Using MDD to the Automatic Synthesis of Graphical User Interfaces for Health Information Systems

Iuri M. Teixeira[2], Regina M. Braga[1], Antônio Tadeu A. Gomes[2]

[1] Federal University of Juiz de Fora (UFJF), Juiz de Fora, MG, Brazil
regina.braga@ufjf.edu.br

[2] National Laboratory for Scientific Computing (LNCC) and
National Institute of Science and Technology in Medicine Assisted by Scientific Computing (INCT-MACC), Petrópolis, RJ, Brazil
{iuri,atagomes}@lncc.br

Abstract. Model-Driven Development (MDD) techniques provide a better articulation between domain experts and developers, allowing reductions on software development cost. Clinical data models based on open standard specifications facilitate the application of MDD techniques in the domain of Health Information Systems (HIS). Nevertheless, the use of clinical data models alone does not solve the fundamental problem of high development cost for HIS, since further information (e.g. architectural information) is usually necessary in the model transformation processes that MDD techniques employ. In this context, we present a code generation system that combines clinical data models based on *open*EHR specifications and high-level architectural descriptions based on constructs usually offered by Architectural Description Languages (ADL). We apply this system to the automatic synthesis of graphical user interfaces (GUI) for HIS, and argue about the hypothesis that our proposal can better articulate domain experts and developers of HIS, while simplifying the specification of model transformations.

Keywords: Model-Driven Software Development; Health Information System; openEHR; Model Transformation Component

1 Introduction

Models can capture the essence of reality in an abstract way in order to provide appropriate solutions for complex problems. Computable abstract models can turn the software development process more effective through the use of techniques such as Model-Driven Development (MDD) [1].

Applying MDD techniques for Health Information Systems (HIS) is still an important challenge if one wants to reach a higher level of interoperability and reuse [2]. Conversely, in the last years there has been a growing discussion about HIS effectiveness in healthcare [3]. Clinical data models based on open standard specifications for the conception of interoperable Electronic Health Records (EHR) [6] help with both, improving HIS effectiveness and establishing MDD techniques for HIS. These models allow a better articulation between domain experts and developers and can also help with the automatic code generation for HIS.

Nevertheless, clinical data models do not solve alone the fundamental problem of high development costs for HIS. Model transformation processes [7] are a key MDD technique and their application to clinical data models may vary considerably between HIS families, such as prehospital emergency and epidemiological surveillance [8][9]. Crucially, such processes need more information—in particular architectural information—that is not found in clinical data models.

The aim of this work is to present a system for the automatic synthesis of Graphical User Interfaces (GUI) for HIS. As input, the system gets clinical data models that follow the openEHR specifications [5], associated with architectural descriptions inspired by constructs present in the Acme Architectural Description Language (ADL) [10]. As output, the system provides GUI code for HIS. The generated code is a skeleton intended for mitigating the overall development effort and for guiding developers of HIS to build the complete functional system. An MDD technique was developed for this system, based on a repository of reusable rules that drive the transformation processes. The MDD technique embodies a strategy for combining both models that consists in an architectural style that describes GUI properties. These properties allow enriching components defined on HIS families – themselves also described as architectural styles – with GUI information. In this strategy, not only the implementation of HIS, but also the specification of transformation rules can be simplified, allowing cost reduction on HIS development. It is worth mentioning that the research effort presented herein is part of a broader project, called SPLiCE (Software Product Lines in healthCarE), which aims at the automatic synthesis of skeletons of complete HIS [8] based on the combination of MDD and software product line (SPL) techniques.

This paper is organized in five sections besides this introduction. In Section 2, we present the basic principles of MDD needed for our work. In Section 3, the proposed system is detailed. In Section 4, we illustrate the technical feasibility of our approach. In Section 5, we discuss related work. Finally, in Section 6 we provide some concluding remarks and possible paths for future work.

2 Theoretical Background

MDD techniques have the principle that changes that are made in a more abstract level can be propagated to more concrete levels of a given project, to mitigate redundancies and inconsistencies in the overall software development process. Two central concepts of MDD techniques that enable the principle of model specialization are *metamodels* and *model transformations*.

In the literature there are some practical uses for metamodels, such as the four layers of the OMG [21] specification: (M3) metametamodel; (M2) metamodel; (M1) model; and (M0) executable instances. The OMG specification defines the M3 layer as MOF (Meta Object Facility), a model that defines itself.

The work proposed in this paper uses Ecore as the metametamodel to provide the base of all MDD techniques developed in this work. Ecore is a subset of the MOF metametamodel implemented on the Eclipse platform [22], and the adoption of Ecore in the work proposed in this paper is related with the extensive tool support offered by such platform.

The ATL language (Atlas Transformation Language) [14] follows the OMG specification. In this work, we adopt the ATL language due to its full-fledged definitions, extensive documentation, and good integration with the Ecore metametamodel of the Eclipse platform. The ATL language offers support to declarative rules (*matched rules*) and imperative rules (*called rules*). Matched rules are developed on metadata between source metamodels and target metamodels. Matched rules have subtypes, such as *standard rules* and *lazy rules*. Standard rules are applied once for each metadata found in a source model. Lazy rules are applied many times for each element from a metadata found in a source model. Called rules are explicit calls with arguments. The main difference between matched rules and called rules is that the latter do not have source metadata and the target elements are simply created in an imperative way.

It has been observed along the years an increasing use of computing resources in health care [2]. In this context, [23] reflects on the promising future of Health Information Systems (HIS), highlighting the following potential benefits to the area: (a) better communication between different health care providers (hospitals, health centers, laboratories etc.), (b) elimination of unnecessary clinical studies, (c) readability in medical records and, especially, (d) fewer medical errors. Concrete examples of some of these benefits can be shown in different medical practices:

- Prehospital emergency: Information and Communication Technologies (ICTs) offers benefits in effective remote support to patient care, allowing experts to guide teams in primary care of patients in medical emergency situations (as in a mobile unit—ambulance—or in a geographically isolated region). The quick and correct decision in emergency pre-hospital care related to heart attacks, for example, allows to considerably reduce the mortality and morbidity of patients who suffer from this disease [4].
- Epidemiological surveillance: ICT allows to speed up tasks such as analysis and interpretation of data collected on public health, and can detect and notify events that require immediate attention to epidemiological outbreaks. The anticipation of actions from the detection of outbreaks is a priority in health, but this detection is very time consuming if done from diagnostic laboratories only. Therefore, epidemiological surveillance systems have significant potential to accelerate the detection of epidemic outbreaks [25].

Many solutions have been proposed in the literature for the HIS interoperability problem, most of them defining terminologies and ontologies for EHR [28]. However, the high cost of implementing and maintaining these EHR has inhibited its adoption [29]. The approach based on standard specifications for clinical data models—such as openEHR [5], HL7 [30] and MLHIM [31]—has been considered a feasible alternative for the design of interoperable EHR. In the case of openEHR specifications, technology concepts are defined in a Reference Model (RM). The RM—in fact a metamodel from the point of view of OMG layering—defines a set of general data types that can be used by multidisciplinary committees and medical technologists to produce the so-called "archetypes". Archetypes combine the data types defined in the RM [5] with clinical information types (observation, instruction, action and evaluation). Archetypes have their own metamodel—the Archetype Model (AM). Moreover, archetypes may also constrain the data types they combine to represent the clinical data of interest. As an example, the archetype "Blood Pressure", found in the Clinical Knowledge Manager repository [24] (Figure 1) of openEHR Foundation, is an archetype of observation type with clinical data elements such as "Systolic" and "Diastolic", which define constraints on the values accepted by the more general data type "Number" in the RM.

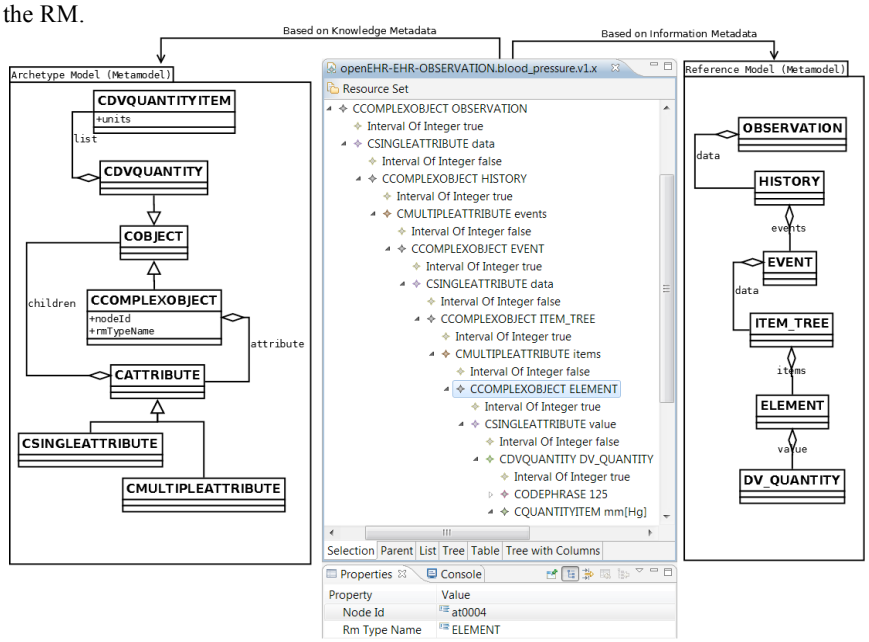

Fig. 1. Blood Pressure Archetype, considering information metamodels [5]

3 IMT – Interface Model Transformation System

In this section we present our system, named IMT (in Figure 2, "`Interface Model Transformation System`"), for the automatic code generation of GUI for HIS. This system gets as input the desired clinical data models and the intended architectural model and generates as output the GUI code. As illustrated in Figure 2, the proposed system has three main elements: "`M2M transformation`", "`M2C transformation`" and "`Transformation Rules Repository`". The first two elements are the components of the IMT system that perform transformations, respectively, for M2M and M2C processes. The third element is a rule repository of model transformations.

In Figure 2, the "`M2M Transformation`" component performs the synthesis of clinical data models related with the intended HIS architectural model. Based on these models, the transformation rules are retrieved from the "`Transformation Rules Repository`" and employed on such models. The result of the M2M transformation processes is an abstract model with the main domain information for platform-independent GUI ("`GUI Clinical Data Model`"). The M2C processes perform transformations of the models received from M2M transformation component to concrete GUI models ("`GUI Clinical Data Code`"). This transformation uses the RichUbi tool [11] to generate code in JSP (JavaServer Pages) and HTML (HyperText Markup Language), among others. It is important to note that other M2C tools can be used instead of the RichUbi tool, as long as the main information is contained in the models generated by the "`M2M Transformation`" component.

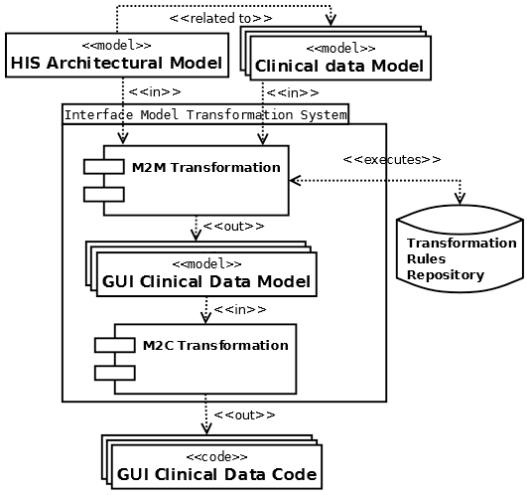

Fig. 2. System proposed for code generation of Graphic User Interface for HIS.

3.1 A Strategy for Clinical Data Transformation with Architectural Information

Architectural Description Languages (ADL) [27] can provide a formal basis to design a common representation for architectural information. While different ADLs focus on different aspects of architecture, Acme [10] provides a solid, yet highly flexible concept of common architectural constructs, such as *components, connectors, ports, roles, properties, configurations,* and *architectural styles*. For this reason, the strategy discussed in this section for model transformation with architectural information was inspired on some of such constructs found in the Acme language.

In our previous work [26], two different strategies were analyzed to define model transformation rules: (a) build different transformation rules for clinical data models considering different families of HIS, and (b) use architectural annotations models to clinical data models. The second strategy was improved in the present work through the use of three key architectural constructs inspired by the corresponding constructs found in the Acme language: *properties*, *components* and *architectural styles*. A key hypothesis considered in this work is that model transformation without considering architectural information can lead to a combinatorial explosion of rules and, consequently, to an additional cognitive overload to developers of transformation rules. As examples, we can mention HIS families, such as prehospital emergency and the epidemiological surveillance, discussed in [8][9], where different combinations of clinical data models occur due to the presence of different architectural structures:

- In a prehospital emergency system, we can identify two fundamental architectural components related to GUI: the “paramedic”, which represents the patient first care; and the “specialist”, which remotely helps the paramedic in the decision making of the procedure to be adopted. An example of this family, focused on the emergency care problem of heart attack victims, is described in [4]. In this family, observation data (such as "Blood Pressure"), are typically associated with GUI “form elements” on the paramedic component and “visualization elements” on the specialist component, whereas the opposite happens for instruction data (such as "Intervention to Be Adopted").
- In an epidemiological surveillance system, different combinations occur due to the presence of different architectural structures (a survey of these systems is presented in [12]). However, in general, there is a greater relevance of observation and evaluation clinical data (such as "clusters of cholera cases"), in contrast to the observation and instruction data of prehospital emergency systems, as well as a higher tendency to the data streams involved being unidirectional. In a family of epidemiological surveillance systems we can identify two architectural components related to GUI: "collector" and "epidemiologist". The first component performs collection of clinical observation data in form elements. The second component allows an epidemiologist to analyze the collected data, therefore, this component is related to visualization elements.

The strategy proposed in this work for the GUI code generation considers the hypothesis discussed above. The representation of the HIS families in this work is based on Ecore metamodels that define architectural constructs for properties, components and architectural styles (as discussed before). The strategy consists in the definition of a basic architectural style to describe a property family related to GUI. These properties enrich components of specific HIS families. These families are also defined as architectural styles that derive from the basic architectural style mentioned above. The properties of the basic architectural style allow to associate the clinical data types (such as observation and instruction) with GUI type elements (such as form and visualization), and these associations are defined in a particular way for each HIS family.

Due to space constraints, the Ecore metamodels and relationships mentioned above are illustrated in Figure 3 in a higher level of abstraction.[1] As shown in Figure 3, all metamodels required by the M2M component were defined as instances of the "`Ecore`" metametamodel. The proposed relationships between models and metamodels are needed to the development and execution of the transformation rules defined in ATL ("`ATL Rules`"). In this relationship, different HIS families ("`HIS Architecture Family`") can use the same property family to enrich architectural components with GUI properties ("`GUI Clinical Property Family`"). The ATL rules get as input the desired clinical data model and the intended HIS architectural model. When the rules are executed, the component of M2M transformations generates the GUI abstract models for clinical data ("`GUI Clinical Data Model`").

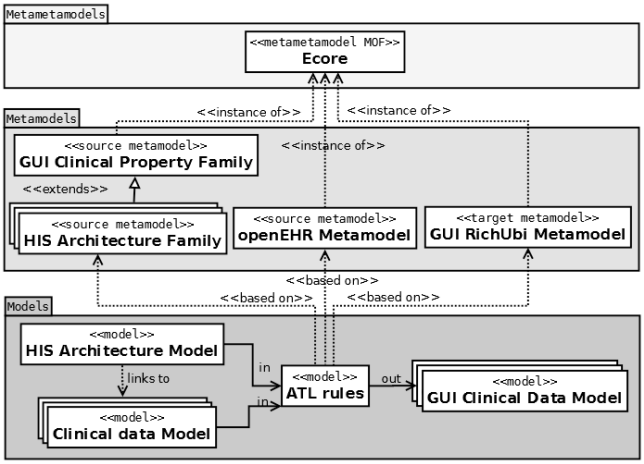

Fig. 3. Relationship between models and metamodel model transformation rules proposed.

[1] These metamodels are available at: http://martin.lncc.br/main-software

3.2 Transformation rule repository

The definition of the transformation rules in the rules repository is fundamental to the component of M2M transformation processes. These rules were structured according to a source metamodel hierarchy, and are automatically executed in sequence and iteratively according the input models allowing rule reuse. Again due to space constraints, the proposed rules are illustrated in Figure 4 in a higher level of abstraction by a UML (Unified Modeling Language) class diagram: the class names represent the rules names; classes attributes represent the "from" and "to" ATL keywords, which indicate the source and target from metadata, respectively. We employ the Composite design pattern [13] for the definition of such rules. Rules inheritance is illustrated by the stereotypes "`<<leaf rule>>`", "`<<composite rule>>`" and "`<<component rule>>`", besides ATL helpers, illustrated in the figure as "`<<helper>>`".[2] The proposed rules were structured in three main groups, which are described below.

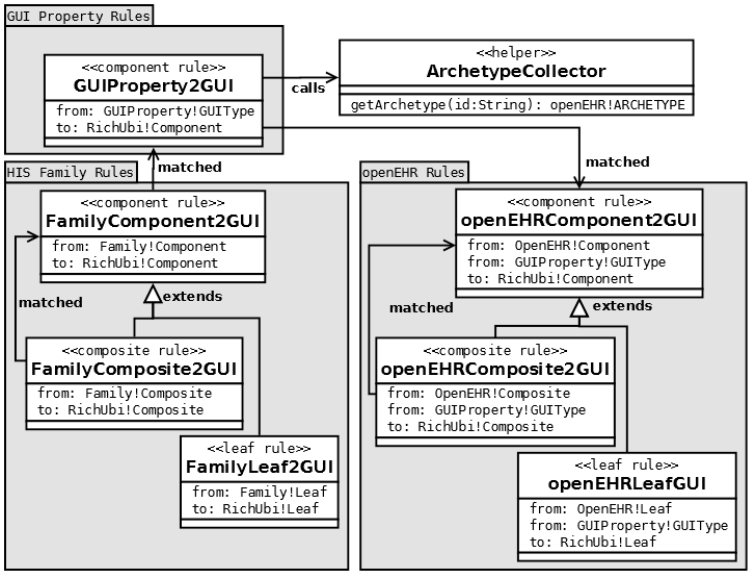

Fig. 4. Transformation rules groups: architectural family metamodel to GUI metamodel; and clinical data metamodel to GUI metamodel.

[2] These rules are available at: http://martin.lncc.br/main-software

The first group of rules ("`HIS Family Rules`") generates the basic structure of GUI clinical data models. This basic structure can be GUI metadata such as "portal", "document", "tab", "div" among others. As illustrated in Figure 4, the rules can be defined to reproduce the HIS metamodel hierarchy, using, for example, the Composite design pattern. This group of rules was developed by standard rules to process each HIS family components with GUI properties. It was considered in these transformation rules only the transformations of components from families of HIS onto web portals using the Portal metadata of the RichUbi metamodel. The complete definition of transformation rules for fully-fledged architectural style descriptions is out of the scope of this work.

The second group of rules ("`GUI Property Rules`") was defined by standard rules and generates content regions for form elements and visualization elements according to GUI properties. For example, the GUI properties of "view observation" and "view instruction" modeled in HIS family components are transformed in "view elements", such as the "div" RichUbi metadata, while "form observation" and "form instruction" lead to form elements, such as "form" RichUbi metadata. The more abstract rule in this group uses the helpers "`Archetype Collector`" and, using parameters, sends the type of GUI property to the third group of rules "`openEHR rules`", responsible for the clinical data model transformations described ahead. It is important to highlight the central role of this second group of rules in the strategy adopted for this work. This second set of rules acts as a binding between the first and third set of rules, allowing these groups to evolve independently. Thus, new rules for new families of HIS (first group) can be created without the need for creating new rules for the transformation of clinical data models (third group), resulting in a better reuse of the latter.

The third group of rules ("`openEHR rules`") is responsible for transformations of clinical data elements according to GUI properties in order to reproduce form elements (such as "list box" and "text area" RichUbi metadata) or visualization elements (such as "span" RichUbi metadata). These rules were also developed considering the Composite design pattern to preserve the hierarchical information from the clinical data models. This group of rules was developed by lazy rules to process the clinical data model defined in the GUI properties many times. It is important to note that the last two groups have only one *from* metadata source, whereas the third has two *from* metadata sources. This technique allows the development of distinct rules for each combination between GUI properties and clinical data model elements, to generate distinct GUI clinical data models.

4 Proof Of Concept

This section presents a basic prototyping of the proposed system to illustrate its technical feasibility. This proof-of-concept may not yield the same results in controlled experiments, but can provide a deeper understanding of the code generation through the model transformations proposed herein. Thus, we present a usage scenario for automatic generation of GUI code for HIS from the synthesis between the desired clinical data models (*open*EHR archetypes) and the intended HIS architectural model. In this practical application we used the HIS families of prehospital emergency and epidemiological surveillance discussed earlier.

The basic GUI property family, defined as an Ecore metamodel, was developed to enrich HIS families of prehospital emergency and epidemiological surveillance, also described on Ecore metamodels. We only considered architectural components from these families. By using the proposed IMT system, it was possible to generate different GUI clinical data codes from clinical data models considering architectural information.

To illustrate some results in this proof-of-concept, we present the GUI code generation for the prehospital emergency system family (see Figure 5 at “Emergency Family” package). In this family it is possible to construct the system for emergency care of heart attack victims described in [4]. Considering this family, the “paramedic” and “specialist” components are restricted to the observation clinical data type, such as the required clinical data models "Blood Pressure" and “Body Weight”, respectively, with "form elements” and "visualization elements” GUI properties. The leftmost part of Figure 6 shows a summary of the "Blood Pressure" and “Body Weight” clinical data models (for the “paramedic” component only) as a result of the transformation rules.

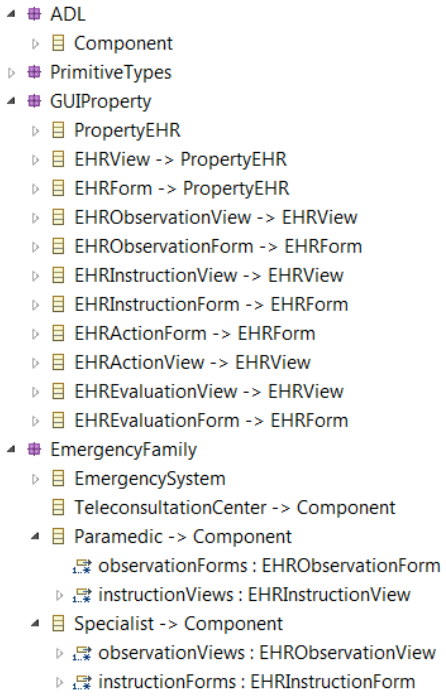

Fig. 5. Ecore Definition of a HIS family considering GUI properties and ADL specification.

Theses models are transformed, by the M2C processes of the proposed system, in form elements for the “paramedic” component, and visualization elements for the

"specialist" component. This archetype is transformed into GUI clinical data models for each component regarding the archetype information hierarchy, besides the rest of the defined archetypes, such as "Body Weight". From these models, the M2C component transforms this abstract model into code, as illustrated in the rightmost part of Figure 6 (HTML format).

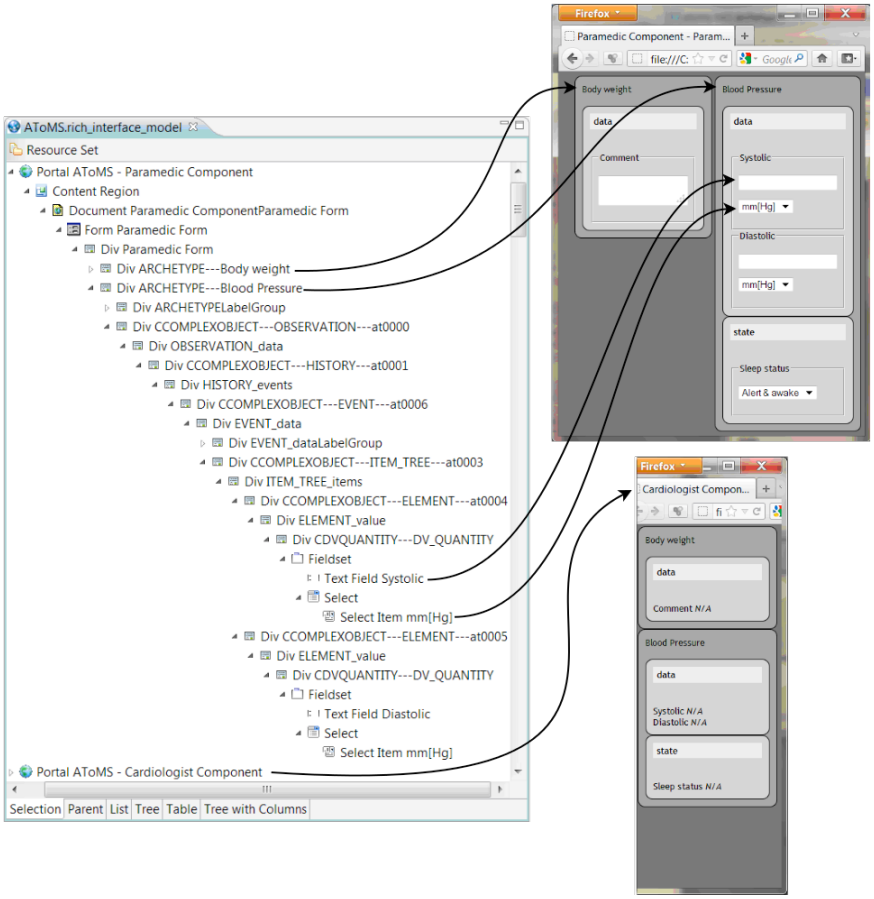

Fig. 6. GUI Clinical Data Model and some of its correspondences with HTML code.

5 Related work

Related work found in [15], [16], [17] and [18] explores, from different perspectives, the GUI code generation for HIS using clinical data models based on *open*EHR specifications. However, unlike our work, these approaches do not apply metamodeling for transformation rules in a formal way. This lack of formalization can

compromise software reuse in all software development phases. The work in [19] proposes an approach for GUI code generation for information systems as a whole. This approach uses metamodeling and transformation rules. However, the approach is not specific to HIS and does not use a common metametamodel, which decreases the interoperability and software reuse. The work in [20] presents an approach to full HIS code generation based on MDD techniques applied to *open*EHR [5]. This approach allows a high level of interoperability. However, unlike our work, it does not explore distinct HIS families, each of which needs different treatment for clinical data models, thus rendering a costly model from the point of view of transformation rules definition.

6 Conclusion

Health Information Systems (HIS) still have some inhibiting factors of a technological nature, that may hamper [23]: (a) interoperability between different HIS (such as hospitals, clinics, laboratories, etc.), (b) effort reduction of development of HIS, and (c) readability in clinical records (which may result in medical errors). Given these factors, this work uses consolidated healthcare data model specifications and formal techniques for automatic generation of code with domain information. This work allows partially achieving the benefits above through automatic GUI code generation, considering formal architectural models associated with GUI properties and clinical data models. MDD techniques offer benefits to mitigate known technological problems in HIS development. Model transformation processes are crucial in these techniques. These processes allow software reuse, such as code generation with domain concepts. For the code generation development in a high level of abstraction, the Ecore metametamodel provides a development environment that ensures interoperability and may decrease the effort of software developers. Clinical data models allow a better articulation between domain experts and developers, which makes the GUI project an important element for HIS development. However, we argue that such models are not sufficient to code generation for HIS. The present work proposes a system for the synthesis of GUI for HIS using clinical data models, based on *open*EHR specifications, and associated with architectural descriptions inspired by constructs present in the Acme ADL.

The work presented herein employs MDD techniques such as metamodeling and formally specified transformation rules, considering the Ecore metametamodel and design patterns. These techniques allow software reuse and interoperability. We also presented a strategy where clinical data models are used for GUI code generation for HIS without losing domain concepts. This work explores the fact that different HIS families requires different treatment for the relevant clinical data on different components of HIS families, which can only be done when clinical data are enriched with architectural information, as proposed in this work. We also presented a strategy for defining code generation for distinct HIS families that could generate different GUI codes from clinical data models considering architectural information in system families such as prehospital emergency and epidemiological surveillance.

As future work, we are investigating how to extend the proposed system to facilitate code generation using other architectural constructs such as connectors, ports, constraints, rules, attachments, bindings, among others, so as to generate other parts of HIS, such as communication modules and database schemas.

Acknowledgments

This work is conducted with the support of the FAPEMIG research agency, and also the CNPq research agency through the National Institute of Science and Technology in Medicine Assisted by Scientific Computing (INCT-MACC).

References

1. T. Stahl e M. Völter. Model-Driven Software Development: Technology, Engineering, Management. Wiley, Chichester, UK, 2006.
2. R. Haux. Medical informatics: Past, present, future. I. J. Medical Informatics, 79(9):599-610, 2010.
3. J. R. Vest. More than just a question of technology: factors related to hospitals? adoption and implementation of health information exchange. International Journal of Medical Informatics, 79(12):797-806, 2010.
4. B. S. P. M. Correa, B. Gonçalves, I. M. Teixeira, A. T. A. Gomes, e A. Ziviani. AToMS: a ubiquitous teleconsultation system for supporting ami patients with prehospital thrombolysis. Int. J. Telemedicine Appl. 2011.
5. T. Beale. Archetypes: Constraint-based domain models for future-proof information systems. OpenEHR Standard document, 2002.
6. M. Eichelberg, T. Aden, J. Riesmeier, A. Dogac, e G. B. Laleci. A survey and analysis of electronic healthcare record standards. ACM Comput. Surv. 2005.
7. K. Czarnecki e S. Helsen. Feature-based survey of model transformation approaches. IBM Systems Journal, 2006.
8. A. T. A. Gomes, A. Ziviani, B. S. P. M. Correa, I. M. Teixeira, e V. M. Moreira. SPLiCE: a software product line for healthcare. In Proceedings of the 2nd ACM SIGHIT International Health Informatics Symposium, IHI '12, pages 721-726, 2012. ACM.
9. Braga, R. M.; Cavalini, L. T.; Cirilo, C. E.; Cook, T. W.; Correia, B. S. P. M.; Freire, S. M. ; Gomes, A. T. A. ; Moreira, V. M.; Menezes, A.; Moraes, J. L. C.; Prado, A. F.; Souza, W. L.; Teixeira, I. M. ; Ziviani, A. . Model-Driven Development of Healthcare Applications. In: Raúl A. Feijóo; Artur Ziviani; Pablo J. Blanco. (Org.). Scientific Computing Applied to Medicine and Healthcare. Scientific Computing Applied to Medicine and Healthcare. 1ed. 2012, v. , p. 315-354.
10. Garlan, David, Robert T. Monroe, and David Wile. "Acme: Architectural description of component-based systems." *Foundations of component-based systems* 68 (2000): 47-68.
11. C. E. Cirilo, A. F. Prado, W. L. D. Souza, e L. A. M. Zai-na. Model driven RichUbi: a model driven process for building rich interfaces of context-sensitive ubiquitous applications. Proceedings of the 28th, 2010.
12. D. H. Job, A. T. A. Gomes, e A. Ziviani. Health Systems for Syndromic and Epidemiological Surveillance. In: Joel Rodrigues; Isabel de la Torre Díez; Beatriz Sainz de Aba-jo.

(Org.). Telemedicine and E-Health Services, Policies and Applications: Advancements and Developments. IGI Global, 2011, p. 245-262.
13. Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional, 1994.
14. F. Jouault, F. Allilaire, J. Bézivin, e I. Kurtev. ATL: A model transformation tool. Sci. Comput. Program., 72(1-2):31-39, June 2008.
15. T. Schuler, S. Garde, S. Heard, e T. Beale. Towards automatically generating graphical user interfaces from openehr archetypes. In A. Hasman, R. Haux, J. van der Lei, E. D. Clercq, and F. H. R. France, editors, MIE, volume 124 of Studies in Health Technology and Informatics, pages 221-226. IOS Press, 2006.
16. F. B. Nardon, T. França, e H. Naves. Construção de apli-cações em saúde baseadas em arquétipos. Acta Scientia-rum Biological Sciences, 28(4), 2007.
17. H. Van Der Linden, T. Austin, e J. Talmon. Generic screen representations for future-proof systems, is it possible? there is more to a gui than meets the eye. Computer Methods and Programs in Biomedicine, 2009.
18. K. Atalag e H. Y. Yang. From openEHR Domain Models to Advanced User Interfaces: A Case Study in Endoscopy. Health Informatics New Zealand Conference, 2010.
19. S. L. da Costa. Uma Abordagem Baseada em Modelos para Construção Automática de Interfaces de Usuário para Sistemas de Informação. Dissertação de Mestrado. Goiânia, UFG, 2011.(in Portuguese)
20. M. Menárguez-Tortosa, C. Martínez-Costa, e J. T. Fer-nández-Breis. A generative tool for building health applications driven by iso 13606 archetypes. J Med Syst, 2011.
21. OMG (2012). Object Management Group. Available at: http://www.omg.org.
22. Eclipse Foundation (2012). Eclipse Foundation. Available at: http://www.eclipse.org.
23. Cantrill, S. V. Computers in patient care: The promise and the challenge. Commun. ACM, v.53, p. 42_47, 2010
24. CKM. Clinical Knowledge Manager. 2012. Available at: http://openehr.org/knowledge/.
25. WHO. World Health Organization. 2012. Available at: http://www.who.int/en/.
26. TEIXEIRA, I. M.; BRAGA, R. M. ; Gomes, A. T. A. . Síntese Automática de Interfaces Gráficas de Usuário para Sistemas de Informação em Saúde. In: III Brazilian Workshop on Model-Driven Software Development (http://web.inf.ufpr.br/dsdm2012/), 2012, Natal, RN. Porto Alegre, RS - Brasil: SBC
27. Paul C. Clements. 1996. A Survey of Architecture Description Languages. In *Proceedings of the 8th International Workshop on Software Specification and Design* (IWSSD '96). IEEE Computer Society, Washington, DC, USA.
28. Blobel, B., Pharow, P.: Analysis and evaluation of EHR approaches. Methods Inf Med 48(2), 162–9 (2009)
29. NHS. National Health Service Media Centre: Dismantling the nhs national programme for it (2011)
30. Beeler, G.: HL7 Version 3—An object-oriented methodology for collaborative standards development. International Journal of Medical Informatics 48(1-3), 151–161 (Feb 1998).
31. Cavalini, L.T., Cook, T.W.: Health informatics: The relevance of open source and multilevel modeling. In: Hissam, S.A., Russo, B., de Mendonça Neto, M.G., Kon, F. (eds.) OSS. IFIP Publications, vol. 365, pp. 338–347. Springer (2011)