Generación automática de interfaces de usuario para sistemas de información clínicos basados en una metodología multi-nivel

Arianne Palau[a], Laura Cuadrado[a], Pablo Pazos[b]

[a] *Facultad de Ingeniería, Universidad de la República, Uruguay*
[b] *openEHR en español, Asociación Chilena de Informática en Salud, CaboLabs*

Resumen y Objetivos

En general los procesos de desarrollo de software de uso clínico son complejos, costosos, informales y de calidad variable. Esto se debe en gran medida a la falta de buenas prácticas metodológicas, al no uso de estándares y herramientas existentes.
Este trabajo presenta una metodología de Modelado Multi-Nivel (MMN), orientada por estándares y centrada en la automatización de tareas, en particular de la generación automática de Interfaces de Usuario (IU) para el Registro Clínico. Cada modelo del MMN representa una capa arquitectónica del software (bases de datos, lógica de negocio, y presentación), utilizando los modelos del estándar openEHR como base y creando un nuevo modelo para especificar IU.
La automatización de tareas basada en estos modelos permite reducir los tiempos de desarrollo y validación, lograr sistemas más homogéneos, mantenibles a largo plazo y bajo costo.
El MMN plantea una solución formal, viable y sostenible a largo plazo, para construir sistemas de información clínicos, donde el experto del dominio clínico juega un papel preponderante en el proceso de desarrollo.

Palabras Clave:

Interfaz de usuario, modelado multi-nivel, usabilidad, historia clínica electrónica, openEHR, arquetipos, eSalud.

Introducción

Los sistemas informáticos de uso clínico son altamente complejos debido a la heterogeneidad y complejidad inherentes al dominio de la salud en cuanto a procesos, registros y objetivos de los distintos roles que utilizan estos sistemas. Además se requieren modificaciones con alta frecuencia, y es necesario adaptar rápidamente los sistemas a nuevos requisitos, conservando cierto nivel de calidad.

Las características de calidad del software como la mantenibilidad, flexibilidad, escalabilidad y usabilidad, dependen en gran medida de la arquitectura, modelos de información e intercambio, buenas prácticas y estándares, también dependerán de las metodologías de análisis, especificación, desarrollo y aseguramiento de la calidad que sean utilizados.

Hoy en día es difícil encontrar metodologías, procesos de desarrollo de software y herramientas que se adapten debidamente a la realidad de la salud. Los procesos son burocráticos, lentos y costosos, los sistemas se adaptan lentamente a los nuevos requerimientos y se van atrasando con respecto a las necesidades actuales.

Este trabajo propone un cambio en la metodología de desarrollo de software clínico, haciendo foco en el uso de estándares y tomando a la mantenibilidad como un requerimiento fundamental, donde la participación de expertos del dominio clínico (no informáticos) debe considerarse como básico durante todo el proceso. Estos expertos serán quienes definan los requerimientos a ser implementados por los informáticos. Para lograrlo es necesario contar con métodos, modelos y herramientas que utilizados en conjunto permitan definir los requerimientos del sistema, en particular la estructura de los registros clínicos. Este es un gran contraste con respecto a los procesos de desarrollo actuales, en donde el equipo informático interroga al equipo clínico para extraer requerimientos y luego el equipo informático trabaja de forma aislada en la interpretación, especificación e implementación de estos requerimientos, obteniendo resultados de calidad variable o no cumpliendo con los requerimientos de los profesionales clínicos.

La metodología de Modelado Multi-Nivel (MMN) propuesta en este trabajo es una combinación de estándares, buenas prácticas, herramientas y tecnologías de forma armónica. En particular MMN aporta un modelo genérico de Interfaces de Usuario (IU), llamado UITemplate (fig. 4), que utiliza como base las especificaciones del estándar openEHR [1], el cual define dos modelos: Modelo de Información y Modelo de Contenido. Este esquema podría extenderse con más niveles, por ejemplo un nivel que especifique reglas complejas para Soporte a las Decisiones Clínicas [2] o un nivel para especificar Flujos de Trabajo de los usuarios clínicos, que se adapten a los procesos asistenciales.

Se especificó un formato XML [3] para representar definiciones de UITemplate (fig. 8) que permite definir, procesar y compartir definiciones de IU entre distintos sistemas. De esta forma se busca emular la forma en que se publican y comparten los arquetipos en la comunidad de openEHR [4]. Se defi-

nió un esquema XML [5] que formaliza y permite validar instancias de XML para UITemplate.

Por último, el MMN formaliza cada etapa del proceso de definición y generación automática de Interfaces de Usuario, y ofrece lineamientos generales de integración de las IU generadas con software de registro clínico.

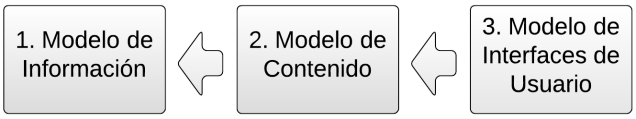

Figura 1: Diagrama MMN

Motivación

El estándar openEHR es considerado una buena solución para resolver algunos de los problemas mencionados, como la mantenibilidad y estandarización de la información clínica. Sabiendo que este estándar no incluye un componente para definir, compartir y generar Interfaces de Usuario, es interesante poder proponer una solución formal, flexible y genérica, haciendo foco en la mantenibilidad a largo plazo y bajo costo. Por lo tanto, este trabajo podría ser la primera aproximación para la definición de un estándar de Interfaces de Usuario para sistemas clínicos openEHR.

Por otro lado, el problema de la generación automática de IU es en sí interesante, ya que permitirá acortar los tiempos de prototipado y validación, desarrollo y adaptación de los sistema de uso clínico a nuevos requerimientos. Esto aporta otro elemento para mejorar la mantenibilidad del software.

La metodología MMN, especificaciones, modelos y herramientas creadas durante el proyecto serán publicadas para ser revisados y mejorados por la comunidad.

Estado del arte

Como parte del trabajo se realizó un relevamiento de las herramientas existentes para la especificación y generación automática de IU. Se exigieron tres requerimientos básicos: 1. específica para el ámbito de la salud (openEHR compatible), 2. independiente de la tecnología, 3. de código abierto y uso libre. Al momento de realizar este relevamiento, no existían metodologías y herramientas capaces de cumplir con las características buscadas, lo que motivó este trabajo para diseñar un enfoque genérico de definición y generación de IU para sistemas de información en salud.

Objetivos

El principal objetivo de este trabajo es aportar elementos para mejorar y simplificar las metodologías de desarrollo de software en salud, con un enfoque orientado por estándares, buenas prácticas, considerando requerimientos de calidad desde el inicio, y de forma independiente de tecnologías particulares.

Como fue mencionado, se hizo foco en la generación automática de IU, con esto se buscan las siguientes características:

1. *Consistencia.* Las IU se pueden generar en masa, de forma consistente entre si, permitiendo aplicar estándares de usabilidad uniformes para el usuario.
2. *Flexibilidad*: Permitir la definición de IU con distintas estructuras. Permitir adaptar rápidamente las IU a cambios y nuevos requerimientos.
3. *Portabilidad*. El generador debe ser configurable, de manera de generar IU para distintas tecnologías.

En el plano práctico, el objetivo del proyecto es mejorar algunas características del componente de generación de IU de EHRGen [6]: una herramienta de generación de sistemas de registro clínico que implementa el estándar openEHR. EHRGen permite crear y modificar registros clínicos mediante arquetipos openEHR sin necesidad de modificar la estructura de la base de datos o el código fuente de la aplicación. En particular, las características que se desean mejorar son: generalidad, flexibilidad, expresividad y completitud. Pero el MMN está definido de forma genérica, lo que permite implementar la metodología independientemente a EHRGen.

Marco de trabajo: el estándar openEHR

El estándar openEHR define dos modelos que sirven de base para los modelos necesarios en la metodología MMN.

Modelo de Información [7]: define estructuras de datos genéricas capaces de representar cualquier tipo de registro o documentación clínica y demográfica, sin incluir semántica de los conceptos clínicos específicos que son parte de la estructura de cada documento (contenido). Representa la parte estable y más general de los registros clínicos y es implementado dentro del software a nivel de lógica de negocios y bases de datos.

Modelo de Contenido [8]: representa el contenido específico de los registros clínicos mediante restricciones sobre el Modelo de Información. Esta es la parte cambiante del registro clínico, donde impactan la mayoría de las modificaciones solicitadas por los usuarios clínicos. Define las estructuras de datos, restricciones y terminología, que pueden verse como definiciones de conceptos clínicos específicos (ej. Medida de la Presión Arterial, Prescripción de Medicamentos, Aplicación de Vacunas). Este modelo es gestionado por fuera del software, lo que baja el acoplamiento entre la implementación de los registros clínicos y los cambios que los afectan, elemento de gran valor para la mantenibilidad del sistema.

El Modelo de Contenido de openEHR está formado por el Modelo de Arquetipos y el Modelo de Plantillas. Los arquetipos representan unidades de contenido o conceptos clínicos individuales de forma amplia y genérica, mientras que las plantillas representan agregaciones de arquetipos que representan documentos clínicos específicos para cada ámbito de registro clínico. El estándar openEHR especifica la sintaxis ADL (Archetype Definition Language) para definir, procesar

y compartir arquetipos. ADL [9] es parte del estándar internacional ISO 13606-2 [10]. En cambio se utiliza XML como sintaxis para las plantillas openEHR. En la fig. 2 se aprecia la relación entre estos modelos, haciendo la salvedad de que las plantillas definen la estructura de los registros clínicos, y no cómo estos serán vistos en una pantalla por un usuario clínico, de esto se encargará el modelo de IU (fig. 4).

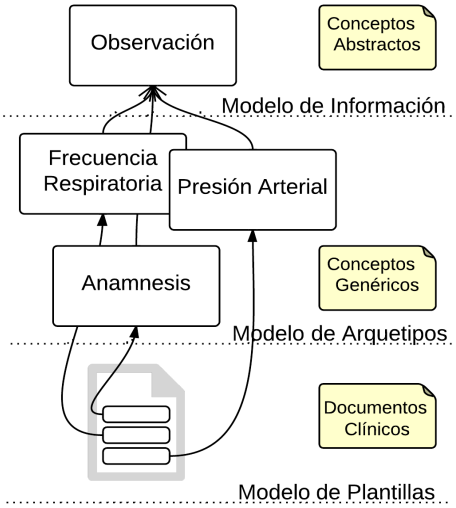

Figura 2: Relación entre modelos de openEHR

La metodología de modelado dual (en dos niveles) de openEHR plantea que los informáticos deben ocuparse de implementar software con el modelo de información, que sea capaz procesar arquetipos y plantillas, mientras que los expertos del dominio clínico deben crear estos elementos para definir la estructura de los registros clínicos. Lo interesante de este enfoque es que: a) los informáticos y clínicos se encargan de tareas para las que están capacitados, al contrario del enfoque habitual, donde el análisis e interpretación de los requerimientos clínicos es realizada solo por informáticos; b) el software es más genérico, flexible y modificable, porque el contenido específico (estructuras de registro clínico) están definidas por fuera del software, como consecuencia, el conocimiento clínico no está implementado de forma "dura" en el software, lo que sí pasa con el enfoque habitual (en un nivel). Estas características son aprovechadas por el MMN.
Debido a que diferentes Plantillas utilizan los mismos Arquetipos, todos los registros realizados en diversos contextos, organizaciones y sistemas, contendrán datos que son semánticamente compatibles, lo que permite el intercambio, la integración, la comparación y análisis de datos distribuidos en diversos sistemas de información.

El software que implementa openEHR puede adoptar de forma sencilla nuevos Arquetipos y Plantillas, o incluso los cambios hechos a Arquetipos y Plantillas existentes. Estos cambios son versionados, lo que permite gestionar la evolución del registro clínico de forma controlada, manteniendo un sistema estable durante la etapa de evolución/mantenimiento, que en general es la más larga y costosa en el ciclo de vida del software, entre 67% y 75% del costo total de un proyecto [11].

Metodología

La Metodología Multi-Nivel propuesta en este trabajo, agrega un nivel al modelo dual de openEHR, el que define un Modelo de Interfaces de Usuario donde se especifican las reglas de presentación de datos clínicos para ingreso, visualización y modificación. Mediante este modelo se generan IU de manera automática. Entonces, MMN define un modelo para cada nivel y una representación de este modelo:

Nivel	Modelo	Representación
1	Información	Software
2	Contenido (Arquetipo / Plantilla)	ADL / XML
3	Interfaz de Usuario (UITemplate)	XML

Tabla 1: niveles, modelos y representaciones del MMN

Cada nivel describe una capa arquitectónica del software (modelo de datos, lógica de negocio y la presentación), por lo que teniendo Arquetipos, Plantillas y UITemplates, gran parte del software clínico podría generarse de forma automática, lo que aceleraría enormemente el proceso de desarrollo de software en salud facilitando el mantenimiento y evolución de este. Esta evolución se haría de forma controlada y trazable a lo largo del tiempo, ya que los cambios en el Contenido e Interfaz de Usuario son versionados.

MMN se implementará según los siguientes pasos:

1. Buscar o crear arquetipos que describan estructuras de registro clínico, utilizando las herramientas CKM [4] y Archetype Editor [12].
2. Especificar las interfaces de usuario para las estructuras definidas en 1., creando un UITemplate mediante un editor de IU (fuera del alcance de este trabajo).
3. Para las tecnologías en las que se desea generar las IU, crear el archivo Map o usar uno existente. Este paso se realizaría una sola vez por cada tecnología.
4. Ejecutar el generador con el UITemplate, el archivo Map y otros parámetros de entrada como el idioma de la IU.
5. Integrar las IU generadas en un sistema de registro clínico, vinculando las pantallas con la lógica de negocios de la aplicación.

A continuación se describe el trabajo realizado, tanto a nivel metodológico como tecnológico, para implementar MMN y probar su viabilidad como metodología.

I. Modelo de plantillas de interfaz de usuario

Como punto inicial de la definición del tercer nivel de MMN se diseñó un modelo de IU, que es la evolución del modelo implementado en EHRGen [6]. El nuevo modelo incluye mejoras basadas en experiencias previas, logrando un mayor nivel de flexibilidad, expresividad y formalidad. En la fig. 4 se presenta el diagrama UML de este modelo.

Descripción del modelo:

UITemplate: plantilla sobre la cual se definen las diferentes pantallas o vistas, siguiendo distintas estructuras de disposición y distribución de zonas y campos (Layout).

View: definición de una pantalla o vista que contiene referencias a partes de distintos Arquetipos, las cuales pueden ser a estructuras complejas (ArchetypeContainerReference), o a elementos de datos simples (ArchetypeFieldReference).

Layout y Zone: representan las formas de estructurar las vistas. Las zonas se pueden subdividir vertical y horizontalmente sin límites, lo que brinda una gran flexibilidad de organización de controles para ingreso y visualización de datos (fig. 3).

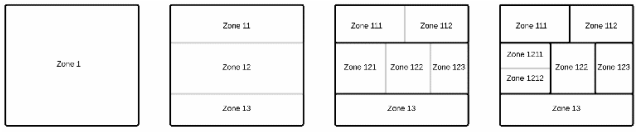

Figura 3: Opciones de disposición (Layout) definidas por zonas

Control y FieldLabel: respectivamente, se trata de los controles que se van a generar en la interfaz de usuario para el ingreso o visualización de un determinado dato que se corresponde con un campo de un Arquetipo, y las etiquetas que acompañan a cada campo, que son referencias a terminología definida en el mismo Arquetipo.

Dado que los arquetipos están traducidos a distintos idiomas, las interfaces de usuario definidas por un UITemplate, podrán generarse para diferentes idiomas, es decir que las etiquetas de los campos y los valores de los listados se adaptan automáticamente a cualquier país o región.

El modelo incluye valores enumerados que representan las clases y tipos de datos del modelo de información de openEHR. Los nombres de tipos de datos se utilizan para saber qué controles mostrar para ingreso y visualización de datos en las IU generadas.

II. Análisis de IU en tecnologías específicas

El objetivo principal del generador es transformar instancias de UITemplate (entrada) y generar IU en forma de definiciones declarativas para diversas tecnologías objetivo (salida). Luego, cada definición declarativa tendrá un intérprete o procesador en una tecnología específica, lo que facilita la integración de las IU generadas en aplicaciones implementadas que utilicen dichas tecnologías.
Esto implica que cualquier tecnología que soporte definiciones declarativas de IU podrá ser integrada como salida del generador de IU construido como parte en este trabajo.

Se realizó un relevamiento de distintas tecnologías que implementan IU declarativas, y, considerando su popularidad y cantidad/calidad de la documentación disponible, se encontraron las siguientes candidatas para integrar al generador de IU del tercer nivel del MMN:

Def. declarativa	Tipo	Intérprete	Tecnología
XHTML [13]	Web	Web Browser	Chrome, Firefox, Safari, IE
HTML5 [14]	Web	Web Browser	Chrome, Firefox, Safari, IE
XAML [15]	Web y Desktop	WPF	Microsoft
SwiXML [16]	Desktop	SwiXML	Java
Mozilla XUL [17]	Web y Desktop	Gecko / XULRunner	Multiplataforma

Tabla 2: Tecnologías de salida para el generador de IU

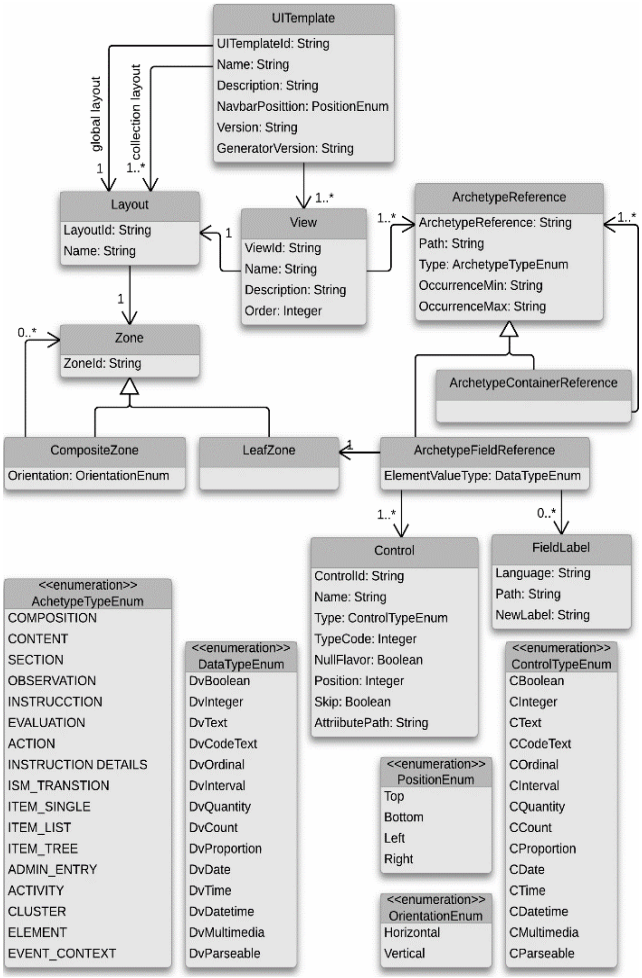

Figura 4: modelo de UITemplate

Tipos de campos y controles de IU

La metodología MMN incluye la definición de correspondencias entre cada tipo de dato del modelo de información de openEHR [18] y cada control de IU (tabla 3), estandarizando los nombres que serán utilizados en UITemplate. Estas definiciones serán utilizadas como restricciones para generar correspondencias con campos de IU en las tecnologías de salida.

En el modelo UITemplate (fig. 4) cada campo (ArchetypeFieldReference) puede tener más de un control asociado, por ejemplo para registrar la altura del paciente en metros o centímetros se utilizaría el tipo DV_QUANTITY [18] que tiene los atributos para la magnitud y las unidades.

Tipo de dato OpenEHR	Atributo	Descripción
DV_BOOLEAN	Value	RadioButton {Y, N, nullflavor(op.)}
	Value	ComboBox {Y, N, nullflavor(op.)}
	Value	CheckBox {Y, N}
DV_IDENTIFIER	*issuer*	TextBlock, no editable. Depende de type.
	assigner	TextBlock, no editable. Depende de type.
	Id	TextBlock, editable. No editable si es autogenerado.
	Type	ComboBox
	issuer	TextBlock, no editable. Depende de type
	assigner	TextBlock, no editable. Depende de type
	id	TextBlock, editable. No editable si es autogenerado.
	type	RadioButton. Si #elementos <=5
DV_TEXT	*value*	TextBlock
	value	TextArea
	Value	TextEditor
DV_CODED_TEXT	*defining_code*	ComboBox en caso de que estén definidos los valores en el arquetipo. No permite la selección múltiple.
	defining_code	RadioButton en caso de que estén definidos los valores en el arquetipo y #elementos <=5. Sin selección múltiple
	defining_code	CheckBox en caso de que estén definidos los valores en el arquetipo y #elementos <=5. Con selección múltiple.
DV_ORDINAL	*symbol*	ComboBox en caso de que estén definidos los valores en el arquetipo. Con selección múltiple.

Tabla 3: Extracto de correspondencias entre tipos de datos de openEHR y controles de IU de UITemplate

III. Generador de Interfaces de Usuario

El generador se diseñó de forma genérica, por lo que las reglas de correspondencia entre los formatos de entrada (UITemplate) y definiciones de IU de salida, puedan especificarse mediante archivos de configuración del generador. Las IU pueden generarse para diferentes idiomas, facilitando la implementación de sistemas clínicos multi-lenguaje.

El generador de **IU** es un tipo de generador pasivo basado en un modelo por nivel [19]. Por lo tanto cumple con las siguientes características:

1. Una vez que genera la salida, se deslinda totalmente del producto final obtenido.
2. Genera código en un nivel, es decir, toma como insumo archivos de configuración y, en combinación con las plantillas definidas, retorna el código final.

Para el prototipo del generador se decidió acotar el alcance según algunas restricciones que no afectan la generalidad de la solución:

1. Se utilizaron arquetipos planos: no hacen referencia a otros arquetipos a través de slots, no tienen referencias internas entre nodos del mismo arquetipo.
2. Se soportan solo los tipos de datos más frecuentes dentro del registro clínico: DV_TEXT (texto narrativo), DV_CODED_TEXT (texto codificado), DV_COUNT (cantidades), DV_QUANTITY (cantidades físicas), DV_DATE (fecha y hora).

Interfaces de Usuario y Registros Clínicos

Existe un vínculo implícito entre los UITemplates y los registros clínicos que serán especificados por los expertos del dominio clínico. Estas con las correspondencias:

1. Cada UITemplate equivale a un documento clínico conformado por varios formularios, pantallas o vistas (Views), tanto para el ingreso como para la visualización de datos.
2. Cada formulario podrá tener su propio diseño o definición de estructura interna (Layout). Estos diseños estarán definidos dentro del UITemplate.
3. Cada referencia a un campo simple o complejo de un arquetipo puede tener características propias de cómo se visualizará en la IU generada, como el tipo de control, posición dentro del Layout, valores por defecto y texto descriptivo (FieldLabel).

Se generarán tres definiciones de IU por cada vista dentro del UITemplate de entrada, cada IU permite implementar una de las transacciones básicas de los sistemas de registro clínico:

1. *Crear registros*: ingreso de datos por parte de un usuario, validación y almacenamiento de datos.
2. *Visualización de registros*: consultas a la base de datos, transformación para visualización, presentación al usuario.
3. *Edición de registros*: consultas a la base de datos, transformación de datos para visualización, presentación al usuario, modificación de datos por el usuario, validación y almacenamiento.

La arquitectura del generador incluye cuatro grandes componentes con responsabilidades bien definidas (fig. 5):

Generator: aplicación de consola que recibe parámetros de configuración, un UITemplate de entrada y el nombre de la tecnología de salida. Este componente es responsable manejar todo el flujo de generación, desde la carga del UITemplate, los archivos de configuración y arquetipos, hasta la ejecución de las reglas de mapeo y generación de la salida (instancias de definiciones declarativas de IU en la tecnología de salida especificada).

Loader: se encarga de realizar todas las lecturas de archivos. Actualmente soporta únicamente la lectura desde el sistema de archivos, pero se creó de forma que se pueda extender, implementando una interfaz, para poder cargar archivos de otras fuentes, incluyendo fuentes remotas (TCP, FTP, HTTP, SOAP, etc.).

Parser. carga, procesa y valida instancias de UITemplate en formato XML, antes de que la misma sea procesada por el Mapper. La validación incluye controles tanto a nivel de estructura como de restricciones de dominio. La validación de la instancia de UITemplate se realiza contra un esquema XML (XSD) definido [5]. Además implementa funciones útiles para facilitar el trabajo con arquetipos.

ArchetypeManager: este componente se encarga de buscar y cargar arquetipos en la aplicación desde un repositorio dado. Esta clase incluye un caché de arquetipos que permite que los demás componentes accedan a los arquetipos sin necesidad de cargarlos cada vez que se utilicen.

Mapper: ejecuta las reglas de correspondencia, definidas en archivos Map, entre elementos definidos en el UITemplate de entrada y la definición de interfaz de usuario en la tecnología seleccionada. Todas estas correspondencias se resuelven de forma automática basándose en los archivos de configuración.

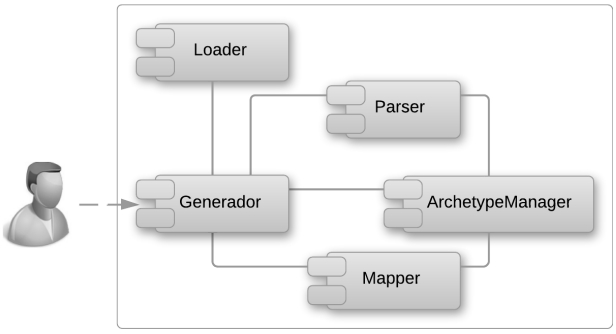

Figura 5: Arquitectura del generador de IU

IV. Validación de MMN mediante prototipo

Para validar la metodología, el modelo de UITemplate y el generador, se desarrolló un prototipo y varios casos de prueba, obteniendo los resultados esperados, y detectando puntos de mejora y perfeccionamiento que se anotan como trabajo futuro. A continuación se expone uno de los casos de prueba.

Archivos de correspondencia (Map)

Para ejecutar el caso de prueba, es necesario crear un archivo "Map" [20] que define las reglas de correspondencia entre el modelo de UITemplate y las definiciones declarativas de IU en la tecnología de salida (HTML5, XAML, SwiXML, etc.). Cada archivo Map está conformado por un conjunto de variables "de sustitución", donde se configura el código a generar en la tecnología específica. Cada variable de sustitución permite establecer una estructura configurable, que utiliza variables "globales". A través de estas variables se referencian a valores definidos en el UITemplate de entrada y en los arquetipos openEHR referenciados. Todas las variables mencionadas fueron especificadas formalmente como parte del tercer nivel de la MMN [21].

En el Map de prueba se creó un diseño general para contener todos los formularios generados para cada View del UITemplate. Esta estructura está definida por variables de sustitución que representan (figs. 6 y 7):

- Menú principal para acceder a las distintas vistas definidas en el UITemplate (NavBar).
- Sección donde se muestra el contenido de la vista seleccionada. (ViewContainer)
- Cada vista generada contendrá:
  - Un título en la parte superior, una tabla que se corresponderá con la disposición definida en el diseño (Layout) asociado a la vista, campos con sus etiquetas para el ingreso y visualización de datos.

Para el prototipo se utilizó un UITemplate que utiliza un arquetipo openEHR plano (fig. 8), y se crearon tres archivos Map para generar IU en las tecnologías HTML5, XAML Web y XAML WinForm [15]. Las figs. 9, 10, y 11 muestran las interfaces generadas tal como las vería un usuario final. Como se puede apreciar hay una gran consistencia estructural entre las distintas IU generadas. Todos los textos de las etiquetas y controles para seleccionar valores de una lista, son extraídos de los propios arquetipos. Si estos contienen traducciones, las IU se podrán generar para distintos idiomas.

Tecnologías utilizadas

Para implementar el prototipo se eligió el lenguaje de programación Groovy [23], un lenguaje dinámico sobre Java. Además EHRGen [6] está desarrollado en Groovy, lo que permitió reutilizar código. Para trabajar con XML se utilizó la API de Groovy XMLSlurper [24], que además permite utilizar expresiones XPath (XML Path Language) [25] para extracción de datos desde archivos XML.

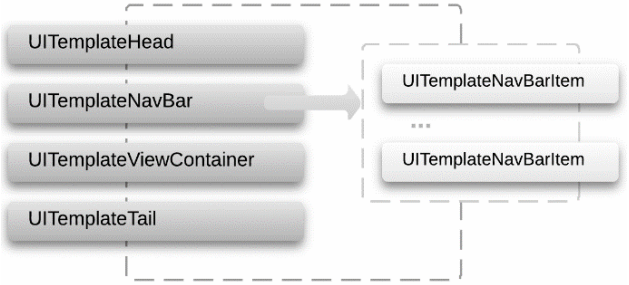

Figura 6: Estructura general de las UIs a generar en función de variables de sustitución

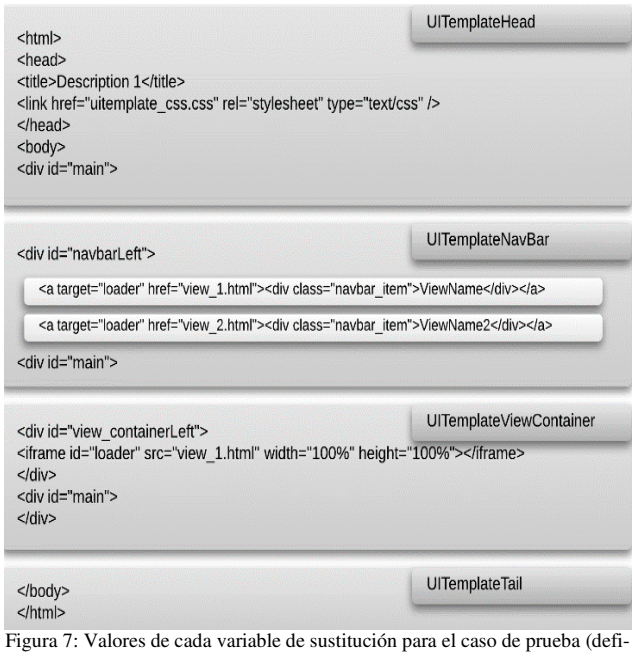

Figura 7: Valores de cada variable de sustitución para el caso de prueba (definidas en el archivo Map)

```
<UITemplate UITemplateId="Template1" Name="Template1" Version="1"
  Description="UITemp Description 1" NavbarPosition="left" LayoutId="l1" GeneratorVersion="1.0">
  <Layout LayoutId="l1" Name="Layout1">
    <CompositeZone ZoneId="z1" Orientation="horizontal">
      <LeafZone ZoneId="z11"/>
      <LeafZone ZoneId="z12"/>
      <LeafZone ZoneId="z13"/>
    </CompositeZone>
  </Layout>
  <View ViewId="v1" Name="ViewName" Description="Registro de Datos Paciente" Order="1">
    <ArchetypeContainerReference Path="/" Type="COMPOSITION" OccurrenceMin="1"
      OccurrenceMax="1" ArchetypeReference="openEHR-EHR-COMPOSITION.arquetipo_prueba_001.v1">
      <ArchetypeContainerReference Path="/context" Type="EVENT_CONTEXT" OccurrenceMin="1"
        OccurrenceMax="1" ArchetypeReference="openEHR-EHR-COMPOSITION.arquetipo_prueba_001.v1">
        <ArchetypeContainerReference  Type="ITEM_TREE" OccurrenceMin="1" OccurrenceMax="1"
          ArchetypeReference="openEHR-EHR-COMPOSITION.arquetipo_prueba_001.v1"
          Path="/context/other_context[at0001]">
          <ArchetypeContainerReference  Type="CLUSTER" OccurrenceMin="1" OccurrenceMax="1"
            ArchetypeReference="openEHR-EHR-COMPOSITION.arquetipo_prueba_001.v1"
            Path="/context/other_context[at0001]/items[at0014]">
            <!-- Definicion de campos -->
            <!-- Zona 11 -->
            <!-- Tipo Identificador  -->
            <ArchetypeFieldReference Type="ELEMENT" OccurrenceMin="1" OccurrenceMax="1"
              ZoneId="z11" ArchetypeReference="openEHR-EHR-COMPOSITION.arquetipo_prueba_001.v1"
              Path="/context/other_context[at0001]/items[at0014]/items[at0004]">
              <Control ControlId="tmp0000001" Name="CTipoIdentificador_1"
                Type="ccodedtext" TypeCode="1" NullFlavor="0" Position="1" Skip="0"
                AttributePath="/context/other_context[at0001]/items[at0014]/items[at0004]/value"/>
            </ArchetypeFieldReference>
            <!-- Nro Identificador -->
            <ArchetypeFieldReference Type="ELEMENT" OccurrenceMin="1" OccurrenceMax="1"
              ZoneId="z11" ArchetypeReference="openEHR-EHR-COMPOSITION.arquetipo_prueba_001.v1"
              Path="/context/other_context[at0001]/items[at0014]/items[at0003]">
              <Control ControlId="tmp0000002" Name="CIdentificador_1" Type="ccount"
                TypeCode="1" NullFlavor="0" Position="2" Skip="0"
                AttributePath="/context/other_context[at0001]/items[at0014]/items[at0003]/value"/>
            </ArchetypeFieldReference>
```

Figura 8: Extracto de XML de UITemplate [22]

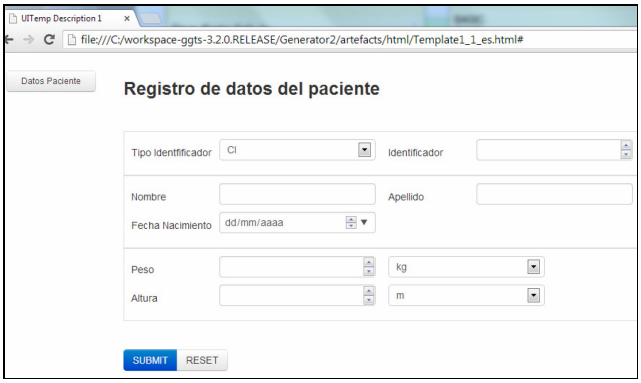

Figura 9: IU HTML5 (Web) generada

Figura 10: UI XAML Web generada

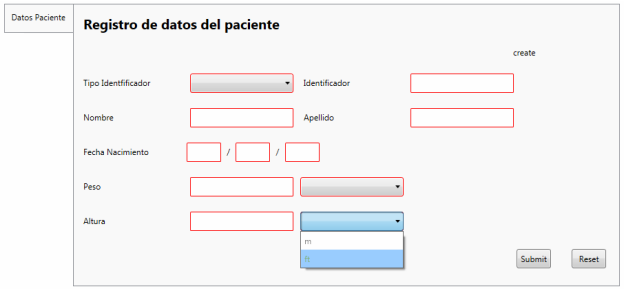

Figura 11: UI XAML WinForm generada

V. Integración de IU con sistemas de información

Una vez generadas las definiciones declarativas de IU, un programador puede integrarlas al software de registro clínico tal como si estas hubieran sido creadas por otro programador, sin conocimientos previos de openEHR o del MMN. Es posible describir componentes genéricos de software (fig. 12) que facilitarían la integración, para implementar una de las tres transacciones básicas de ingreso de datos, visualización de datos, y modificación de datos.

Se desarrolló una pequeña aplicación que implementa las transacciones antes mencionadas, como prueba de que esta integración era posible.

Conclusiones

La metodología de Modelado Multi-Nivel formaliza los modelos que describen distintas capas de la arquitectura de software como persistencia de datos, lógica de negocios y presentación de información. En este trabajo se definió un modelo, herramientas y metodología para poder definir, procesar y compartir definiciones de Interfaces de Usuario para sistemas clínicos, con foco en la generación automática de IU. También se describió como los modelos que conforman en MMN permiten formalizar y automatizar distintas tareas en los procesos y metodologías de desarrollo de software clínico.

Se pude decir que uno de los principales resultados fue demostrar mediante pruebas prácticas que el enfoque MMN es viable, que tiene un gran potencial para disminuir los tiempos de desarrollo y mantenimiento de software, y que aportan en gran

medida a la mantenibilidad, consistencia y estandarización del software clínico, gracias a las características heredadas del modelo dual de openEHR y el nuevo modelo de UITemplate.

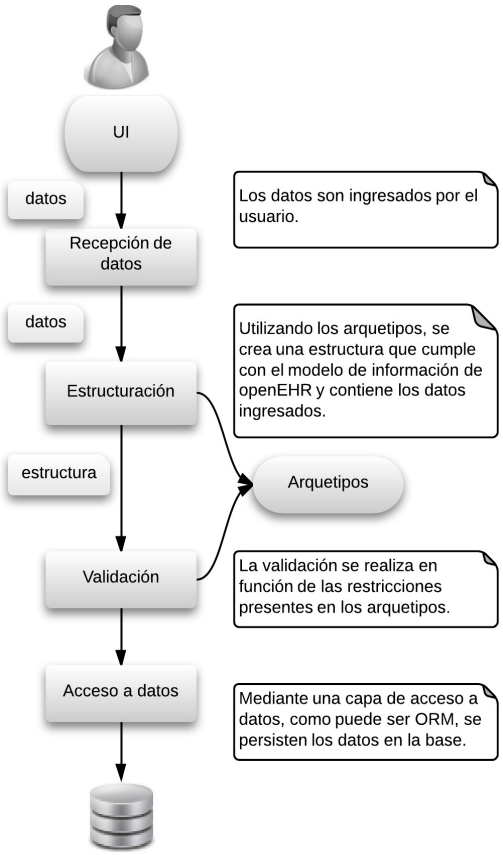

Figura 12: componentes de software involucrados en la transacción de ingreso de datos actualmente implementados en EHRGen [6]

Trabajo futuro

Para lograr una amplia adopción de la metodología de Modelado Multi-Nivel, es necesario contar con herramientas que faciliten su implementación, como lo son el Editor de Definiciones de Interfaz de Usuario (UITemplate), el Editor de archivos Map para generar IU en diversas tecnologías de salida, y alguna herramienta que permita publicar, compartir y revisar definiciones de UITemplate y archivos Map. Si bien como parte de este trabajo se desarrolló un prototipo de un Editor de UITemplate llamado Diseñador de IU, es necesario extender esta implementación a una herramienta visual, para facilitar su uso por parte de expertos del dominio clínico.

Por otro lado, existen múltiples mejoras para implementar dentro del Generador de IU, como agregar soporte para los demás tipos de datos openEHR, arquetipos no planos, plantillas openEHR, y crear una interfaz gráfica para el generador.

El nuevo modelo de UITemplate y el generador específico para HTML5 se implementarán dentro de EHRGen, sustituyendo el componente actual de generación de IU de EHRGen. Esto dará como resultado una primera implementación de la metodología MMN en un software específico para registro de información clínica.

Por último, la metodología MMN será publicada para su revisión y mejora en la comunidad de openEHR y será propuesto como estándar de especificaciones de IU para sistemas que implementan openEHR.

Referencias

[1] Beale, T., Heard, S., Dual Model, openEHR Architecture Overview, 2008, p.15.
http://openehr.org/releases/1.0.2/architecture/overview.pdf

[2] Chen, R., Corbal, I., Guideline Definition Language, Cambio Healthcare Systems, 2013.
https://github.com/openEHR/gdl-tools/wiki

[3] Extensible Markup Language (XML), W3C, 2014.
http://www.w3.org/XML/

[4] openEHR Clinical Knowledge Manager
http://ckm.openehr.org/ckm/

[5] UITemplate XML Schema Definition
https://github.com/ppazos/openehr-uitemplates/tree/master/xsd

[6] Pazos, P., EHRGen: Generador de Sistemas Normalizados de Historia Clínica Electrónica Basados en openEHR, CAIS, 2012 .
http://41jaiio.sadio.org.ar/sites/default/files/15_CAIS_2012.pdf

[7] Beale, T., et. al., EHR Information Model, openEHR Foundation,, 2008.
http://www.openehr.org/releases/1.0.2/architecture/rm/ehr_im.pdf

[8] Beale, T., Archetype Object Model, openEHR Foundation,, 2008.
http://www.openehr.org/releases/1.0.2/architecture/am/aom.pdf

[9] Beale, T., Heard, S., Archetype Definition Language ADL 1.4, openEHR Foundation, 2008.
http://www.openehr.org/releases/1.0.2/architecture/am/adl.pdf

[10]ISO 13606-2:2008, Health informatics -- Electronic health record communication -- Part 2: Archetype interchange specification
http://www.iso.org/iso/catalogue_detail.htm?csnumber=50119

[11]Schach, S., Object-Oriented and Classical Software Engineering, 5th ed., p. 12, McGraw-Hill, 2002.

[12]Archetype Editor, Ocean Informatics
http://www.openehr.org/downloads/archetypeeditor/home

[13]The Extensible HyperText Markup Language (Second Edition), W3C, 2002.
http://www.w3.org/TR/xhtml1/

[14]HTML5, W3C, 2014.
http://www.w3.org/TR/html5/

[15]XAML Overview (WPF), Microsoft.
http://msdn.microsoft.com/en-us/library/ms752059(v=vs.110).aspx

[16]SwiXml.
http://www.swixml.org

[17]XUL, XML User Interface Language, Mozilla.
https://developer.mozilla.org/en-US/docs/Mozilla/Tech/XUL

[18]Beale, T., et. al., openEHR Data Types Information Model, 2008.
http://www.openehr.org/releases/1.0.2/architecture/rm/data_types_im.pdf

[19]Herrington, Jack. Code Generation in Action. Greenwich : s.n., 2003. 1930110979.

[20]Archivo Map con reglas para el generador de IU https://github.com/ppazos/openehr-uitemplates/blob/master/generator/map/html2.map

[21]Definición de variables de correspondencia utilizadas por el generador de IU https://github.com/ppazos/openehr-uitemplates/blob/master/generator/VariablesInfo_v4.xml

[22]Ejemplo de UITemplate en XML. https://github.com/ppazos/openehr-uitemplates/blob/master/uitemplates/uitemplate_simple.xml

[23]Groovy. A dynamic language for the Java platform. http://groovy.codehaus.org/

[24]Groovy. XMLSlurper http://groovy.codehaus.org/Reading+XML+using+Groovy's+XmlSlurper

[25]XML Path Language (XPath), W3C, 1999. http://www.w3.org/TR/xpath/

Dirección para correspondencia

Ing. Pablo Pazos Gutiérrez

pablo.pazos@cabolabs.com