A Model-Driven Approach to Create and Maintain an Executable Transferal Management Platform

Emanuele Laurenzi
Institute for Information & Process Management, University of Applied Sciences St. Gallen, St. Gallen, Switzerland

1 STAGE OF THE RESEARCH

My work falls within the eHealth application domain and it is embedded into the just started research project Patient Radar. The project is funded by the Swiss Commission for Technology and Innovation (CTI) under the lead of the Institute for Information and Process Management (IPM-FHS) - University of Applied Sciences St. Gallen. Two regional hospitals and a Swiss national technology provider are involved in the project in which they contribute with their expertise.

The Patient Radar project wants to facilitate intersectoral collaboration within the inpatient sector, also called “transferal management”, i.e. between acute hospitals and rehabilitation clinics in Switzerland.

My research aims at supporting and optimizing such a collaboration by setting up a framework which adopts a model-driven approach to enable the creation and maintenance of a transferal management platform. The model-driven approach makes the platform highly configurable to accommodate new clinical pathways and be easily extendable to include additional functions to meet future needs. All domain-specific aspects are described declaratively in application models. Hence, domain experts will be able to create/use/manage application models with no required programming skills. To provide an executable platform, models are first specified in the formal semantics description logics and then their elements are mapped to corresponding elements in an application framework. In this way, we will ensure that executable code can be derived from all application models. Additionally, the transferal management platform includes reference models from which domain experts can easily create and adapt application models.

Research questions are established and along my work, these will be refined more and more.

2 OUTLINE OF OBJECTIVES

Developing a transferal management platform includes the challenge of combining the clinical pathways of several hospitals and several rehabilitation clinics into one coherent treatment process. The project intends to address this challenge by creating a **transferal management platform** that will serve as a hub for post-acute care, which enables and supports the interaction between acute hospitals and rehabilitation clinics as well as further actors such as nursing facilities, family doctors and health insurances for granting cost reimbursement. While a patient is still in the acute hospital, information such as concerning the patient’s health status, required rehabilitation treatment, planned transfer date need to be available on the platform so that rehabilitation clinics can apply for those patients and plan ahead their resources and determine early on therapies which are specifically adapted to the expected patient. In particular, the transferal management platform should

- know about clinical pathway (i.e. the patient treatment process) of acute hospitals and partly of the rehabilitation clinics;
- monitor the progress of a patient along the clinical pathway according to predefined medical indicators;
- give the rehabilitation clinics access to the progress of the patients as well as further rehabilitation-specific parameters such as age, weight, and diagnosis; all other personal data are anonymized until the patient is transferred.

The transferal management platform will have to support wide variety of clinical pathways for all kinds of acute hospitals and rehabilitation clinics. Therefore, the platform must be highly configurable to accommodate new pathways, and it must permit that a given pathway slightly differs between hospitals. Moreover, the platform should be easily extendable to include additional functions to meet future needs. As a consequence, we decided to adopt a **model-driven approach** where all domain-specific aspects are described declaratively in an application model. The elements in the **application model** will be mapped to corresponding elements in an **application framework** to obtain the executable transferal management platform.

In order to guide and support building and maintaining the application model we introduce **reference models** that provide blueprints for common application scenarios, such as typical pathways with the necessary patient data, generic models of the associated administration processes as well as role models for typical users of the platform with their rights.

While the use of reference models helps create specific application models it is very difficult to constrain the adaptations to a reference model so that only models are created that can be mapped to the underlying application framework, i.e. can be made executable. Since our primary purpose is to obtain executable platforms from the application models losing executability must be avoided.

We therefore decided to ground the reference model in a **domain-specific language** (DSL) (Jouault F. und Bézivin 2006) (Mernik et al. 2005) (Ranabahu et al. 2012) (van Deursen et al. 2000) instead of a general-purpose modelling language such as UML or BPMN. The reference models and all application models are created using representation constructs from the DSL. Since the definitions of the mapping from model elements to elements of the application framework are tied to the DSL contructs any model expressed in the DSL can be mapped to the framework and thus be made executable.

3 RESEARCH PROBLEM

As outlined in the previous two sub-sections we adopt a model-driven approach where the domain-specific aspects of the transferal management platform are specified by an application model from which code fragments for the actual transferal management platform are generated.

Without the framework given by both, the reference model and application model, the transferal management platform would just consist of an ordinary model that can be modified in any possible way, including desired ways as well as undesired ones. In particular, our framework should

- enable domain experts to adapt and extend the platform to future needs without the help of computer specialists; this is done by adapting the platform through modelling instead of programming;
- ensure that changes to the model do not destroy the code generation property.

The following three sub-sections describe our proposed approach. The research questions are defined subsequently.

3.1 Basic Approach

We embed our approach into a meta modelling framework (Karagiannis und Kühn) (Laarman und Kurtev 2010) where the application models reside on level 1 and the DSL is defined as a meta model on level 2, thus providing the modelling language in which the level 1 models are represented (cf. Figure 1).

The reference models also reside on level 1 and serve as blueprints for the application models. An application model is derived by modifying and adapting a reference model but not by instantiating it. That is why reference models and application models are on the same level. Level 3 provides the language constructs needed to define the DSL.

By using the two-tier approach shown in Figure 1 we aim for two kinds of extensibility:

- On level 1 new application models are created and existing ones adapted to suit new needs, e.g. to accommodate completely new or slightly adapted clinical pathways. This is the task of a *domain expert* who is guided by the reference models and uses the domain-specific constructs provided by the DSL. We especially aim at enabling domain experts for this task who do not have to be modelling experts by providing a user interface which translates modelling into an interactive visual paradigm.
- New requirements that cannot be met by adapting an existing model on level 1 are taken care of on level 2 by extending the DSL to provide the additional expressiveness needed on level 1. This is the task of a *modelling expert*.

Executability of the models on level 1 will be provided by mappings from the DSL constructs to corresponding elements in the application framework. Via this approach we will ensure that all models on level 1 can in fact be properly mapped to the application framework. An alternative approach without a DSL would have used a general purpose language, such as UML, and extended or refined its semantics by adding constraints that restrict the range of possible changes to the model (Lodderstedt et al. 2002).

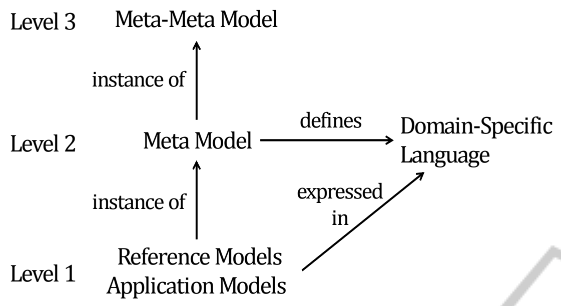

Figure 1: Our approach in a meta modelling framework.

Such an approach, however, has the disadvantage that it increases the complexity of the modellers' task because they have then to know all these constraints and take them into account, or they are hindered by the system to make certain modifications to a model when they would violate a constraint. Our approach using a DSL promises to be much easier for the modellers.

3.2 The Approach in Detail

A first idea for designing the DSL might be to use a traditional process modelling language as its basis since a description of the clinical pathways are at the center of the transferal management platform. In our case, however, we do not need to describe the pathways in detail but only certain aspects of them, mainly from a healing progress point of view and not from a medical treatment perspective. The DSL must allow to represent what the main healing and rehabilitation phases are and which conditions must be met (called *gateways*) to get to the next phase. For this we do not need a full-fledged process modelling language. Instead, our DSL only includes some basic process modelling elements, thus reducing model complexity, increasing modelling productivity (Ulrich F. 2010) and enabling a simpler mapping to executable elements in an application framework.

The backbone of our DSL is the domain-specific terminology which includes the object types relevant for our domain as *meta classes*. Currently we have the following meta classes: "ClinicalPathway" for the patient treatment process, which consists of activities. An activity leads to gateways that stand for certain conditions being tested. Gateways with fulfilled conditions enable further activities (cf. Figure 2). Activities are associated with patients and can be specialized into various subclasses such as treatment activities and administrative activities.

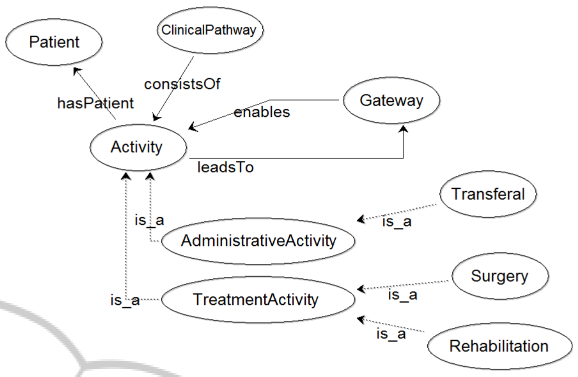

Figure 2: Part of the domain-specific language.

All the *classes* used in an **application model** are instances of a *meta class* in the DSL. We have already created an application model for knee replacement surgery. It contains a sequence of activities with gateways between two activities. Most important are the gateways since they are the indicators how soon a patient can be released to a rehabilitation clinic. Each activity has two duration attributes: "typical duration" says how long the activity usually lasts, "actual duration" says how long the activity has already lasted. From these attributes the transferal management platform can derive the expected transferal date. Each gateway has an attribute which says if the condition the gateway stands for has been fulfilled, not been fulfilled or not yet been tested.

Besides their duration the activities themselves are often not of interest and are then represented by the class "SomePostSurgeryTreatment". In cases of complications a second surgery or even a stay in an intensive care unit might become necessary. When covering such cases in the model it might be necessary to represent these treatment activities explicitly so that a rehabilitation clinic can see the reason why a patient will be released later than originally expected.

Reference models are on the same level in the meta-model hierarchy as application models (cf. Figure 1) so that their classes are instances of meta classes of the DSL as well. The difference to an application model is that a reference model is more abstract and thus can serve as a blueprint for many application models. We have also created an example of a reference model for knee surgery, which is more abstract than the application model (above described) and only includes the absolute necessary activities and gateways. There might also be a reference model for surgery, which would be even more generic. In fact, many different reference models might be created over time, some of them specializing or generalizing already existing ones. Thus, for example, even an application model might be regarded as a reference model, which would then still have to be refined to reflect the specific procedures of different hospitals. Summarising, if a model is an application model or a reference model depends on its degree of generality and on the purpose it needs to fulfil and is thus much a matter of viewpoint.

3.3 Mapping Application Models to an Application Framework

The application models are not just created to describe an application domain but primarily to be translated into some specific *behaviour* in the Patient Radar transferal management platform. To achieve this, application models will be mapped to corresponding elements of the application framework that is the basis for the transferal management platform (cf. the similar approach in (Reimer et al. 2008)). In our project we use the application framework Vivates by the Swiss Post (http://www.vivates.ch/). The mapping rules will be defined in the DSL where they are attached to each meta class.

For example, certain classes such as "SurgeryPatient" will be mapped to corresponding **object types** in Vivates. The objects belonging to these object types reside in the runtime environment of the application framework and are conceptually instances of the corresponding classes in the application model.

Other classes, such as "Gateway" will be mapped to **states** in Vivates. A sequence of gateways (through intermediate activities) will be mapped to **rules** which reflect that order by specifying which gateways have to be reached before another gateway can be reached. In the case of parallel gateways a rule is generated that requires both gateways to be fulfilled in order to enable the subsequent activity.

The duration attributes of each activity class will be used in a **web service** of Vivates to compute the expected time until transferal to the rehabilitation clinics is possible and makes this value available on the transferal management platform.

3.4 Research Questions

The development of reference/application models by adopting the model-driven approach pose the following research questions:

- How can DSL constructs be mapped to corresponding elements of the application framework to enable and preserve executability in the platform?
- How much variability can the application model and/or the reference model allow without losing its executability property for the platform?
- How can an application model and/or a reference model take advantage from a domain-specific modelling language?
- How do reference models and application models differentiate each other to enhance the domain expert understanding of managing the work space platform environment?
- How can the employment of a description logic with formal semantics and reasoning capabilities enhance the definition of a DSL in a meta modelling approach?
- How and in which level of our proposed meta modelling framework (cf. Figure 1) does the integration of the description logics take place in order to assure the dynamic behaviour of the transferal management platform?

These questions are intended to be answered within our application domain. In this way, insights gained in the application domain will be valuable and will be a contribution to a general understanding of how to make use of application models, reference models, DSLs, description logics and model-driven development.

4 STATE OF THE ART

There are various approaches that combine a DSL with a model-driven approach. For example, (Nunes und Schwabe 2006) describes their HyperDe system as an environment to support the rapid prototyping of Web applications by combining model-driven development with the use of DSLs. This combination allows the developer to create code by manipulating models that specify the application. HyperDe environment supports designing Web application through a meta model instantiation. Furthermore, HyperDe extends the Ruby on Rails framework into a DSL, allowing direct manipulation within Ruby scripts of both the model and the meta model.

Similarly, (Cadavid et al. 2009) introduces a DSL into a model-driven software development process to reduce the complexity of Web applications development. The work describes the elements used in the process of transforming a UML domain model into a deployable Web application. In this way they demonstrate that models can be transformed and executed for the automatic generation of applications.

Different to existing approaches we not only employ a DSL but also include reference models in our approach to provide modelling guidance. Thus the reference models will make it easier for domain experts to create and adapt application models because they do not have to start from scratch but only need to adapt already existing meaningful model fragments from the reference models.

Among the several meta modelling frameworks available today (Kern et al. 2011), we adopt the one provided by ADOxx. ADOxx features a three-step modelling hierarchy with a rich meta-meta model (Kern 2008), i.e. the ADOxx hierarchy (Karagiannis und Visic 2011). This hierarchical approach suits our approach of first defining a DSL and then a reference model which is created using the DSL.

Additionally, in contrast to most other approaches, which are e.g. based on UML, we will employ a **description logic** for defining the DSL. We will utilize the model-theoretic semantics of the description logic to obtain a sound semantic foundation of the DSL. This will allow for terminological inferences and integrity control. The semantics will specify in which ways the language constructs can be combined and will enable the modelling tools to support the users in creating consistent and meaningful models, e.g. by prohibiting inconsistent combination of constructs (see e.g. (Staab et al. 2010)).

The application models are not just created to describe an application domain but primarily to be translated into some specific *behaviour* in the Patient Radar transferal management platform. To achieve this, application models are mapped to corresponding elements of the application framework that is the basis for the transferal management platform (cf. the similar approach in (Reimer et al. 2008). In our project we use the application framework Vivates by the Swiss Post (http://www.vivates.ch/). The mapping rules are defined in the DSL where they are attached to each meta class.

5 METHODOLOGY

Since my research work is about developing a novel kind of artefact, the design science research methodology has been adopted to successfully carry out this dissertation. The design science research allows building a sound knowledge base through cycles of artefacts construction and subsequent evaluation. In particular, my work will be based on the general design cycle (GDC) (cf. Figure 3) elaborated by (Vaishnavi and Kuechler 2007) whom applied it into design science research.

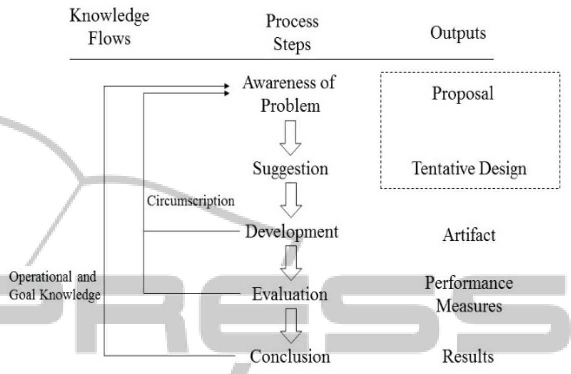

Figure 3: Reasoning in the general design (GDC) adapted from (Vaishnavi and Kuechler 2007).

The cycle depicted in Figure 3 shows that the design begins with *Awareness of Problem*. In it, the problem is identified and defined. More specifically, a case study will be created and approved among the participant of the Patient Radar project. The research strategy "case study" thus provides a concrete contribution to better understand the application domain, i.e.

- Which terminology needs to be developed within the application domain
- To which general degree a reference model and an application model need to be designed,
- Which elements of an application model need to be mapped to corresponding elements of the give application framework.

Techniques to be used in creating the case study are literature review, expert interviews involved in the Patient Radar project such as Physician, Nurses, eHealth professionals.

Next, in the *Suggestion* phase, a problem solution is abductively developed and a tentative design is given. Collected data in the *Awareness of Problem* together with further literature review will help elaborate the conceptual model related to the domain-specific terminology, reference model and application models. Additionally, the mapping rules in the DSL where they are attached to each meta class will be defined.

Then in the *Development* phase, the design is further refined and an actual artefact is produced through many iterations.

I will implement the conceptual model by using the ADOxx Modelling Toolkit Platform. Hence, guidelines and best practices implemented in ADOxx will be taken into consideration. Moreover, further literature will be sought aimed at implementing the mapping rule which then allow to establish and maintain executability of the transferal management platform.

Subsequently, we reach the *Evaluation* phase. Here the artefact is evaluated by using the well-known Technology Acceptance Model (TAM) which provides a valid and reliable measure to predict acceptance or adoption of new technologies by end users (Davis et al. 1989; Davis 1989). Additionally, TAM is a commonly used model to measure technology acceptance (King and He 2006).

Finally, the *Conclusion* phase, in which insights gained from the work are reported and future research will be addressed.

6 EXPECTED OUTCOME

The work aims at adopting a model-driven approach to create and maintain a transferal management platform for supporting the collaboration between acute hospitals and rehabilitation clinics to optimize transferal management. Thus, the expected outcome is a framework which allows the development of the transferal management platform being highly configurable, e.g. to accommodate new clinical pathways, and to permit a given pathway to (slightly) differ between hospitals. Moreover, the platform should be easily extendable to include additional functions to meet future needs.

All domain-specific aspects will be described declaratively in a reference or application model. The elements in the application model will be mapped to corresponding elements in an application framework to obtain the executable platform. The mapping will be enabled by mapping rules that are defined on the constructs of a domain-specific language (DSL). In this way, it can be assured that executable code derived from all reference/application models expressed with the DSL. The mappings from DSL constructs to application framework elements will be specified using the description logic semantics to enable the modelling tools supporting the users in creating consistent and meaningful models.

REFERENCES

Cadavid, J. J.; Quintero, J. B.; Lopez, D. E.; Hincapié, J. A. (2009): A Domain Specific Language to Generate Web Applications. In: Antonio Brogi, João Araújo und Raquel Anaya (Hg.): Memorias de la XII Conferencia Iberoamericana de Software Engineering (CIbSE 2009), Medell\'ın, Colombia, Abril 13-17, 2009, S. 139–144.

Davis, F. D. (1989): Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology. In: *MIS Q* 13 (3), S. 319–340. Online verfügbar unter http://dx.doi.org/10.2307/249008.

Davis, F. D.; Bagozzi, R. P.; Warshaw, P. R. (1989): User Acceptance of Computer Technology: A Comparison of Two Theoretical Models. In: *Management Science* 35 (8), S. 982–1003. Online verfügbar unter http://EconPapers.repec.org/RePEc:inm:ormnsc:v:35:y:1989:i:8:p:982-1003.

Jouault F.; Bézivin, J. (2006): Km3: a dsl for metamodel specification. In: In proc. of 8th FMOODS, LNCS 4037: Springer, S. 171–185.

Karagiannis, D.; Kühn, H.: Metamodelling Platforms. In: In Proceedings of the 3rd International Conference EC-Web 2002 – Dexa 2002, Aix-en-Provence, France, 2002, LNCS 2455: Springer-Verlag, S. 182.

Karagiannis, D.; Visic, N. (2011): Next Generation of Modelling Platforms. In: Janis Grabis und Marite Kirikova (Hg.): Perspectives in Business Informatics Research, Bd. 90: Springer Berlin Heidelberg (Lecture Notes in Business Information Processing), S. 19–28. Online verfügbar unter http://dx.doi.org/10.1007/978-3-642-24511-4_2.

Kern, H. (2008): The Interchange of (Meta)Models between MetaEdit+ and Eclipse EMF Using M3-Level-Based Bridges. In: Jeff Gray, Jonathan Sprinkle, Juha-Pekka Tolvanen und Matti Rossi (Hg.): 8th OOPSLA Workshop on Domain-Specific Modeling at OOPSLA 2008: University of Alabama at Birmingham, S. 14–19.

Kern, H.; Hummel, A.; Kühne, S. (2011): Towards a Comparative Analysis of Meta-Metamodels. In: Proceedings of 11th Workshop on Domain-Specific Modeling (DSM'11). Online verfügbar unter http://www.dsmforum.org/events/DSM11/Papers/kern.pdf.

King, W. R.; He, J. (2006): A Meta-analysis of the Technology Acceptance Model. In: *Inf. Manage.* 43 (6), S. 740–755. Online verfügbar unter http://dx.doi.org/10.1016/j.im.2006.05.003.

Laarman, A.; Kurtev, I. (2010): Ontological Metamodeling with Explicit Instantiation. In: Mark Brand, Dragan Gašević und Jeff Gray (Hg.): Software Language Engineering, Bd. 5969: Springer Berlin Heidelberg (Lecture Notes in Computer Science), S. 174–183. Online verfügbar unter http:// dx.doi.org/10.1007/978-3-642-12107-4_14.

Lodderstedt, T.; Basin, D.; Doser, J. (2002): SecureUML: A UML-Based Modeling Language for Model-Driven Security. In: Springer, S. 426–441.

Mernik, M.; Heering, J.; Sloane, A. M. (2005): When and How to Develop Domain-specific Languages. In: *ACM Comput. Surv.* 37 (4), S. 316–344. Online verfügbar unter http://doi.acm.org/10.1145/ 1118890.1118892.

Nunes, D. A.; Schwabe, D. (2006): Rapid Prototyping of Web Applications Combining Domain Specific Languages and Model Driven Design. In: Proceedings of the 6th International Conference on Web Engineering. New York, NY, USA: ACM (ICWE ’06), S. 153–160. Online verfügbar unter http:// doi.acm.org/10.1145/1145581.1145616.

Ranabahu, A.; Sheth, A.; Manjunatha, A.; Thirunarayan, K. (2012): Towards Cloud Mobile Hybrid Application Generation Using Semantically Enriched Domain Specific Languages. In: Martin Gris und Guang Yang (Hg.): Mobile Computing, Applications, and Services, Bd. 76: Springer Berlin Heidelberg (Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering), S. 349–360. Online verfügbar unter http://dx.doi.org/10.1007/978-3-642-29336-8_24.

Reimer, U.; Heck, U.; Streit, S. (2008): Collaboration-Oriented Knowledge Management Using Interaction Patterns. In: Takahira Yamaguchi (Hg.): Practical Aspects of Knowledge Management, Bd. 5345: Springer Berlin Heidelberg (Lecture Notes in Computer Science), S. 26–37. Online verfügbar unter http://dx.doi.org/10.1007/978-3-540-89447-6_5.

Staab, S.; Walter, T.; Gröner, G.; Parreiras, F. S. (2010): Model Driven Engineering with Ontology Technologies. In: Proceedings of the 6th International Conference on Semantic Technologies for Software Engineering. Berlin, Heidelberg: Springer-Verlag (ReasoningWeb’10), S. 62–98. Online verfügbar unter http://dl.acm.org/citation.cfm?id=1886135.1886138.

Ulrich F. (2010): Outline of a method for designing domain-specific modelling languages. University of Duisburg Essen (42). Online verfügbar unter http://hdl.handle.net/10419/58163.

Vaishnavi, V. K.; Kuechler, W. (2007): Design Science Research Methods and Patterns: Innovating Information and Communication Technology. 1st. Boston, MA, USA: Auerbach Publications.

van Deursen, A.; Klint, P.; Visser, J. (2000): Domain-specific Languages: An Annotated Bibliography. In: *SIGPLAN Not* 35 (6), S. 26–36. Online verfügbar unter http://doi.acm.org/10.1145/352029.352035.