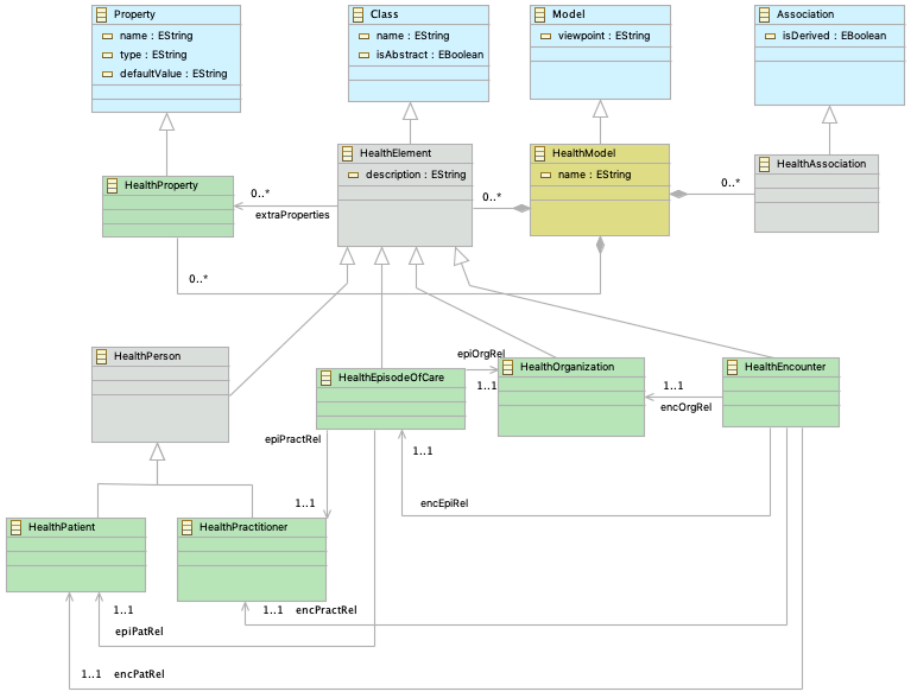

Fig. 3. Metamodelo del DSLSalud

En la Fig. 3 se presenta el metamodelo del DSLSalud, donde fueron incluidos elementos del metamodelo de UML 2.0 [5] para evitar redefinir metaclases útiles para nuestros diseños. Estos elementos se muestran en color celeste. Y los elementos del metamodelo Health se muestran en otros colores, y con el prefijo Health en sus nombres.

La metaclase HealthProperty permite representar los atributos de una clase, a través de asociaciones entre dicha clase y una instancia de HealthProperty por cada uno de esos atributos.

Las relaciones se basan en las referencias entre recursos de FHIR R4 [7]. Y sus nombres se formaron uniendo la abreviatura (las tres primeras letras) del elemento origen con la abreviatura del elemento destino, y agregando el sufijo *Rel*. Así, por ejemplo, la relación entre un HealthEncounter y el HealthPatient atendido en el mismo, es denominada *EncPatRel*, y es navegable solamente en esa dirección.

3 Editor

Se desarrolló un editor gráfico basado en el DSL propuesto: DSLSalud. Para ello se utilizó Sirius, un framework de código abierto, que aprovecha las tecnologías EMF y GMF de la plataforma Eclipse [10].

En las Figs. 4 y 5, se muestra cómo se crea un nuevo diagrama sobre un proyecto DSLSalud.

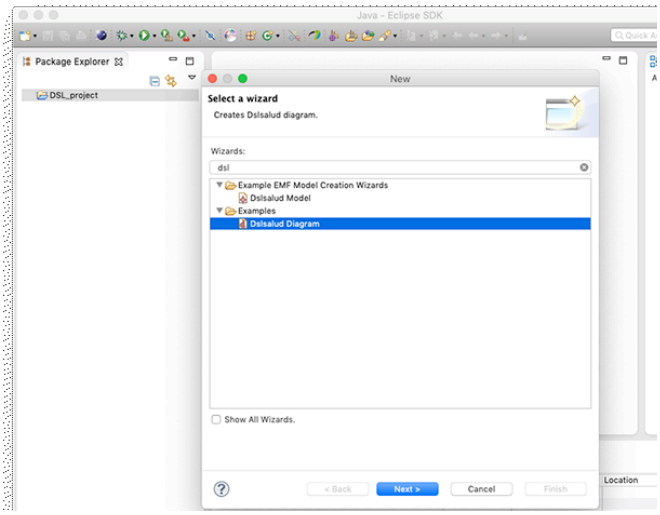

Fig. 4. Creación de un nuevo diagrama. Paso 1

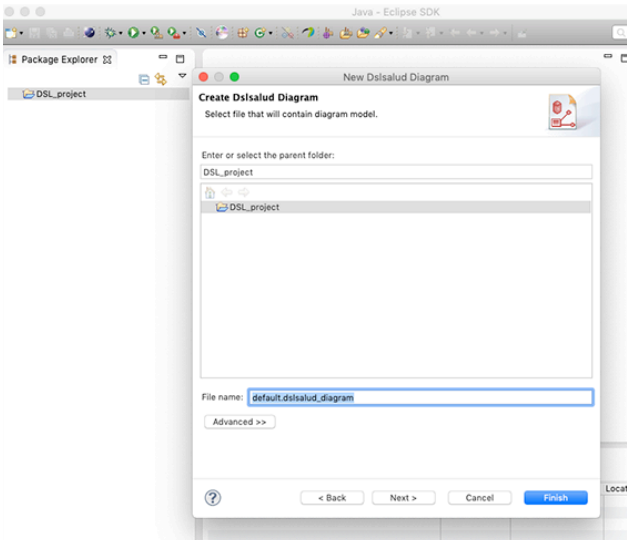

Fig. 5. Creación de un nuevo diagrama. Paso 2

En la Fig. 6, puede verse el paño sobre el que se construye el diagrama. A la derecha, la paleta de elementos y relaciones del DSLSalud. Y un panel inferior para editar los atributos de cada elemento. En esta implementación, todos los nombres de los elementos están en inglés y con el prefijo Health.

Para graficar un diagrama, se selecciona un bloque de construcción (elemento o relación) y se arrastra hasta el espacio de modelado.

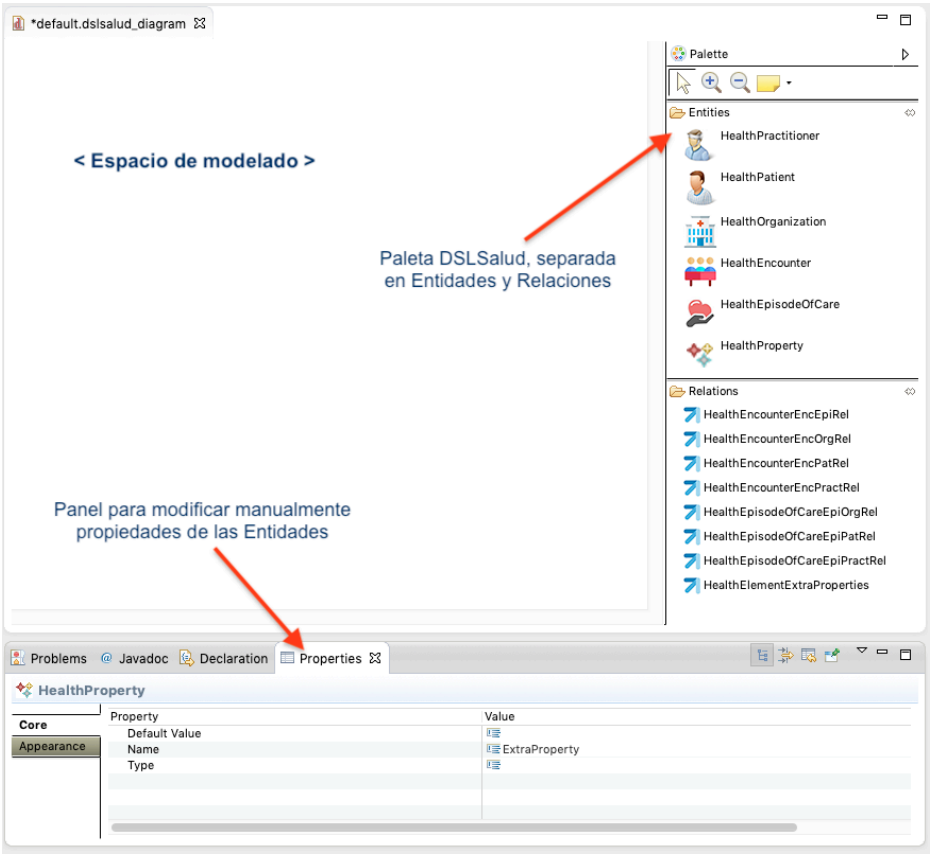

Fig. 6. Editor para el DSLSalud

En la Fig. 7, se modeló una Consulta (instancia de HealthEncounter), y se vinculó a una Clínica Privada, a un Paciente y al Médico que lo atiende (instancias de HealthOrganization, HealthPatient y HealthPractitioner, respectivamente). Además, se agregó el atributo Obra Social al Paciente, asociando la instancia de HealthPatient con una instancia de HealthProperty.

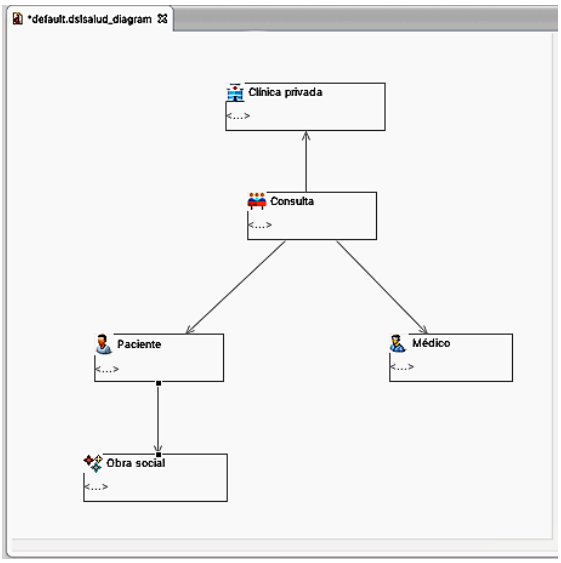

Fig. 7. Ejemplo de modelo M1

En la Fig. 8, se muestra el código XML generado al crear el modelo M1 de la Fig. 7.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- definión M1 a partir de DSL SALUD M2 -->
<dslsalud:HealthModel xmi:version="2.0"
  xmlns:xmi="http://www.omg.org/XMI"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:dslsalud="gidas.frlp.utn.edu.ar">

  <!-- entidad HealthOrganization  -->
  <hasElements xsi:type="dslsalud:HealthOrganization"
    name="Clínica privada"/>

  <!-- entidad HealthPatient, con propiedad extra "Obra social" -->
  <hasElements xsi:type="dslsalud:HealthPatient"
    name="Paciente"
    extraProperties="//@hasProperties.0"/>

  <!-- entidad HealthEncounter  -->
  <hasElements xsi:type="dslsalud:HealthEncounter"
    name="Consulta"
    encOrgRel="//@hasElements.0"
    encPatRel="//@hasElements.1"
    encPractRel="//@hasElements.3"/>

  <!-- entidad HealthPractitioner -->
  <hasElements xsi:type="dslsalud:HealthPractitioner"
  name="Médico"/>

  <!-- propiedad extra para asignar a entidad HealthPatient -->
  <hasProperties name="Obra social"/>
</dslsalud:HealthModel>
```

Fig. 8. Código XML asociado al modelo M1

4 Conclusión

El alto nivel de abstracción de un DSL facilita la adaptabilidad de los sistemas de información sanitaria, en un contexto de profundos cambios tecnológicos. Si además se definen sus bloques de construcción en concordancia con la especificación de algún estándar de interoperabilidad, permite que la información fluya fácilmente por todo el ecosistema de la salud.

En base a estas premisas, se desarrolló un DSL elemental y se implementó una herramienta de edición satisfactoriamente. Cabe destacar que resulta muy beneficioso para el analista/diseñador contar con una herramienta de estas características, ya que aumenta la agilidad en las etapas iniciales de producción de software. Esta ganancia en la agilidad se obtiene gracias a la sencillez de los elementos del lenguaje, que pueden identificarse clara y rápidamente con conceptos del área sanitaria: un paciente, un episodio de atención, etc. Y al vincular estos elementos, la herramienta verifica automáticamente que las relaciones tengan sentido. Por ejemplo: no puede relacionarse un paciente directamente con un médico, sino a través de una consulta o una práctica. De este modo se evitan errores desde una fase temprana.

Para dar continuidad al proyecto, se propone realizar una transformación automática de modelo a texto, para generar código ejecutable a partir del archivo XML asociado al mismo. Asimismo se pretende agregar una vista dinámica para enriquecer el lenguaje.

Referencias

1. Pons, C., Giandini, R., Pérez, G.: Desarrollo de Software Dirigido por Modelos: conceptos teóricos y su aplicación práctica. EDULP&Mc-Graw-Hill, La Plata (2010)
2. Kelly, S., Tolvanen, J.: Domain-Specific Modeling: Enable Full Code Generation. Wiley-IEEE Computer Society, Hobaken (2008)
3. García Molina, J. *et al*: Desarrollo de Software Dirigido por Modelos: Conceptos, Métodos y Herramientas. Ra-Ma, Madrid (2012)
4. Topol, E.: The Creative Destruction of Medicine: How the Digital Revolution Will Create Better Health Care. Basic Books, New York (2012)
5. OMG Unified Modeling Lannguage™ (OMG UML), Infrastructure. OMG (2011), https://www.omg.org/spec/UML/2.4.1/Infrastructure/PDF
6. Foro de la OMS sobre la Estandarización y la Interoperabilidad de los Datos Sanitarios. Organización Mundial de la Salud, Ginebra (2012)
7. Resourcelist – FHIR v4.0.0, https://hl7.org/fhir/resourcelist.html
8. Martínez García, A.: Resolviendo el Diseño de Modelos de Dominio HL7 mediante Soluciones Guiadas por Modelos. Departamento de Lenguajes y Sistemas Informáticos, Universidad de Sevilla (2016)
9. Villegas, A. *et al*: UML Profile for MIF Static Models. Version 1.0 (2013), http://www.vico.org/HL7_Tooling/Submission/MIF.pdf
10. Sirius – The easiest way to get your own Modeling Tool, https://www.eclipse.org/sirius/