Model Driven Development of Clinical Information Sytems using openEHR

Koray ATALAG [a,1], Hong Yul YANG [a], Ewan TEMPERO [a], Jim WARREN [a,b]
[a]*Department of Computer Science,* [b]*National Institute for Health Innovation*
The University of Auckland, Auckland, New Zealand

Abstract. openEHR and the recent international standard (ISO 13606) defined a model driven software development methodology for health information systems. However there is little evidence in the literature describing implementation; especially for desktop clinical applications. This paper presents an implementation pathway using .Net/C# technology for Microsoft Windows desktop platforms. An endoscopy reporting application driven by openEHR Archetypes and Templates has been developed. A set of novel GUI directives has been defined and presented which guides the automatic graphical user interface generator to render widgets properly. We also reveal the development steps and important design decisions; from modelling to the final software product. This might provide guidance for other developers and form evidence required for the adoption of these standards for vendors and national programs alike.

Keywords. EHR, HIS, openEHR, interoperability, GUI.

Introduction

In this paper, we present the development methodology of a .Net/C# desktop application (GastrOS) for endoscopy reporting which is driven by openEHR models.

The main drivers of such a model driven software development are:

1) Transfer of domain knowledge from healthcare professionals into software is ineffective using traditional development process where technical professionals need to capture and transform this into code. Put simply, software can be as good as this *hand-over* [1]. Two-level modelling technique in openEHR, essentially a model driven approach, allows clinicians to engineer knowledge using high-level tools which is then fed into the technical environment and consumed readily. This ensures the requirements are correct, complete and collected in a timely fashion.

2) The main challenge in achieving semantic interoperability lies in the non-technical domain and has to do with establishing common language, sharing data set definitions and creating computable information and knowledge artefacts [2]. openEHR defines methods and processes which meets these requirements.

3) The main determinant of software cost is the maintenance phase [3]. Healthcare is no exception. Redevelopment due to modifications includes redesign, coding, testing and deployment, which is very costly. Therefore being able to introduce these changes

[1] Corresponding Author: Koray Atalag, MD, PhD. Department of Computer Science, The University of Auckland, Private Bag 92019 Auckland 1142, New Zealand; k.atalag@auckland.ac.nz.

by modelling without redevelopment is very tempting and can potentially reduce the total cost of health information systems (HIS) significantly.

We have selected digestive endoscopy as the clinical domain which is a niche area with excellent standardisation of domain content. The Minimal Standard Terminology for Digestive Endoscopy (MST) contains a "minimal" list of terms and structure which is used to record the results of an endoscopic examination [4]. It provides a simple and uniform hierarchy for data entry which allows for consistent and intuitive generation of graphical user interfaces (GUI) automatically.

1. Methods

openEHR formalism effectively separates domain knowledge from software code using domain specific models called *Archetypes*. This is commonly known as Two-Level Modelling. Archetypes (top level) represent clinically meaningful concepts such as blood pressure measurement. They use common technical building blocks expressed in Reference Model (RM) (lower level). In the runtime software are driven by these models for dynamic GUI creation, data binding, validation and querying [1]. Thus altering software after deployment mainly involves remodelling by domain experts without the need for another redevelopment cycle.

RM consists of a small set of technical models which depicts the generic characteristics of health records (e.g. data structures and types) and context information to meet ethical, medico-legal and provenance requirements. In GastrOS RM entities usually correspond to individual GUI widgets. Archetypes provide the semantics and structure of domain concepts. They constrain RM building-blocks and form a computer processable model. Practically they specify particular record entry names, data structures, data types, value sets and default values. It is also possible to link each data item to biomedical terminologies. openEHR *Templates* bring together relevant Archetypes to define higher level models such as a discharge summary. Tighter constraints can be put on Archetypes (e.g. exclude some data items and values or renaming them). During implementation Templates are serialised into operational templates which contain all the structure and data items in included archetypes.

1.1. Modelling

A number of openEHR Archetypes have been created using the free and open source (FOSS) openEHR Archetype Editor. The following sections from MST are included:

- Examination Information: consists of reasons for endoscopy, examination characteristics and complications.
- Endoscopic Findings: for each organ using MST hierarchy, terms, attributes, attribute values and anatomical sites.
- Interventions: diagnostic and therapeutic procedures performed.
- Diagnoses: list of diagnoses for each organ.

These sections are then filled with appropriate entry archetypes which further chain a myriad of structural archetypes carrying bulk of the MST content. Finally openEHR templates have been created using the Ocean Template Designer for each of the three examination types: upper and lower gastrointestinal endoscopy and ERCP.

1.2. Implementation

GastrOS has been developed using the .Net platform and C# programming language. We have used MS Visual Studio 2008 IDE. The C# openEHR library[2] (openEHR.Net) has been included in the project which implements the 1.0.1 release of the openEHR RM and Archetype Object Model (AOM) specifications. It is used to build applications by composing RM objects, validate against AOM and serialise to/from XML [5].

GastrOS architecturally consists of a simple wrapper application which is used for patient management, and the model-driven structured data entry (SDE) component. SDE takes in an operational template and dynamically creates appropriate GUI forms. This component has the additional capability of validating and persisting data. SDE follows the model-view-controller (MVC) paradigm; such that the user interaction and presentation logic is completely independent of the logic for handling and persistence.

SDE first parses the input operational template into a tree-like data structure which consists of archetype objects conforming to AOM. Each archetype object acts as a blueprint for a specific part of the data to be entered and stored, as well as sets its GUI widget. SDE defines a set of mapping rules to determine what kind of GUI widget to create for what kinds of data elements. For example it would create a text box for a textual entry (e.g. name of a drug), a drop-down list for a restricted range of values (e.g. organ types), or a panel for a composite value that further contains sub-values (e.g. a list of diagnoses). These rules, which are fairly generic so as to accommodate as wide a range of clinical domains as possible, are combined with the novel GUI directives to finely adjust the aesthetics and visual behaviour of the GUI. Currently there is a hot debate about these directives as the openEHR formalism does not provide any means to handle presentation of information. It is generally accepted that this should be modelled as a different layer along with the archetypes and templates. So far studies in this area have been very scarce in the literature [6,7]. We have taken a more practical approach and exploited the annotations property in openEHR Templates, which can be defined for any data item at a distinct path in any included archetype.

A skeleton data instance, which we call as *value instance* to hold the user entered data, is created initially by the GUI generator at once which comprises only the top level hierarchy and mandatory items depicted by the RM. Then, the GUI generator recursively creates the associated widgets on the form. Each widget, representing a leaf node data item, instantiates its own value instance and then binds to the skeleton. By this way, an exact representation of the AOM is formed. During data entry, if the user wants to create additional instances of certain data elements where multiplicity is allowed, additional data instances are appended to the skeleton. When the user decides to save, parts of the value instance which correspond to the empty widgets are first pruned and then serialised into XML and persisted saved in a relational database (both MS Access and SQLite). When a value is cleared (set to null) after data have been committed, that part of the value instance is removed from the value instance.

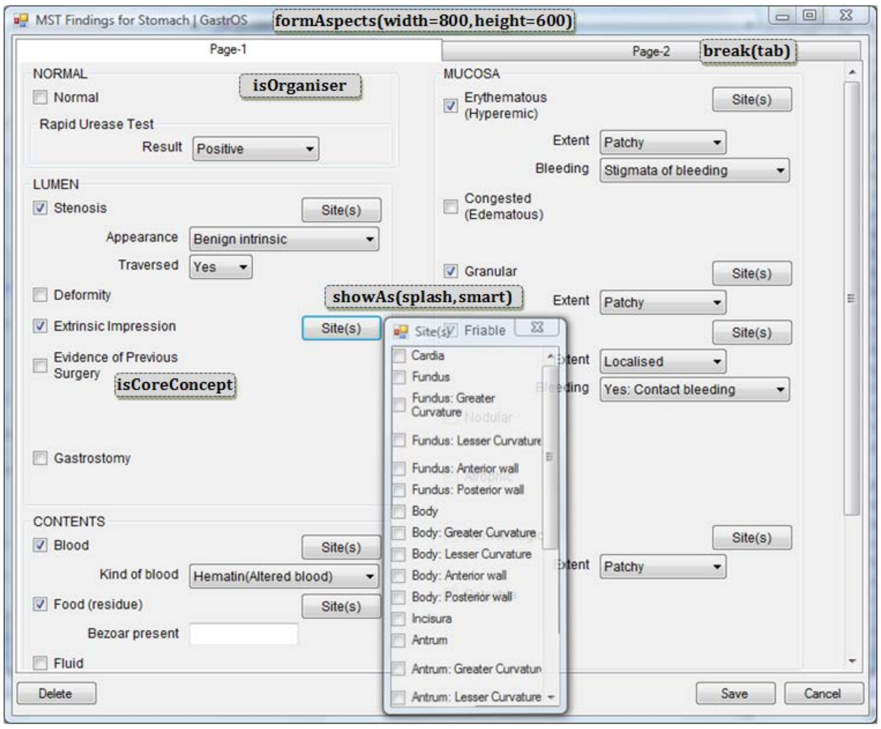
2. Results

Box 1 shows some of the pertinent GUI Directives defined by our group which appear in Figure 2. For the full list please refer to the project website [8].

[2] openEHR.Net has initially been developed by Ocean Informatics Pty. Ltd. and then extended by our team.

isOrganiser: when this is set item will be <u>displayed as a group</u> (e.g.within a frame, form etc.) which will contain all its children. The Heading items in MST, such as NORMAL, LUMEN, STENOSIS etc., have this directive set causing them to be displayed in groups within a frame. Any container item will simply be ignored when isOrganiser is not set and will be grouped under the first isOrganiser parent (if any). This simplifies working with highly nested clinical models.
isCoreConcept: We assume that Core Concepts are real-world entities which we can talk about their absence. For example a clinical finding (tumour, bleeding etc.) can be reported as present but also as absent or unknown. However it doesn't make sense to report absence of tumour grade or physical examination. This directive depicts that an item with all its children (if any) will be <u>handled and repeated as a whole</u> on the GUI and saved data.
When data are saved it wouldn't make sense to repeat attributes of a clinical finding defining its nature. For example in Figure 2 when Stenosis term is selected as a finding it should not have more than one Appearance attribute because the values might be mutually exclusive or potentially contraindicate with other selected attributes. Rather, the Core Concept as a group should repeat with a different set of attributes and values. An exception is the anatomical sites; in most cases more than one site will be involved. When data are saved, for each core concept only one attribute can be expected and one or more anatomical sites. The example below illustrates a case with a repeating attribute where values are mutually exclusive and should not be permitted (second):
<Stenosis; Appearance=Extrinsic, Traversed=Yes, Sites=Cardia,Fundus,Incisura>
<Stenosis; Appearance=Extrinsic, <u>Traversed=Yes, Traversed=No</u>, Sites=Cardia,Fundus,Incisura>
showAs (form|splash, modal|modeless|smart**):** this determines the behaviour when an item's values or children are displayed. The item's label will be shown as a reference (e.g. link, button or similar) and the contents will be shown on another page, a separate form (form) or a pop-up screen (splash). (smart) parameter causes to create a modeless form which closes when loses focus which saves one click during data entry.

Box 1. Pertinent GUI Directives defined and used in GastrOS.


MST Findings for Stomach | GastrOS
formAspects(width=800,height=600)

Page-1

NORMAL
isOrganiser
☐ Normal
Rapid Urease Test
Result: Positive

LUMEN
☑ Stenosis — Site(s)
Appearance: Benign intrinsic
Traversed: Yes
☐ Deformity
showAs(splash,smart)
☑ Extrinsic Impression — Site(s)
☐ Evidence of Previous Surgery
isCoreConcept
☐ Gastrostomy

CONTENTS
☑ Blood — Site(s)
Kind of blood: Hematin(Altered blood)
☑ Food (residue) — Site(s)
Bezoar present:
☐ Fluid

Page-2
break(tab)

MUCOSA
☑ Erythematous (Hyperemic) — Site(s)
Extent: Patchy
Bleeding: Stigmata of bleeding
☐ Congested (Edematous)
☑ Granular — Site(s)
Extent: Patchy
☑ Friable — Site(s)
Extent: Localised
Bleeding: Yes: Contact bleeding
Site(s)
Extent: Patchy

Site(s)
☐ Cardia
☐ Fundus
☐ Fundus: Greater Curvature
☐ Fundus: Lesser Curvature
☐ Fundus: Anterior wall
☐ Fundus: Posterior wall
☐ Body
☐ Body: Greater Curvature
☐ Body: Lesser Curvature
☐ Body: Anterior wall
☐ Body: Posterior wall
☐ Incisura
☐ Antrum
☐ Antrum: Greater Curvature
☐ Antrum: Lesser Curvature

Delete
Save
Cancel

Figure 2. Sample GUI form with associated GUI directives.

3. Discussion and Conclusion

The preliminary results of our larger study indicated that the openEHR based application, on the average, took nine times less time and were seven times less complex to implement; thereby making it significantly more maintainable [9]. Considering the paramount contribution of maintenance phase to total software cost (approximately 70-80%) this may translate into significant cost savings [3].

Since endoscopy domain is a narrow domain it can be argued that generalisability of our results will be limited. However as we experiment with other domains, such as anatomical pathology, our initial impression is that the GUI Directives may be applicable beyond endoscopy. However current work to extend GastrOS model to include generic archetypes such as Blood Pressure and Adverse Reactions revealed that further additions to the GUI Directives presented in this study are required; therefore more work is needed.

With regard to software usability, since the appearance and behaviour is depicted rather mechanically, good usability principles can be embedded into program logic and may result in more consistent GUI.

GastrOS source code, models and documentation have been published on Codeplex (http://gastros.codeplex.com) as FOSS software to enable wider dissemination of research results and also to foster collaboration [8].

In conclusion, we believe this study will help materialise how the model driven methodology, brought about by openEHR, works and bridges the gap between modelling and software development. Another important premise is the potential for enabling a high level of semantic interoperability among different HIS which is particularly important in developed jurisdictions, such as Europe, where this is not only desirable but essential.

Acknowledgments

This work was supported by a research grant from the University of Auckland (Project No: 3624469/9843).

References

[1] Beale T. Archetypes: Constraint-based domain models for future-proof information systems. In: Eleventh OOPSLA Workshop on Behavioral Semantics: Serving the Customer. Seattle, Washington, USA: Northeastern University; 2002. p. 16-32.
[2] ISO TR 20514 - Electronic Health Record Definition, Scope and Context. ISO; 2005.
[3] Sommerville I. Software Engineering. 6th ed. Addison Wesley; 2000.
[4] Delvaux M, Korman L, Armengol-Miro J, Crespi M, Cass O, Hagenmüller F, Zwiebel F. The minimal standard terminology for digestive endoscopy: introduction to structured reporting. International Journal of Medical Informatics 1998 Feb;48(1-3):217-225.
[5] openEHR.Net Programming Library. Available from http://openehr.codeplex.com
[6] Schuler T, Garde S, Heard S, Beale T. Towards automatically generating graphical user interfaces from openEHR archetypes. Stud Health Technol Inform 2006;124:221-6.
[7] van der Linden H, Austin T, Talmon J. Generic screen representations for future-proof systems, is it possible? There is more to a GUI than meets the eye. Comput Methods Programs Biomed 2009 Sep;95(3):213-226.
[8] GastrOS Endoscopy Application Project. Available from: http://gastros.codeplex.com
[9] Atalag K, Yang HY, Warren J. On the maintainability of openEHR based health information systems – an evaluation study in endoscopy. In: Proceedings of the 18th Annual Health Informatics Conference, HIC 2010. Melbourne, Australia: HISA; 2010 p. 1-5.