From openEHR Domain Models to Advanced User Interfaces: a Case Study in Endoscopy

Koray Atalag
University of Auckland Department of Computer Science
Private Bag 92019, Auckland 1142, New Zealand
koray@cs.auckland.ac.nz

Hong Yul Yang
University of Auckland Department of Computer Science
Private Bag 92019, Auckland 1142, New Zealand
hongyul@cs.auckland.ac.nz

Abstract

Good user interface is critical for the acceptance of health information systems. However due to the rapid and constant changes in the body of knowledge and medical practice it is difficult to keep them intuitive and user friendly as systems evolve. There is a need for dynamic creation of graphical user interfaces which are as intuitive and user friendly as the ones manually designed. This paper presents a case study in which a standard means to fulfil this need has been explored by using openEHR Archetypes and Templates. We have implemented an endoscopic reporting application (GastrOS) using C#/.Net which is able to generate structured data entry forms on the fly based on the underlying domain model. Our main contribution is the development of 'GUI Directives' which provide visual hints to the GUI generator in order to correctly render data entry forms. We have presented these directives as well as implementation details using visual examples. Our results indicate that dynamically creating GUI using openEHR domain models is a viable approach.

1. Introduction

Having intuitive and user friendly graphical user interfaces (GUI) has proved to be an extremely important factor for the acceptance of health information systems [1]. However it requires a significant amount of effort and time to design and implement them. The biggest challenge comes into play when making changes after deployment due to rapidly changing knowledge and the high variability of medical practice. Making radical changes to GUI is almost guaranteed which essentially translates into high cost. Therefore it has long been a compelling idea to be able to separate GUI from program logic and database, and then generate them dynamically.

Various approaches for managing GUI have been developed. The model-view-controller (MVC) paradigm has widely been established as a de facto standard for developing data-driven GUI applications [2]. Many web and application development frameworks that support MVC exist, e.g. Spring MVC [3], Apache Struts [4], Ruby on Rails [5] and Mozilla's XForms implementation [6], to ease the burden of creating GUI applications from scratch. Some database management systems such as Oracle Forms and Microsoft Access allow automatic generation of rudimentary user forms based on the underlying table schema. Similar techniques exist for automatic creation of GUI from program objects instead of database tables; Naked Objects [7] and OpenXava [8] are both Java-based frameworks that allow automatic creation of graphical forms based on Java objects with some degree of customisation.

Using general-purpose methods for GUI creation has one major shortcoming: data usually becomes difficult to manage and query over time because underlying data model usually degrades in quality and linking to previous versions of data becomes cumbersome. This is especially important in healthcare domain where interoperability and semantic coherence of clinical information is of utmost importance. Another problem is that the needs of domain experts need to be translated into technical requirements which then require developers to implement.

In the healthcare domain openSDE has been developed to tackle the automatic GUI creation and persistence by means of a formal domain model [9]. However its uptake has not been wide enough due to the propriety means of modelling. Therefore a standard means to manage GUI, data model and their binding has been needed.

We have chosen the openEHR formalism, a modelling and architectural standard for electronic health records (EHR), to develop an endoscopy reporting application with advanced GUI for structured data entry. As a novel contribution we have developed a number of 'GUI Directives' to be used in conjunction with the domain models for automatically generating GUI forms. In this paper we present these directives and also flesh out the implementation methodology.

2. openEHR Formalism

openEHR refers to the publicly accessible engineering specifications to build full blown EHR systems and also to the governing body – the openEHR Foundation [10]. At the heart of the formalism lies the Archetype paradigm which is a way of constructing formal domain models by means of putting constraints on a common reference (information) model (RM). Thus the methodology is commonly known as Two-Level or Multi-Level Modelling (MLM). A good analogy to understand how RM, archetypes and terminologies relate to each other is using a limited set of standard LEGO® blocks to assemble many different structures (Figure 1). There is clear separation of domain knowledge from the technical environment which helps to separate tasks of developers and domain experts. The main premise of the formalism with regard to GUI is that openEHR MLM provides the standard means to manage GUI and persistence layers consistently in HIS.

In MLM, otherwise hard-coded definitions are expressed in the form of RM, Archetypes and Templates to create runtime functions and GUI of software on the fly. Thus if there is a need to make changes to GUI software does not need to be redesigned, coded, tested and deployed [11].

RM consists of a small set of object oriented classes which depict the generic characteristics of health records (i.e. data structures and types) and the means to define context information to meet ethical, medico-legal and provenance requirements. Thus a blood pressure measurement will be represented as an instance of RM OBSERVATION class, which is a sub-class of ENTRY class. These carefully engineered classes can faithfully capture results of all medical entries together with necessary contextual information such as cuff size (of a sphyngomanometer) and position (i.e. sitting, lying) in order for correct medical interpretation. The stable and well-defined RM entities usually correspond to individual GUI widgets; such as a CLUSTER container data structure can be represented as a frame around its children elements or the DV_TEXT data type corresponds to a text box control or a drop down list. Therefore by using standardised visual elements GUI is expected to be consistent and extensible.

Archetypes provide the full semantics and structure of domain concepts by constraining relevant classes from RM and then use them as building-blocks to form a computer processable clinical model. Practically they specify particular record entry names, data structures, data types, prescribed value ranges and values for some of the context attributes.

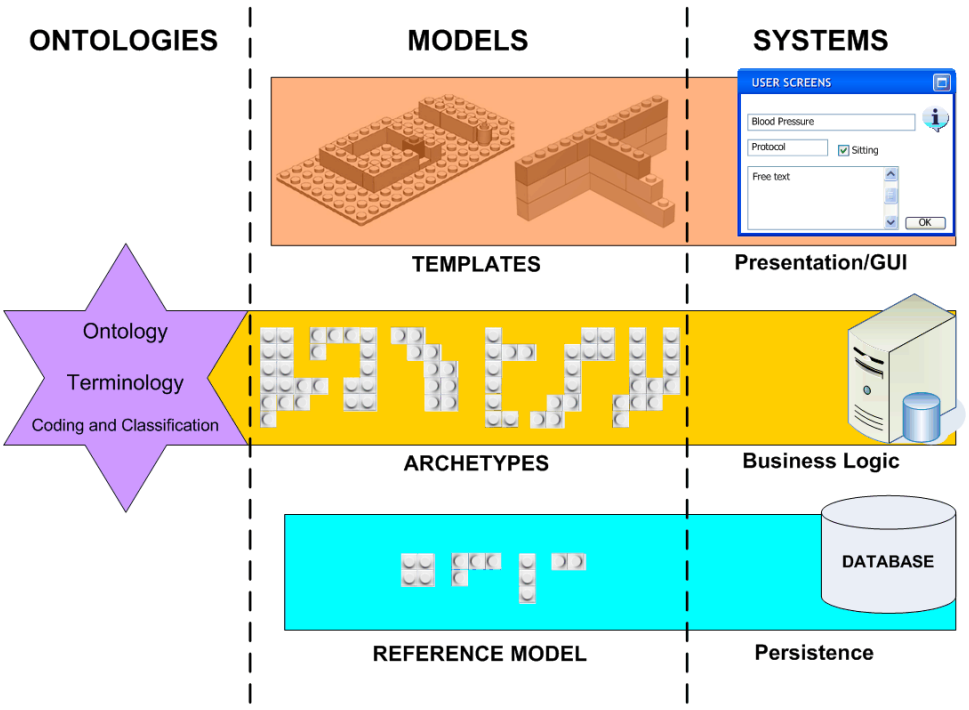

Figure 1. Schematic representation of the openEHR Multi-Level Modelling Paradigm

From the point of GUI, Archetypes provide two crucial elements: 1) They help domain experts to specify information capture requirements themselves without relying on developers; i.e. definition of clinical forms; 2) they make possible to leverage the vast amount of standardised terms and semantics by linking to biomedical terminologies – thus ensuring consistency within and among HIS which may be expected to help control the changeability and variability in Medicine.

On top of Archetypes one more layer exists: Templates which assemble numerous Archetypes and further constrain them. During implementation templates provide the means to automatically create GUI. They combine a set of archetypes, terminologies, language and other details for local use. Templates can be used for designing screen forms and also as persistence models.

These high quality specifications have recently been adopted by both CEN and ISO, and released as 13606 suite of *de jure* health informatics standards.

3. The Study Domain: Endoscopy

Gastrointestinal endoscopy has been selected as the clinical domain on the basis that it is a niche domain with excellent standardisation of domain terminology and information capture requirements. Most of the domain knowledge in computer processable form is available through comprehensive and widely accepted international terminology – the Minimal Standard Terminology for Endoscopic Gastroenterology (MST) [12]. MST contains a "minimal" list of terms and structure that could be included within any computer system used to record the results of an endoscopic examination. While creating the clinical expressions one needs to follow the depicted uniform hierarchy and specify a class/heading (i.e. Protruding Lesions), term (Tumour/Mass), attributes (i.e. Type), attribute values (i.e. small, submucosal and bleeding), anatomical sites (fundus and cardia of stomach) and interventions (i.e. biopsy).

Formerly by the primary author, a research prototype has been built for the purpose of validating MST for clinical use and usability for building an endoscopic application with a novel GUI design. While clinical evaluation was very positive, from a technical point of view making changes to software was quite problematic [13,14]. As the name implies, this “minimal” terminology had to be extended frequently for local needs which almost always resulted in significant development effort for GUI redesign, data model changes and data binding of GUI elements. This has been the main motivation for the study and resulted in implementing a new application – named **GastrOS** – using openEHR MLM.

4. Materials and Methods

MST has been checked thoroughly for errors and inconsistencies before starting with the modelling work. The organisation and depicted hierarchy for specifying endoscopic findings and other clinical expressions have been transformed into a comprehensive mind map using the free XMind tool [15]. This has been very useful for communication with domain experts and getting advice. Afterwards, a number of openEHR Archetypes have been created using the free and open source openEHR Archetype Editor. This consisted of a top level COMPOSITION archetype which held sections for representing the whole MST:

- Examination Information (consists of reasons for endoscopy, examination characteristics and complications)
- Findings (for each organ using the depicted hierarchy and a set of anatomic sites)
- Interventions (additional diagnostic and therapeutic procedures)
- Diagnoses (for each organ)

These sections are then filled with appropriate ENTRY archetypes which further chain a myriad of structural CLUSTER archetypes carrying MST content. Finally three separate openEHR templates each constraining the single COMPOSITION archetype for the three examination types have been created using the Ocean Template Designer. The model as viewed from the tools is given in Figure 2.

As the Archetypes are destined to model domain knowledge only any implementation specific or local requirements including GUI are left to Templates. However the current openEHR Template specifications do not define the means to provide any GUI related information. Therefore we have exploited the “Template Annotations” to provide hints to our automatic GUI generator. Our novel contribution with regard to GUI aspects is a set of “GUI Directives” which we have defined and implemented in GastrOS. These are presented in the Results Section.

The next step was to generate operational templates to generate GUI and data bindings by using the same tool. An operational template is an XML based serialised form of the model which can be directly consumed by applications.

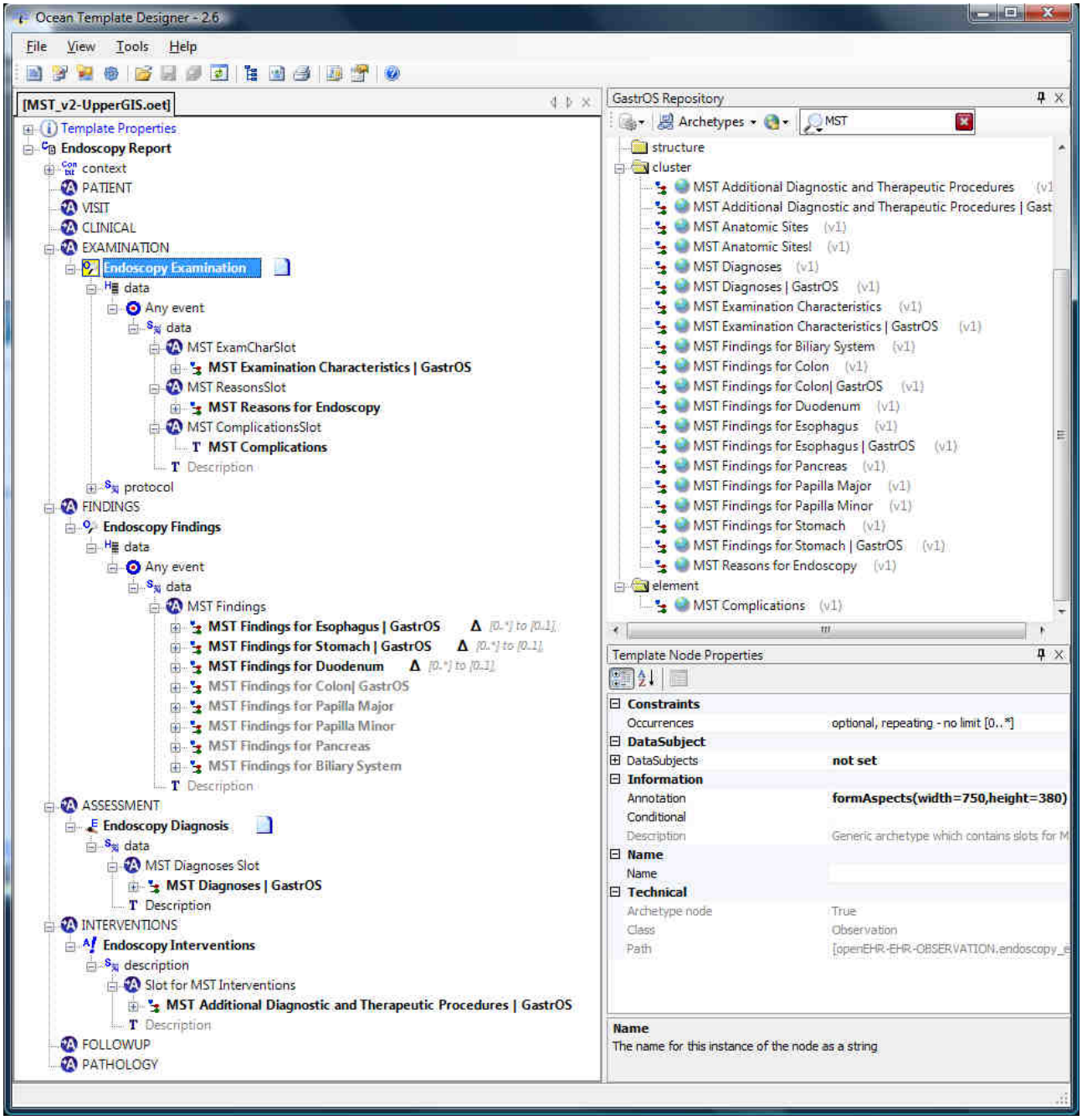

Figure 2. The template for upper gastrointestinal endoscopy

We developed GastrOS using the .Net platform and C# programming language because we have foreseen the possible deployment as a desktop application on MS Windows boxes. We have used MS Visual Studio 2008 IDE and the C# openEHR RM library provided by Ocean Informatics Pty. Ltd. GastrOS's core functionality is handled by the model-driven structural data entry (SDE) component, which takes in an operational template and dynamically constructs appropriate GUI forms. The SDE component has the additional capability of validating user-entered data and persisting the data across user sessions. The SDE follows the model-view-controller paradigm faithfully, such that the user interaction and presentation logic is completely independent of the logic for handling and persisting openEHR RM objects. If, later in the development cycle, the domain model changes, then the SDE would automatically generate updated GUI forms that reflects these changes, without any need for modification of program code.

To make this work, the SDE first parses the input operational template into a tree-like data structure, called archetype objects. Each archetype object acts as a blueprint for a specific part of the data to be entered and stored, as well as the GUI widget to represent the data (Figure 3). It can be as atomic as a single textual entry or as complex as an entire group of findings. The 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 field for a textual entry (i.e. name of a drug), a drop-down list for a restricted range of values (i.e. organ types), or a panel for a cluster value that further contains sub-values (i.e. a diagnosis entry). These rules, which are fairly generic so as to accommodate as wide a range of usage domains as possible, are combined with the GUI directives to finely adjust the aesthetics and visual behaviour of the widgets generated by the SDE.

5. Results

The GUI directives are presented in Box 1.

A sample GUI form from GastrOS representing upper gastrointestinal endoscopy findings for stomach with multiple terms, attributes and anatomic sites is given in Figure 3. This example demonstrates the use of several of the GUI directives shown above. The size of the form is determined by the *formAspects* directive. Each of the MST headings (e.g. "NORMAL", "LUMEN", etc.) is displayed as a framed container due to the *isOrganiser* directive. Each MST term is marked with the *isCoreConcept* directive, which renders it as a special checkboxed container, whose inner contents are only displayed when the box is checked. The anatomical sites associated with each MST term are displayed specially through a splash form, as indicated by the *showAs(splash, smart)* directive. Finally, the form is divided into two tabs in order to utilise the given form space, and that is achieved through the *break(tab)* directive.

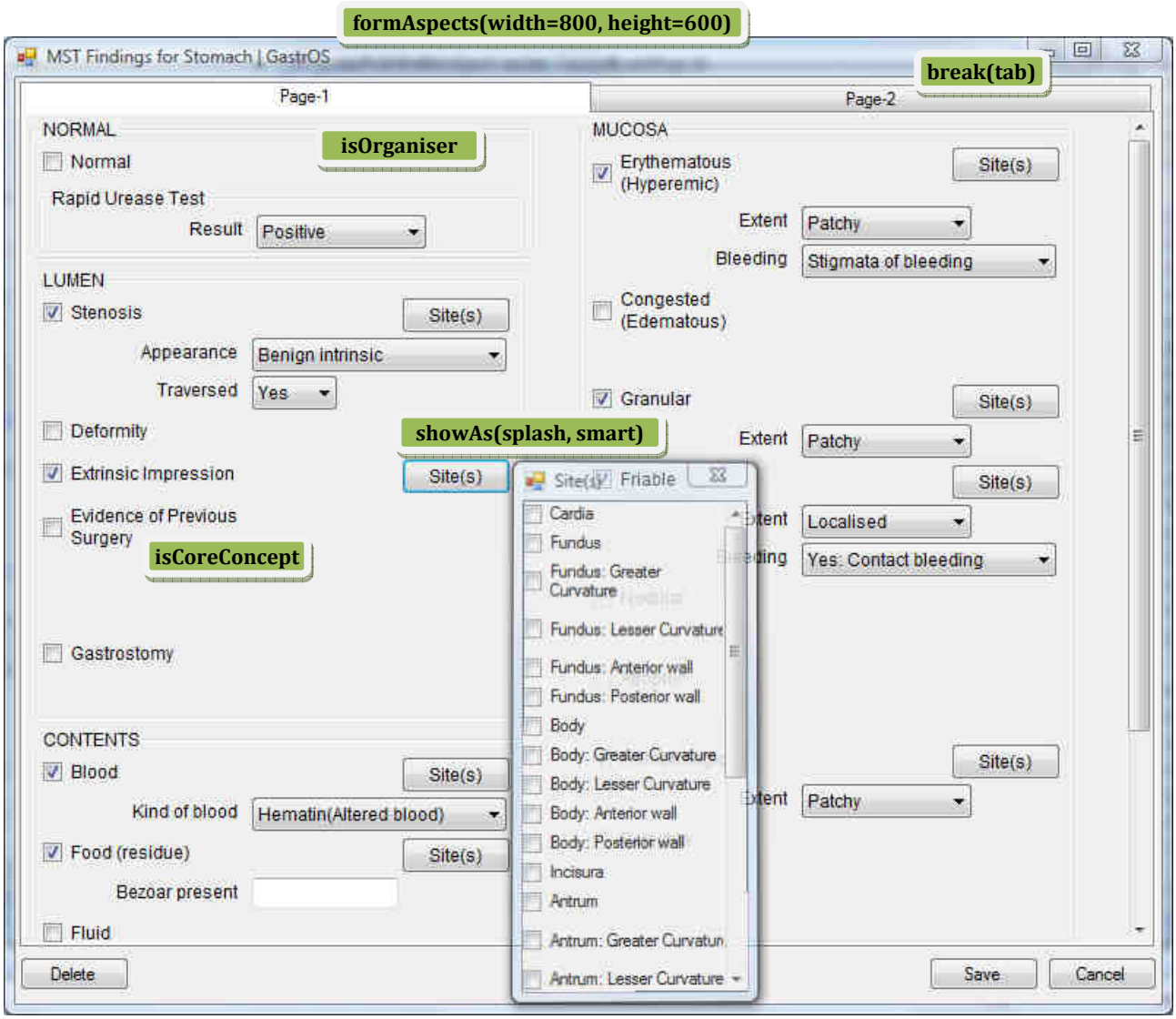

Figure 3. Upper gastrointestinal endoscopy findings for stomach GUI from GastrOS, with examples of used directives shown in round boxes

- **isOrganiser (g)**: when this is set then it will be shown as a group which will contain all its children (i.e. as a frame, form etc.). A container object will simply be ignored when this is not set.
- **isCoreConcept (g)**: This is an abstract concept; but we can say that Core Concepts are real-world entities which we can talk about their absence (i.e. a clinical finding, a disease but not tumour grade or physical examination). The directive depicts whether a node with all its children (if any) shall be handled and repeated as a whole in an archetype (i.e. makes sense together such as a clinical finding with other attributes defining its nature). When the node and/or its children are selected, its presence information is stored in the corresponding ELEMENT node which records this (i.e. in MST Findings archetypes [Present?] node).
  In this case this node by becomes the "**default axis**" (i.e. if a clinical finding is selected as a core concept which has a number of descriptions and anatomical sites within then it can be said that this is a *"finding oriented"* view or in other words the default axis is "finding"). There can be an "**alternate axis**" in a Core Concept, in some cases more than one, which means that all the attributes and the default axis can be regrouped and rendered to an alternative view (i.e. this becomes a "*site oriented*" view). The alternate axis will be depicted by another directive.
  Splitting out of GUI container is not allowed unless **allowChildrenSplit** directive has also been set.
- **showAs (form|splash, modal|modeless|smart) (g)**: Determines the behaviour when node's values or children are displayed.The node's label is shown as a reference (i.e. link, button or similar) and the contents will be shown on another modal form - (**form**) or on a pop-up form (**splash**). (**smart**) is a special type of modeless form which closes when loses focus).
  - If this is a leaf node (i.e. ELEMENT) then its values will be listed to be selected (depending on the cardinality and occurrences allowed in the archetype single or multiple selection will be possible.
  - If this is not a leaf node then the node and all its children will be displayed using regular display options.
- **showWithParent (simple|merged)(g)**:This directive applies to nodes of type ELEMENT and causes display of its widget (label and values) at the same level as its immediate parent (**simple**); or display without label (**merged**) does not need further specification because the semantics is implied with the term alone (i.e. term tumour with attributes malignant/benign). Sites is an example for simple where it will simply be shown next to term. This directive is intended to make GUI simpler and require less space.
- **break (next|parent|tab) (g)**: causes the node and rest of the model to appear in the next column but within the same organiser (**next**); in the next column within a separate organiser which is semantically the continuation of current one (**parent**); in a new tab within the same form (**tab**).
- **formAspects (width=pixels, height=pixels, position, resize:none|hor|ver|both, startup mode etc.) (g)**: determine form's all aspects (the keywords TBD)
- **showText (multi|smart) (g)**: this applies to free text nodes (ELEMENT DV_TEXT). Only free text or coded list can be specified at archetype level. Normally a single line text box will be displayed; when this directive is used it will be possible to depict whether:
  - multiple lines are allowed but within a fixed height - with scroll bars (**multi**)
  - multiple lines are allowed in a text box which may expand (vertically and horizontally) to accommodate the content when entering data. When it loses focus it will return back to its initial size but will let use know that there is more content inside than can fit visible area. When user wants to see its contents (i.e. when control gets focus or mouse hover) it will expand again to show full contents (**smart**).
- **showValueContext (append|organise) (g)**: this applies to ELEMENT and cause the display of Description and Term together. Need for representing anatomical sites from multiple organs in single widget. It will simply append two or when used with organise parameter it will show distinct descriptions as organisers (i.e. as Separators in pick lists or separate columns with headings in Splash forms).
- **showInstances (number) (g)**: this applies to nodes where multiplicity is allowed and sets the number (n=positive integer) of repetitions displayed initially on GUI. Used for MST Diagnoses on form.
- **showDescription (g)**: when set text in "Description" of term_id will be shown together with label (later can be further specialised as appended or tooltip).

Box 1. GUI Directives developed and used in GastrOS. Directives in red/black are implemented/not implemented and g denotes that it is generic (i.e. not domain specific)

6. Discussion / Conclusion

This work has been undertaken as part of the larger research study being conducted to assess the software maintainability of openEHR based software. GastrOS is functionally an exact copy of an earlier research prototype which has been clinically used successfully. The GUI design has been replicated by using openEHR formalism together with our GUI Directives. We have shown that almost all visual appearance and functionality of the previous GUI have

successfully been provided by the new implementation. Therefore we propose that dynamically creating clinically usable GUI using openEHR domain models and presentation hints such as our GUI Directives is a viable approach.

Since endoscopy domain is a highly specialised and narrow domain it can be argued that generalisability of our results is limited. However we have experimented with other domains, such as anatomic pathology, and although not formal our initial impression is that the GUI Directives are applicable equally. More work is needed towards this direction.

GastrOS is a free and open source project which is still under development. It is hosted on the SourceForge portal. Documentation, design artefacts and code can be found at this URL: http://gastros.sourceforge.net.

7. Acknowledgments

This work was supported by a research grant from the University of Auckland (Project No: 3624469/9843). Special thanks to Dr. Louis Korman for his help during modelling. We would like to express our gratitude to the Bedogni family who established the Clinton Bedogni Fellowship in Open Systems which enabled this research. Training, support and C# openEHR Reference Model library have been provided by Ocean Informatics Pty. Ltd.

8. References

[1] Sittig D, Kuperman G, Fiskio J. Evaluating physician satisfaction regarding user interactions with an electronic medical record system. Proceedings / AMIA . Annual Symposium. AMIA Symposium. 1999;:400-404.

[2] Burbeck S. Application Programming in Smalltalk-80: How to use Model-View-Controller (MVC). [Internet]. 1992;Available from: http://st-www.cs.illinois.edu/users/smarch/st-docs/mvc.html

[3] Johnson R, Hoeller J, Arendsen A, Risberg T, Sampaleanu C. Professional Java Development with the Spring Framework. First. Wrox Press; 2005.

[4] Apache Struts homepage [Internet]. [cited 2010 Jun 28];Available from: http://struts.apache.org/

[5] Ruby on Rails [Internet]. [cited 2010 Jun 28];Available from: http://rubyonrails.org/

[6] Mozilla XForms Project [Internet]. [cited 2010 Jun 28];Available from: http://www.mozilla.org/projects/xforms/

[7] Naked Objects homepage [Internet]. [cited 2010 Jun 28];Available from: http://www.nakedobjects.org/index.html

[8] OpenXava homepage [Internet]. [cited 2010 Jun 28];Available from: http://www.openxava.org

[9] Los RK, van Ginneken AM, van der Lei J. OpenSDE: A strategy for expressive and flexible structured data entry. International Journal of Medical Informatics. 2005 Jul;74(6):481-490.

[10] Kalra D, Beale T, Heard S. The openEHR Foundation. Stud Health Technol Inform. 2005;115:153-73.

[11] 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.

[12] Delvaux M, Korman L, Armengol-Miro J, Crespi M, Cass O, Hagenmüller F, et al. The minimal standard terminology for digestive endoscopy: introduction to structured reporting. International Journal of Medical Informatics. 1998 Feb;48(1-3):217-225.

[13] Atalag K, Bilgen S, Gur G, Boyacioglu S. Evaluation of the Turkish translation of the Minimal Standard Terminology for Digestive Endoscopy by development of an endoscopic information system. Turk J Gastroenterol. 2007 Sep;18(3):157-164.

[14] Atalag K. Archetype Based Domain Modeling for Health Information Systems GastrOS: Case Study on Digestive Endoscopy and Validation of the Minimal Standard Terminology (MST 2) [Internet]. VDM Verlag; 2009. Available from: http://www.amazon.com/exec/obidos/ASIN/3639134168/ejelta5-20

[15] XMind - Mind Mapping and Storming [Internet]. [cited 2010 Jun 28];Available from: http://www.xmind.net/