Derivación de un Proyecto Web a partir del Modelado Específico de Dominio de un Sistema de Información Sanitaria

Cesaretti Juan[1], Calabrese Arian[1], Rocca Leandro[1] y Nahuel Leopoldo[1]

[1] GIDAS, Grupo de I&D Aplicado a Sistemas informáticos y computacionales
Facultad Regional La Plata - Universidad Tecnológica Nacional
{jcesaretti,leorocca,lnahuel}@frlp.utn.edu.ar
acalabrese@alu.frlp.utn.edu.ar

Abstract. Producir sistemas de información sanitaria que puedan intercambiar datos entre sí, en un contexto de permanentes actualizaciones, requiere un esfuerzo cada vez mayor. Con el propósito de facilitar el proceso de desarrollo de software, y a la vez asegurar su adaptabilidad e interoperabilidad, se propuso la construcción de una herramienta de Modelado Específico del Dominio, usando la plataforma MetaEdit+. De este modo, se logró proporcionar un entorno para construir distintos tipos de diagramas: estructurales estáticos, dinámicos y de interfaz gráfica de usuario. Para expresar estas especificaciones de alto nivel, se definieron Lenguajes Específicos del Dominio, basados en un estándar de interoperabilidad clínica. Y aplicando transformaciones de modelo a texto, se alcanzó la meta de generar automáticamente la capa lógica y la vista de un proyecto Java web tomando como entrada dichos diagramas. La herramienta se probó con el modelado de diferentes casos de estudio, y se obtuvieron los proyectos correspondientes, que fueron abiertos y ejecutados correctamente en el ambiente Apache NetBeans. Como líneas de trabajo futuro, se planea avanzar en la generación de las capas de control y persistencia, para incrementar tanto como sea posible el grado de automatización que provee la herramienta desarrollada.

Keywords: Modelado Específico de Dominio (DSM), Lenguaje Específico de Dominio (DSM), Fast Healthcare Interoperability Resources (FHIR), Java Web.

1 Introducción

El desarrollo de Sistemas de Información Sanitaria (SIS) presenta dos problemáticas desafiantes. Por un lado, la información de las atenciones médicas debe ser accesible desde las distintas instituciones de salud que visita un paciente. En los países más desarrollados, la esperanza de vida va en aumento y, con ella, también crece la movilidad de los pacientes [1]. El otro desafío que presentan los SIS es su complejidad creciente. Estos requieren una adaptación continua a las innovaciones tecnológicas [2].

Para abordar la complejidad de estos sistemas, se propone utilizar el poder de abstracción de la MDE (Ingeniería de Software Dirigida por Modelos), que considera a los modelos como los elementos principales del desarrollo del software [3]. Y se propugna la combinación de este enfoque con el uso de un estándar de interoperabilidad sanitaria [4].

Se han publicado diversos trabajos que tomaron esta iniciativa, con el propósito de extender o especializar un lenguaje de propósito general como UML (Unified Modeling Language), o lograr transformaciones de modelos entre el estándar HL7 (Health Level Seven) y UML [5] [6].

El propósito de este proyecto, a diferencia de las experiencias afines antes mencionadas, es construir una herramienta para generar sistemas de información sanitaria, maximizando la automatización de dicho proceso. Con tal fin, se propone encarar el desarrollo en el contexto de la MDE, pero desde otra perspectiva: la del DSM (Modelado Específico del Dominio). Este enfoque 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 [7]. La automatización se concreta mediante derivaciones o transformaciones de modelo a texto (M2T). Estas transformaciones son programas que recorren todos los elementos de un modelo gráfico (diagrama), toman de ellos los datos necesarios y producen como salida un archivo de texto con la estructura deseada: Java class, JSP, CSS, etc.

2 Desarrollo

2.1 Metamodelador

Para desarrollar este proyecto se utilizó MetaEdit+ [8], un metamodelador que provee herramientas muy potentes que facilitan la tarea del DSM : para definir los grafos (diagramas), especificados con el metalenguaje GOPPRR (Graph-Object Property-Port-Role-Relationship), para definir las propiedades de cada tipo de objeto, para editar los símbolos de cada tipo de objeto, tal como aparecen en la paleta del editor, y para editar y ejecutar transformaciones M2T con un lenguaje de programación propio llamado MERL (MetaEdit Reporting Language).

2.2 Trabajo previo

En la primera fase del proyecto, se desarrolló un DSL para representar los aspectos estructurales de un SIS. Para definir la sintaxis abstracta de este DSL, denominado SIS_Static [9], se construyó un metamodelo.

Con el fin de establecer los bloques de construcción del DSL SIS_Static, se seleccionó un subconjunto de elementos del estándar FHIR [10]. En la especificación de dicho estándar, los elementos son conocidos como “recursos”. El criterio de selección fue que conformaran un conjunto mínimo de recursos, suficiente para modelar diferentes SIS. básicos: gestión de atenciones ambulatorias en consultorios sin turno, atenciones en guardias, etc. Los recursos seleccionados son: Organisation, Practitioner, Patient, EpidoseOfCare y Encounter.

También se establecieron las relaciones válidas entre los elementos, para lo cual se tomó como base la especificación de FHIR.

Una vez definido el DSL SIS_Static, se construyó un editor para crear, visualizar y modificar diagramas expresados en dicho DSL. Para definir la sintaxis concreta del DSL, se utilizó el editor de símbolos que provee MetaEdit+. Y así quedaron especificados los íconos, tal cual aparecen en la paleta del editor correspondiente.

Por último, se implementaron transformaciones M2T, para generar automáticamente el código para crear las clases y sus métodos básicos, en Java y otros lenguajes de programación. Para ejecutar dichas transformaciones, se agregó un botón para cada una de ellas en la barra superior del editor.

En la segunda fase del proyecto, se desarrolló un DSL con la capacidad de capturar funcionalidades más específicas de los SIS. Lo llamamos SIS_Dynamic [11], y con él pueden graficarse diagramas semejantes a las máquinas de estados de UML. Estos describen los estados por los que puede transitar un elemento de tipo “Encounter”. Pero los estados que pueden determinarse están restringidos a una lista de valores establecidos en el estándar FHIR: planned, in-progress, on-hold, discharged, completed, cancelled, discontinued, entered-in-error, unknown. Los estados se conectan a través de transiciones. Y a una transición puede vincularse una o más acciones (objetos del tipo “Action”).

Luego de definir el DSL SIS_Dynamic, se construyó un editor para crear, visualizar y modificar diagramas confeccionados con este lenguaje. En la paleta se agregaron los símbolos de cada elemento y relación (sintaxis concreta del SIS_Dynamic).

Las acciones son los elementos más importantes de estos modelos, pues a partir de ellas pueden declararse nuevos métodos, más específicos, cuya signatura (identificador y parámetros) pueden agregarse automáticamente a las clases derivadas de los diagramas estáticos. Con tal fin, se modificó la transformación M2T antes mencionada, para adicionar este nuevo comportamiento a las clases.

En la tercera fase, se definió un DSL para modelar la GUI (Interfaz Gráfica de Usuario) de los SIS. Este DSL, que denominamos SIS_Interface [12], permite diseñar la vista de los sistemas modelados, usando elementos que representan las ventanas y sus componentes, y relaciones que indican cómo será la navegación entre ventanas, y la composición de las mismas.

También se creó un editor para gestionar diagramas representados con el DSL SIS_Interface, y se implementó una transformación (programada en MERL) para producir automáticamente el código HTML y CSS que conformará la interfaz gráfica de cada sistema modelado.

2.3 Avances

El propósito del presente trabajo es exponer la integración de las distintas capas que se fueron desarrollando, y la implementación de una transformación global para generar automáticamente un proyecto Java web ejecutable en el IDE Apache NetBeans. Los proyectos producidos tienen una arquitectura de cuatro capas: lógica, persistencia, servlets y vista (GUI).

La transformación global se descompone en 4 módulos o subreportes:

El primer módulo genera la estructura de directorios del proyecto Java web, tomando como directorio raíz el ingresado al crear el proyecto en MetaEdit+, en la propiedad "projectFolder". Los directorios necesarios son: nbproject, src y web.

El segundo módulo genera los archivos de configuración indispensables para que el proyecto pueda ser reconocido y abierto correctamente en el IDE NetBeans. Dentro de esos archivos se encuentran: build.xml (que describe el proceso de generación y sus dependencias), ant-deploy.xml, project.xml, project.properties, etc.

El tercer módulo es la transformación M2T que navega por todos los elementos de un diagrama estático y genera las clases en Java, con sus constructores, getters y setters. Además, como cada elemento de tipo "Encounter" está conectado a través de una relación de explosión con un diagrama dinámico, recorre todas las transiciones entre estados. Y por cada elemento de tipo "Action" detallado en alguna de esas transiciones, agrega el encabezado de un método específico a la clase correspondiente, con su identificador y la declaración de sus parámetros, y un breve comentario. Todo esto es generado a partir de los datos ingresados en las propiedades de la acción, al crear el diagrama dinámico. Los archivos de las clases son almacenados en el directorio "logica", que reúne el código que implementa la capa lógica.

El cuarto módulo accede al diagrama de GUI, a través de la propiedad "View" del diagrama estático. Al visitar los distintos elementos, produce como salida el código JSP y las hojas de estilo CSS que son almacenadas en la carpeta "web", donde se implementa la capa correspondiente a la vista o GUI.

3 Estudio de Caso

En esta sección se expone la aplicación a un caso particular de las herramientas desarrolladas. Se trata de la gestión de la información de atenciones kinesiológicas en un consultorio. Como muestra el diagrama de la figura 1 (construido con el editor del SIS_Static), se modela un episodio en el que un deportista recibe tratamiento, a cargo de un traumatólogo, en una clínica deportiva. Y este acude a sesiones de rehabilitación efectuadas en un consultorio, donde es atendido por un kinesiólogo.

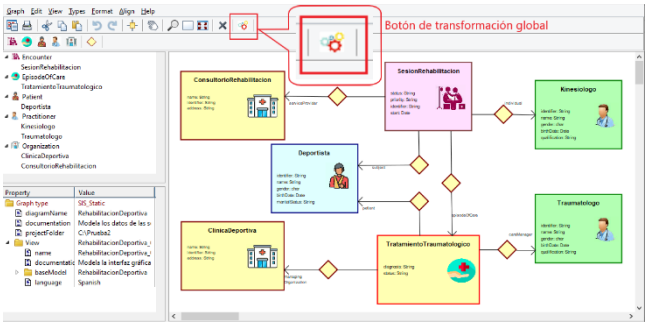

Fig. 1. Diagrama estático

Cada sesión de rehabilitación es una instancia de un Encounter. Y los diferentes estados por los que puede pasar una sesión se modelan en un diagrama dinámico, usando el editor del DSL SIS_Dynamic (ver figura 2). Dicho diagrama está vinculado a la sesión de rehabilitación a través de una relación de explosión.

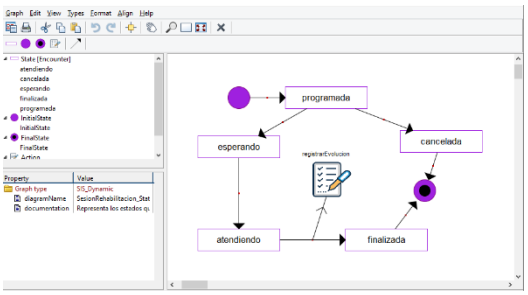

Fig. 2. Diagrama dinámico

El modelado del sistema se completa con el diseño de la interfaz gráfica de usuario. Para eso se utiliza el editor del DSL SIS_Interface. El diagrama puede verse en la figura 3. La ventana principal (Consultorio de Rehabilitación) solamente contiene un menú para navegar hacia las otras ventanas (Kinesiólogos, Pacientes y Sesiones), mediante botones nominados como se muestra en el gráfico. Cada ventana muestra los formularios necesarios para realizar las operaciones especificadas en los componentes que se indican en el diagrama. Por ejemplo, en la ventana de Pacientes, puede registrarse un nuevo paciente, puede buscarse un paciente por distintos criterios, y puede modificarse o eliminarse algún paciente.

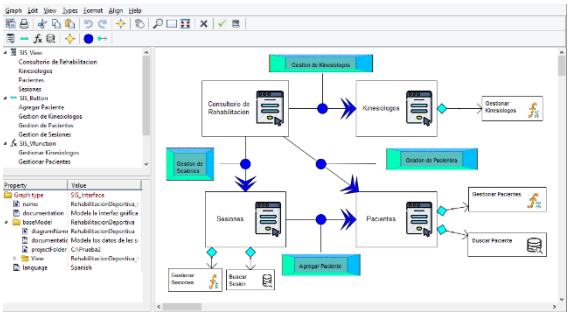

Fig. 3. Diagrama de interfaz gráfica

Luego de completar los tres diagramas, se ejecuta la transformación global, utilizando el botón correspondiente en el editor del DSL SIS_Static (ver figura 1).

Finalmente, se abre en el IDE NetBeans el proyecto Java web generado automáticamente. El entorno no muestra ningún error en los archivos abiertos. Luego, al ejecutarlo (ver figura 4), se cargan exitosamente las páginas web en el navegador escogido, como muestra la figura 5.

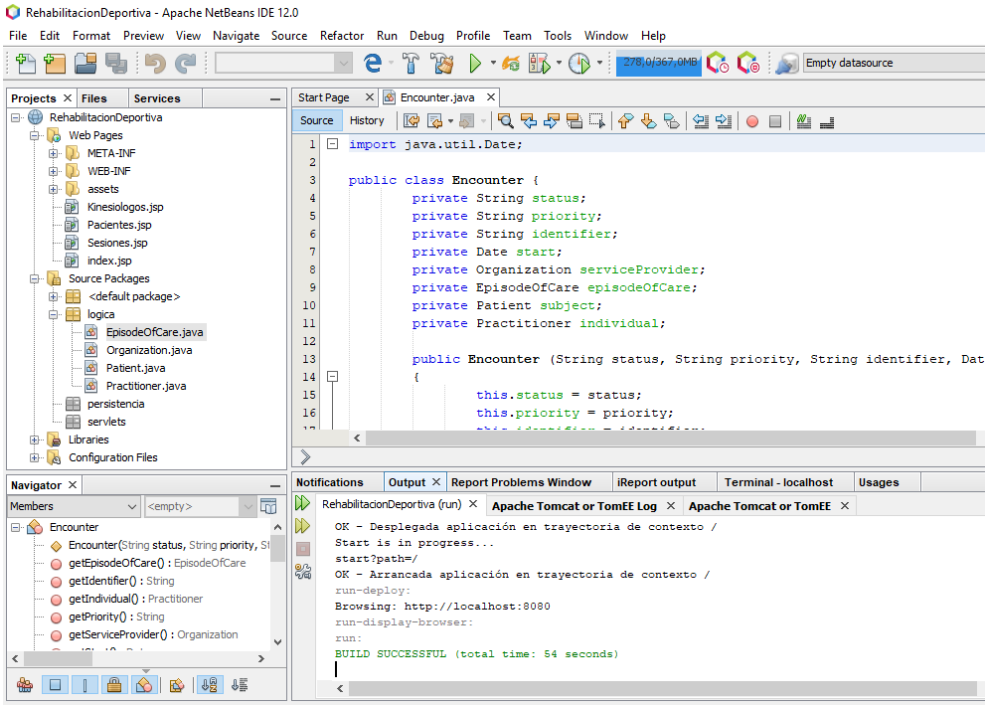

Fig. 4. Proyecto generado, abierto y ejecutado en el IDE NetBeans

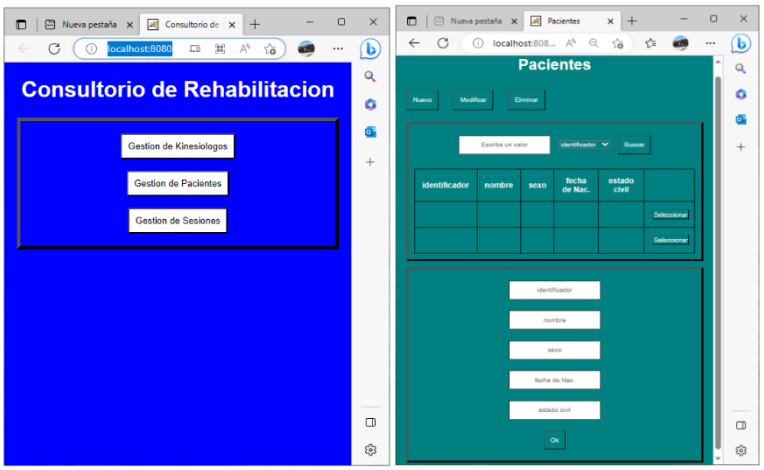

Fig. 5. Páginas web del proyecto en ejecución

4 Conclusión y Trabajo Futuro

En la fase actual del desarrollo del proyecto, se logró la integración de todas las vistas de los sistemas modelados (estática, dinámica y de interfaz gráfica). Y a través de una transformación global M2T, se pudo generar automáticamente un proyecto Java web a partir de los diagramas confeccionados. Los proyectos generados contienen los directorios y los archivos necesarios para que puedan ser reconocidos y abiertos por el IDE Apache NetBeans. Las pruebas realizadas fueron exitosas, y este entorno no detectó errores en las piezas de código producidas a partir de la derivación implementada. Además, los proyectos pueden ejecutarse correctamente.

Este avance es muy significativo para la consecución del propósito final de este proyecto: proveer una herramienta para facilitar y acelerar el proceso de desarrollo de sistemas de información sanitaria, desde las fases iniciales, cumpliendo ciertos requerimientos de calidad: interoperabilidad y adaptabilidad. Además, se busca empoderar a los expertos del dominio y analistas de negocio, que generalmente no están tan habituados al uso de herramientas CASE, como sí sucede con los arquitectos de software, programadores, testers, etc.

Como trabajo futuro, se planea completar las capas de persistencia y servlets, para extender la derivación global M2T, y minimizar la cantidad de código que debe agregarse manualmente.

Asimismo, se proyecta definir métricas y algún indicador adecuado para medir y evaluar el progreso durante el desarrollo de los sistemas modelados, y el grado de automatización alcanzado mediante la herramienta DSM desarrollada.

Referencias

1. Braunstein, M.: Health Care in the Age of Interoperability: The Potential and Challenges. IEEE pulse 9(5), 34–36 (2018).
2. Organización Panamericana de la Salud: De la evolución de los sistemas de información para la salud a la transformación digital del sector de la salud. In: Informe de la conferencia sobre IS4H. Washington D.C, EE. UU. (2021).
3. García Molina, J. et al.: Desarrollo de software dirigido por modelos: Conceptos, métodos y herramientas. 2nd edn. Ra-Ma, Madrid, España (2013).
4. Organización Panamericana de la Salud: Revisión de estándares de interoperabilidad para la eSalud en Latinoamérica y el Caribe. Washington D.C., EE. UU. (2016).
5. Olivero, M. et al.: Facilitating the design of HL7 domain models through a model-driven solution. BMC Medical Informatics and Decision Making 20, 96 (2020).
6. Pfaff, E. et al.: 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 7, 229-241 (2019).
7. Kelly, S., Tolvanen, J.: Domain-Specific Modeling: Enable Full Code Generation. Wiley-IEEE Computer Society, Hoboken (2008).
8. MetaEdit+ Workbench User's Guide, https://www.metacase.com/support/55/manuals/mwb/Mw.html, last accessed 2023/05/10.
9. Cesaretti, J. et al: Uso de DSL y MetaEdit+ para automatizar etapas iniciales en la construcción de Sistemas de Información Sanitaria. En: Anales de las 49 JAIIO - ASSE, pp. 210-224 (2020).
10. Benson, T., Grieve, G.: Principles of health interoperability: SNOMED CT, HL7 and FHIR. Springer-Verlag, Londres, Inglaterra (2016).
11. Cesaretti, J. et al: Tecnología CASE para modelado específico de dominio en sistemas de información sanitaria basado en estándar de interoperabilidad clínica. En: XXVII Congreso Argentino de Ciencias de la Computación (2021).
12. Cesaretti, J. et al: Construcción de una Herramienta de Modelado Específico para Generar Sistemas Interoperables de Información Sanitaria. En: Proceedings of the IADIS Ibero American Conference Applied Computing, pp. 179-182 (2021).