A Flexible Metamodelling Approach for Healthcare Systems

Fazle Rabbi[1,2] , Yngve Lamo[1] , Wendy MacCaull[3]

[1] Bergen University College, Bergen, Norway
Fazle.Rabbi@hib.no, Yngve.Lamo@hib.no
[2] University of Oslo, Oslo, Norway
[3] St. Francis Xavier University, Antigonish, Canada
wmaccaul@stfx.ca

Abstract. Model driven software engineering (MDSE) is an emerging methodology for software development, targeting productivity, flexibility and reliability of systems; metamodelling is at the core of most MDSE approaches. Due to their complexity and plethora of requirements placed upon them, healthcare systems so far have not been adequately modeled; as a result the software developed for them suffers from high development costs and lack of flexibility, and its reliability is at risk. Here we propose a metamodelling approach that captures the complexity of these systems by using a metamodelling hierarchy, built from five metamodels, one each for user access modelling, health process modelling, process monitoring, user interface modelling and modelling of the data sources. These metamodels are coordinated with morphisms. Such a hierarchy allows us to adequately reflect the behavior and complexities of systems and how they interact with different stakeholders. We give details of some of the metamodels and present some suggestions for some different interfaces intended for two different users: the clinicians and the patients.

1 Introduction

Rising costs, ageing populations and increased expectations are making the current healthcare systems in the developed world unsustainable. Numerous studies support this claim, for instance in the US, if the trends of the last 20 years continue, health care spending will eat up the entire GDP within the next generation, and health care spending will eat up the federal government's budget even sooner [11]. Information technology has the potential to support healthcare but its application to support the continuum of care has not nearly reached its full

In: E.A.A. Jaatun, E. Brooks, K. Berntsen, H. Gilstad, M. G. Jaatun (eds.): Proceedings of the 2nd European Workshop on Practical Aspects of Health Informatics (PAHI 2014), Trondheim, Norway, 19-MAY-2014, published at http://ceur-ws.org

potential. Barriers include the proliferation of systems even within one hospital (which often do not support interoperability) [4]; the fact that systems must be highly customized to adequately serve local situations (usually a time consuming, and error prone process); the fact that frequently adaptations to deal with updates in medications, protocols and management strategies, etc., are necessary; the fact that software engineering itself for such safety critical systems as healthcare needs new strategies to ensure that systems behave correctly in every possible scenario [7]; and the fact that many healthcare processes (e.g., cancer care and palliative care) require a team of caregivers, involving many clinicians, therapists and family members all with different needs from information technology and with different, and sometimes limited, ability to handle complex technologies. The very nature of healthcare processes differs from typical processes managed by workflow engines as they involve many exceptional situations, and the sequence of tasks described by a guideline may need to be altered, at the implementation level, in order to meet actual user needs, while maintaining guideline intentions as much as possible [15]. The active participation of the patient (and his/her family) in the management of his/her own health is becoming a critical issue as the cost of chronic diseases is quickly outpacing the resources that can be directed to healthcare.

Model driven software engineering (MDSE) is an emerging and promising methodology for software development, targeting challenges in software engineering relating to productivity, flexibility and reliability of systems. The construction of various kinds of models (e.g., blueprints, mockups etc.) is a well-known approach in the more traditional engineering fields; these models are used as artifacts to enable engineers to describe designs and validate whether a proposed design has desired qualitative and quantitative properties. Metamodelling is at the core of MDSE approaches. Here we propose that a multi metamodelling approach is the appropriate methodology for designing healthcare systems. The use of multiple metamodels for designing different aspects of a system facilitates abstraction and require less coupling among the models; this gives us flexibility as it permits the independent remodelling of parts of the system. This paper is a preliminary look at 5 aspects of healthcare systems, and the necessary coordinations among them. Due to lack of space we focus our discussion on the access control, monitoring, user access and usability aspects.

Many different MDSE technologies automatically generate code from models [10] [13]: these technologies are particularly suited for specifying the structural aspects of software systems generally, whereas the actual behavior is programmed manually. Some technologies for behavioural modelling in MDSE exist ([2], [1]), but current approaches are often at a low level of abstraction and lack domain concepts for specifying behavior [12].

A collaborative group of researchers in Norway and Canada have been working on various issues relating to these problems. We proposed a formal approach to workflow modelling in [21] based on the Diagram Predicate Framework (DPF) [18] which provides a formalism of (meta) modelling and model transformations based on category theory and graph transformations ([6], [8], [9]).

We [22] extended the formal foundation of DPF to define (static) semantics for timed and compensable workflow models and defined the dynamic semantics of models by a transition system where the states are instances and transitions are applications of transformation rules. We developed a domain specific language to expedite workflow system development [17] and began the development of a userfriendly interface to allow the health practitioner to determine the correctness of behavioral properties of a healthcare workflow protocol [20]. We believe that model driven engineering has the potential to develop complex systems formally and can be used in healthcare domain.

2 Modelling Healthcare Processes

Clinicians generally follow clinical guidelines to manage specific diseases. A clinical guideline is a description of processes, treatment procedures, appropriate medications, etc., to manage a particular disease. Interested readers may refer to the guideline for Hypertension Management [3]. Clinical guidelines may be used as basis for the formalization of healthcare processes as workflows. A workflow model consists of tasks and the specification of the order in which they should be executed. While performing a task, a user provides data to the system; typically, this is filling out a standardized web-based form, a mobile app, or through the use of some healthcare technology which provides automatic integration with the appropriate healthcare datasource.

For modeling healthcare processes, we have used the DERF workflow language [19,21] which allows one to graphically model a workflow using the DPF workbench [14]. Fig. 1 (a) (which is one level of the metamodelling hierarchy of the DPF framework) shows the overall model of the Hypertension Management Workflow [3]. We briefly review this workflow as we will refer to features of it throughout this paper.

Remark, there are some composite tasks such as 'Visit1', 'CBPM' (Clinical Blood Pressure Measurement) in the workflow model; those are abstractions of subworkflows. The subworkflow for 'Visit1' composite task is shown in Fig. 1(b).

Initially a patient's blood pressure (BP) is measured at the 'Initial BP' Task, which may cue the clinical hypertension management procedures if the BP is greater than or equal to 140/90. If the initial BP is normal (<= 140/90), the workflow terminates. In Fig. 1 the patient's Hypertension is managed through investigation and treatment. The clinical procedure (i.e., 'Visit1', and other subsequent tasks in Fig. 1) starts at the doctor's office. Patients with high BP have risk of organ failures and/or other chronic illness. During the first visit at the doctor's office ('Visit1') BP is measured twice, an initial assessment is done, and an investigation is started with diagnostic tests. After 'Visit1' the workflow executes 'Ambulatory Blood Pressure Monitoring' ('ABPM') or 'Self Home Blood Pressure Monitoring ('SHBPM') if they are available and a "Clinical Blood Pressure Measurement" (CBPM) is performed. Note the overall workflow model in Fig. 1 uses the abbreviations CBPM, ABPM and SHBPM for these tasks.

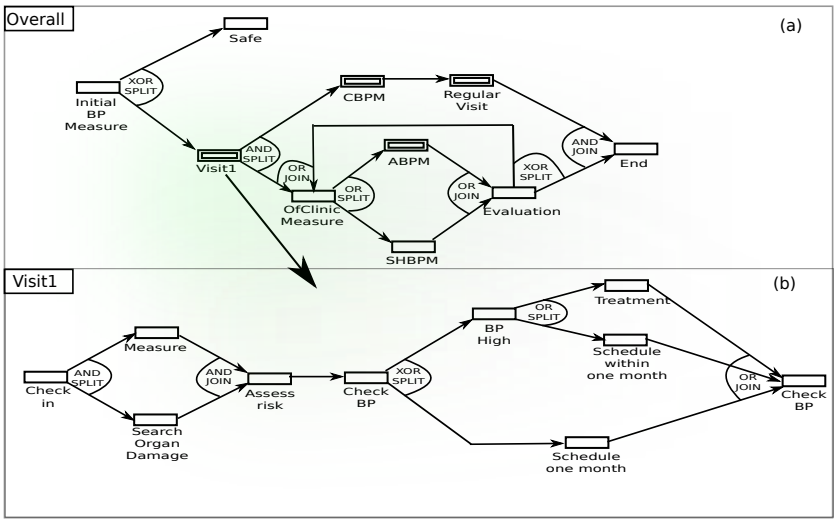

Fig. 1: Hypertension Management Workflow (Overall)

Many healthcare processes involve numerous stakeholders with different requirements. Frequently the user becomes a critical part of the healthcare workflow process whether it be the physician, a specialist or a lab technician, or the patient, in situations where management of lifestyle parameters is a critical component of the process. We are now researching a metamodelling approach to workflows which incorporates the concepts of stakeholders and process monitoring and provides userfriendly interfaces for a variety of users.

The PhD thesis of [5] promoted a metamodelling approach to the development of a framework for modelling care processes. There, Baarah presented a UML-style metamodel for the care process monitoring application that had 3 main components: a process model, a performance model and an enterprise model. The process model defines the care process in terms of states to be monitored, resources and rules that specify the transition from state to state as events are received from the enterprise model. The performance model measures how well the goals for the care process are being achieved in terms of metrics computed from the monitored states, and events for the process. Alerts are defined to flag when targets are not being met. However, no automated implementation of the metamodel was attempted, correctness of the process was investigated only through the use of test scripts, and user interface issues were not considered.

We extend the model Baarah presented in [5] to include users and user interactions, allowing us to model user interaction as part of the process. Users may interact with various tasks in the workflows, with the datasource, and with the monitoring system (users typically receive alerts from the monitoring system and acknowledge them, if required). This takes us from considering healthcare

workflows as an isolated entity to the more realistic modelling of a healthcare systems.

3 Metamodelling Healthcare Systems

Given the complexity of the information requirements in healthcare systems, we propose that separate metamodels, i.e., multi metamodels is the appropriate approach to modelling healthcare systems. In this paper we discuss metamodelling of five aspects of healthcare systems: user access modelling, health process modelling, modelling of process monitoring, user interface modelling, and modelling of the data sources. Links (directed arcs) between the metamodels are used to coordinate them (see Fig. 2), i.e., directed arcs from one metamodel to another in Fig. 2 represent the bindings between metamodels. Using separate metamodels for modeling different aspects of a system gives us the flexibility for remodelling and also makes models more readable. A user of a system may have access to some tasks, views, and data; this requirement is modeled in Fig. 2 by means of 3 morphisms called 'Task-acc', 'View-acc', and 'Data-acc'. A Task may trigger some alerts ('Trigger' arc), may be displayed by a view ('Task-UI' arc), and may read/write some data from/to the datasources ('Data-acc' arc). An alert is sent to some users ('Send' arc) and is displayed by a view ('Alert-UI' arc). The user interface of a system consist of some views; the user interface views access the datasources and displays data ('Displays' arc) in different formats.

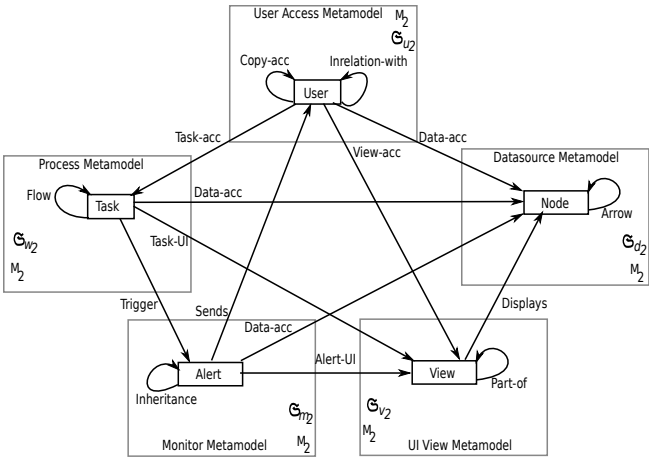

Fig. 2: Multiple metamodelling hierarchy

Note that the workflow model in Fig. 1 is typed by the 'Process metamodel'; moreover, there are some predicate constraints specifying the routing of the workflow. The user access to a DERF workflow model (in Fig. 1) is defined by

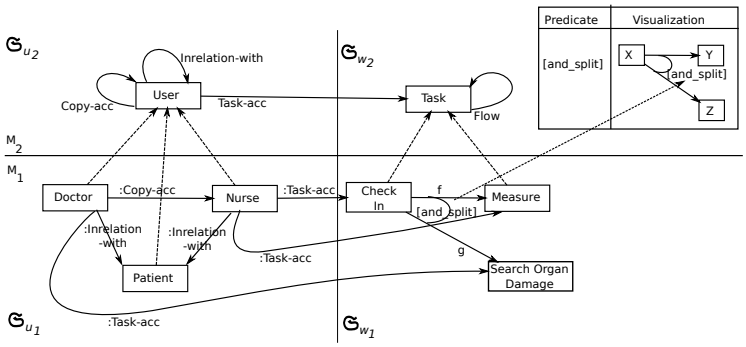

Fig. 3: Modelling and Co-ordination of User Access and Workflow Metamodels

the morphism called 'Task-acc'. Here we discuss the metamodelling hierarchy for the user model. The left hand side of Fig. 3 shows two modelling levels M_2 and M_1 of a user model where $\mathfrak{S}_{u_2}$ and $\mathfrak{S}_{u_1}$ are the specifications (respectively). The 'Copy-acc' morphism is used to copy access from one user to another. One instance of the 'Copy-acc' morphism from 'Doctor' to 'Nurse' in Fig. 3 gives Doctors all the access that Nurses have. The 'Inrelation-with' morphism is used to associate caregivers with patients. In this case, only doctors or nurses who are treating a patient can access that patient's information. This access control aspect is modeled using the 'Inrelation-with' association. The right hand side of Fig. 3 shows two modelling levels of a DERF workflow model with specifications $\mathfrak{S}_{w_2}$ and $\mathfrak{S}_{w_1}$. At level M_1 the predicate $[and\_split]$ is used to model concurrency.

To visualize the user access for a workflow model we propose a user interface in Fig. 4 where the user nodes (e.g., Doctor, Nurse, Patient) are displayed at the bottom of a workflow model. Selection of a user node from the bottom window highlights all the accessible tasks from the workflow model. In the figure, the user Patient has only access to the 'SHBPM' that has been shown in gray. If a doctor and a patient instance are both selected from the drop-down, the system highlights all the tasks that this doctor can execute for this patient.

While executing the 'SHBPM' task the patient registers his lifestyle information; in this workflow the patient is responsible for registering his lifestyle information and doctors and nurses are responsible for the rest of the workflow. Both the doctor and the patient should have access to the lifestyle information. Doctors have a user interface similar to the one the patient uses, but it has many more features (e.g., sending an e-mail to the lab for a lab test).

Fig. 5 shows a model of workflow monitor having its own metamodelling hierarchy and its association with a process metamodelling hierarchy and a user access metamodelling hierarchy. Tasks from a DERF workflow model can trigger alerts. We have two types of alerts: 'Critical Alert', and one less urgent, called 'Reminder'. The 'SHBPM' task from Fig. 1 triggers a 'Data Entry' alert if the patient forgets to enter data on some day. It also triggers a 'Excessive Weight

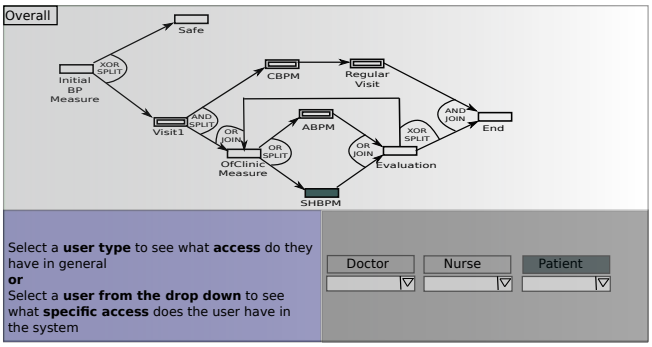

Fig. 4: Visualization of Hypertension Workflow Model and User Access Interface

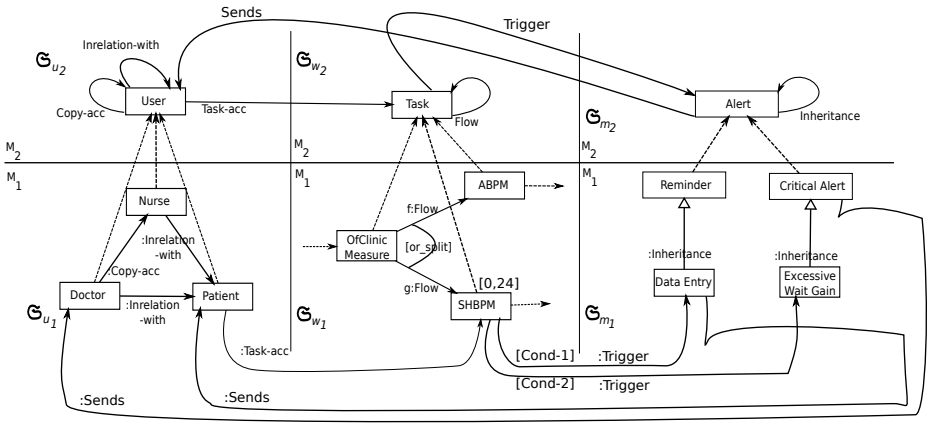

Fig. 5: Modelling and Co-ordination of User Access, Process and Monitor Metamodels

Gain' alert if the patient's weight gain exceeds 10 pounds in less than 5 days. This firing condition is encoded in the predicate $[Cond-2]$ found on the co-ordinating link between the 'SHBPM' task and the 'Data Entry' alert. Different users can receive different alerts. In this figure, only the doctors are alerted about the 'Excessive Weight Gain' to indicate that the patient is retaining excessive fluids and the doctor should consult that patient immediately. The 'Data Entry' alert is sent to the patient to inform the patient that he or she has forgotten to enter information. In a DERF workflow a task may have time constraints [22]. The task 'SHBPM' has a time constraint of '0 hour' delay and '24 hours' duration, meaning the task 'SHBPM' becomes enabled immediately after the execution of the 'OffClinic Measure' task and must be executed within 24 hours from when it became enabled. $[Cond-1]$ is a predicate that triggers the 'Data Entry' alert if 24 hours has elapsed and the 'SHBPM' task is not executed. Discussion of the

'Dataource Metamodel' and the 'UI-View Metamodel' are out of the scope of this paper.

4 User interfaces

Successful of technology depends a great deal on usability or user-friendliness of its user interface (also called "UI" or simply "interface"). A UI is the means by which a person controls a software application or hardware device. A good user interface provides a "userfriendly" experience (where userfriendly with respect to software is defined by the Merriam Webster dictionary as "easy to learn, use, understand, or deal with") allowing the user to interact with the software or hardware in a natural and intuitive way. In this section, we discuss the development of two kinds of user interfaces needed for the Management of Hypertension. (Some preliminary discussion of these interfaces may be found in [16].) The first is a user interface with various features intended for use by the clinician. The second is a user interface, which we call a "Personal Health Monitor", intended for use by the patient for self management of lifestyle attributes that cause the patient to be at risk. Our goal was to provide both the patient and the clinician with technology that can help in the management of the hypertension protocol, but not to overwhelm them with tools that were not userfriendly.

First, consider the interfaces which allow clinicians to interact with the system. Fig. 6 shows 4 windows named 'Workflow Viewer', 'Task Execution Viewer', 'Lookup Viewer', and 'NOVA Browser'. The right hand side of the 'Workflow Viewer' lists all tasks currently enabled by a workflow (e.g., Fig. 1) running underneath the hood; and therefore are available to be done. Whenever a task is executed, the task name is put on a calendar. Dates of scheduled appointments are also presented on the calendar. By default the calendar shows the 'month view' but different levels of granularity may be configured (e.g., weekly view, hourly view, etc.). A task being executed is shown in the 'Task Execution Viewer'. Inputs required from the end user are shown in branches and the end user must select a branch and assign a value to it through the 'Lookup Viewer'. The 'Lookup Viewer' helps the end user to input information either by showing relevant values from an ontology or by allowing the end user to enter information. Once a task is executed, the information are hierarchically displayed in the 'NOVA Browser'. Fig. 6 shows that the user is executing the 'Measure BP' task. Inside the 'Task Execution Viewer' window, the user provides input to execute the task. This is an alternative way of taking user input rather than using *Forms*. This view can also be configured to include a traditional 'Form view' where the user provides input in 'Form fields' (e.g., text boxes, drop down boxes, etc.). While executing tasks the user has access to historical information for this workflow instance; the intent here was to provide the user with a more userfriendly environment to concentrate on care, avoiding the need to go back through forms, either in a paper based or in an electronic format, to get information they need.

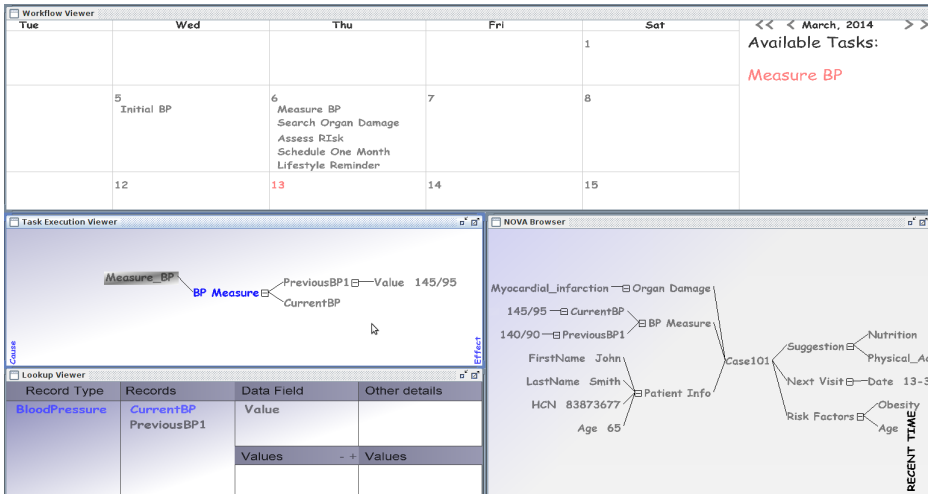

Fig. 6: Hypertension Management Workflow

The 'Lookup Viewer' at the bottom left of Fig. 6 provides options allowing the user to select or enter data. This window is connected to a database and shows only relavent data from the database. This view can also be configured to connect to an ontology and show relevant terminology from an ontology to help the user input information while executing a task. Data inserted or selected by the user in the 'Lookup Viewer' is reflected in the 'Task Execution Viewer' window. When the user is finished entering all input for a task, the task is executed and this updates 'NOVA Browser' nodes. For this system we developed different user interfaces for different user types. These are built depending on the needs and expertise of the user.

We now discuss the Personal health monitor smart phone application that gives the patient a userfriendly interface for self management of lifestyle attributes which cause the patient to be at risk. The patient can input data for lifestyle attributes such as, exercise, smoking, intake of fruits and vegetables and record such attributes as weight and blood pressure. The purpose of the application is to assist patients keeping their health record such as blood pressure record, body mass index, hours of exercise, dietary, etc. and monitor their performance with their lifestyle target that was set by the physician from 'Hypertension management workflow'. The web-based tool allows both the patient and the clinician to view summary data on lifestyle parameters between visits and provide calendar views of past activities, future appointments, etc. In (Fig. 7) we see that using the smart phone application, the patient can monitor their exercise and eating behavior.

The smart phone application allows the patient to execute the 'SHBPM' task (see 'Overall' workflow from Fig. 1) from home. The 'Personal health monitor' application interacts with the 'Hypertension management workflow'. The

Fig. 7: Personal health monitor (Diet and exercise monitoring)

integration is accomplished by the task 'SHBPM' (see Fig. 1) from the 'Hypertension management workflow'. We have developed several interfaces that give summary data to the patient and doctor by projecting patient's data into graphs and charts. Fig. 8 shows two screenshots from the smart phone application that takes blood pressure input from the patient and displays a graph of recent blood pressure measurements. Projecting different data on the same timeline may provide analytical ability to the user. The smart phone application can also fetch an appointment date from the workflow and reminds a patient about the date of the next visit.

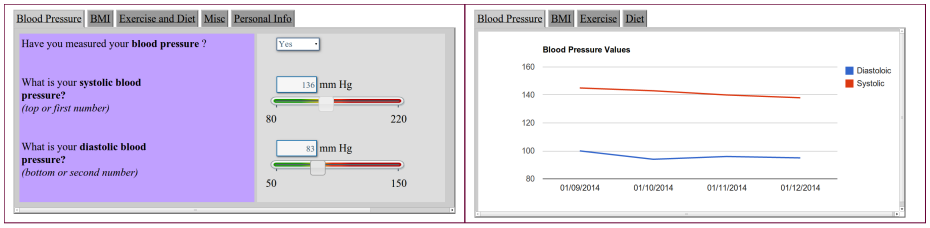

Fig. 8: Personal health monitor (Blood pressure monitoring)

Demos of the system were made to several people including a local GP who deals with many patients with chronic diseases, a nurse who is the VP of Community Services for GASHA, our local health authority, and a physiotherapist, who is the manager of the local Seniors' Wellness Program. The feedback on the interfaces for the Personal health monitor was excellent – indeed the GP is considering using some apps of this nature developed by our students in his practice. The clinician views as shown in Fig. 6 were less enthusiastically received. In general, it was felt that the clinicians would find this tool too hard to use; we plan to work with some clinicians and designers to see what can be done to develop intuitive interfaces for clinical practice guidelines for chronic diseases suitable for use by clinicians.

5 Conclusion

Modelling the flow of tasks outlined in a clinical guideline, even a complex one, is not too difficult if done in consultation with a domain expert (clinician familiar with the procedure). Our modelling tools allow us to deal with the overview first and 'zoom in' to refine tasks into sub workflows. One challenge is to recognize that guidelines are not rigidly defined processes. They involve many exceptional situations, and the sequence of tasks described by a guideline may need to be altered, during the enactment of a workflow for a particular patient, in order to meet user needs, while maintaining guideline intentions as much as possible [15]. Much of this may be dealt using "decision points" in the guideline, where several choices are available, and the choice is left to the physician (sometimes in consultation with the patient, and using background information contained in an ontology or other datasource). In other situations the physician may need to override the execution of the workflow due to a circumstance or exception not covered by the guideline. Allowing the clinician to simply skip part of the workflow (while providing a reason to explain why) is not difficult, but in general dealing with exceptions not covered in the guidelines is a real challenge to workflow modelling.

In previous work, we looked simply at modelling the flow of tasks in a healthcare process; however, conceptualizing a healthcare system as being comprised of five metamodels allows us to more realistically model these complex systems. While incorporating stakeholders and monitors in the MDSE paradigm is highly innovative, these features are essential if software systems are to automatically perform the kinds of tasks users are increasingly demanding. We believe that the metamodelling together with MDSE principles in general can be used as the main methodology in the development process of software for care processes. By separating different concerns of a system into several metamodels we get more flexibility allowing us to modify one aspect of a system described by a particular metamodel without affecting other metamodels. We remark that this is early work in our attempt to model a systems as complex as the healthcare system; we need more effort to capture the complexities of real-life systems. We have a prototype implementation for part of the system using the DPF framework, and we are working to extend it.

Acknowledgement

MacCaull would like to thank Natural Sciences and Engineering Research Council of Canada, and Lamo acknowledges support from the St. Francis Xavier Heaps Chair in Computer Science. We are greatful to the StFX undergraduate student, Miao Huang for his contributions to the Hypertension Workflow Model and to Jane Newlands, Manager of the Seniors Health Wellness Program for GASHA, a health authority in Nova Scotia for providing valuable feedback to us.

References

1. Fujaba Development Team: The Fujaba Tool Suite, http://www.fujaba.de/
2. Object Management Group: Semantics of a Foundational Subset for Executable UML Models (FUML) (February 2011), http://www.omg/spec/FUML/1.0/
3. The Chinook Primary Care Network, http://www.chinookprimarycarenetwork.ab.ca
4. Interoperability is the future (2013), `http://himss.files.cms-plus.com/FileDownloads/AxwayInteroperabilityistheFuture.pdf`
5. Baarah, A.H.: An application framework for monitoring care processes. PhD thesis, University of Ottawa, 2013
6. Barr, M., Wells, C. (eds.): Category Theory for Computing Science, 2Nd Ed. Prentice Hall International (UK) Ltd., Hertfordshire, UK, UK (1995)
7. Cleland, G., Habl, I., Medhurst, J.: Using safety cases in industry and healthcare (December 2012), `http://patientsafety.health.org.uk/sites/default/files/resources/using_safety_cases_in_industry_and_healthcare.pdf`
8. Diskin, Z., Wolter, U.: A diagrammatic logic for object-oriented visual modeling. Electr. Notes Theor. Comput. Sci. 203(6), 19–41 (2008)
9. Ehrig, H., Ehrig, K., Prange, U., Taentzer, G.: Fundamentals of Algebraic Graph Transformation. Monographs in Theoretical Computer Science. An EATCS Series, Springer (2006)
10. Fowler, M.: Domain-specific languages. Addison Wesley Signature Series, Addison Wesley (2011)
11. Hixon, T.: The U.S. does not have a debt problem ... It has a health care cost problem (September 2012), `http://www.forbes.com/sites/toddhixon/2012/02/09/the-u-s-does-not-have-a-debt-problem-it-has-a-health-care-cost-problem/`, [Online; posted 02-September-2012]
12. Kindler, E.: Model-based software engineering: The challenges of modelling behaviour. In: Proceedings of the Second International Workshop on Behaviour Modelling: Foundation and Applications. pp. 4:1–4:8. BM-FA '10, ACM, New York, NY, USA (2010)
13. Kroiss, C., Koch, N., Knapp, A.: UWE4JSF: A Model-Driven Generation Approach for Web Applications. In: Gaedke, M., Grossniklaus, M., Daz, O. (eds.) Web Engineering, Lecture Notes in Computer Science, vol. 5648, pp. 493–496. Springer Berlin Heidelberg (2009)
14. Lamo, Y., Wang, X., Mantz, F., MacCaull, W., Rutle, A.: DPF Workbench: A Diagrammatic Multi-Layer Domain Specific (Meta-)Modelling Environment. In: Lee, R. (ed.) Computer and Information Science 2012, Studies in Computational Intelligence, vol. 429, pp. 37–52. Springer Berlin Heidelberg (2012)
15. Quaglini, S., Stefanelli, M., Lanzola, G., Caporusso, V., Panzarasa, S.: Flexible Guideline-based Patient Careflow Systems. Artificial Intelligence in Medicine 22(1), 65–80 (2001)
16. Rabbi, F., MacCaull, W.: Designing User-friendly UIs for the Execution of Clinical Practice Guidelines. 27th International Symposium on Computer Based Medical Systems, 2014, In press
17. Rabbi, F., MacCaull, W.: $T_{\square}$: A Domain Specific Language for Rapid Workflow Development. In: MODELS 2012. pp. 36–52. Lecture Notes in Computer Science, Springer (2012)
18. Rutle, A.: Diagram Predicate Framework: A Formal Approach to MDE. Ph.D. thesis, Department of Informatics, University of Bergen, Norway (2010)

19. Rutle, A., MacCaull, W., Wang, H., Lamo, Y.: A Metamodelling Approach to Behavioural Modelling. In: Proceedings of BM-FA 2012: 4^{th} Workshop on Behavioural Modelling: Foundations and Applications. pp. 5:1–5:10. ACM (2012)
20. Rutle, A., Rabbi, F., MacCaull, W., Lamo, Y.: A user-friendly tool for model checking healthcare workflows. Procedia Computer Science 21(0), 317 – 326 (2013), the 4th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN-2013) and the 3rd International Conference on Current and Future Trends of Information and Communication Technologies in Healthcare (ICTH)
21. Rutle, A., Wang, H., MacCaull, W.: A formal diagrammatic approach to compensable workflow modelling. In: Weber, J., Perseil, I. (eds.) FHIES. Lecture Notes in Computer Science, vol. 7789, pp. 194–212. Springer (2012)
22. Wang, H., Rutle, A., MacCaull, W.: A Formal Diagrammatic Approach to Timed Workflow Modelling. In: Proceedings of TASE 2012: 6^{th} International Conference on Theoretical Aspects of Software Engineering. vol. 0, pp. 167–174. IEEE Computer Society (2012)