SmartModels – A Framework For Generating On-Line Learning Software Solutions

E. Ţundrea*, P. Lahire**, D. Pescaru*, C. B. Chirilă*

* "Politehnica" University of Timişoara
Dep. of Computer Science, V. Pârvan no 2, Timişoara, România
emanuel@emanuel.ro, dan@cs.utt.ro, chirila@cs.utt.ro

** Laboratoire I3S (CNRS-UNS)
2000 R. des Lucioles, BP121, Sophia-Antipolis, France
Philippe.Lahire@unice.fr

Abstract

Globalization has set a strong mark on the way education can be provided. Every knowledge institution provider has to publish and offer his expertise for a wider readership in order to stay on top. One of the prime-time opportunities is to build web-based software solutions in order to support distance learning. There are so many ways to organize your class and professors can imagine so many ways to evaluate students that the complexity of such a software system can be hard to implement and it can hardly foresee future education forms.

This is one of the main concerns of the Object-Oriented Programming (OOP) principles which did not cure important issues faced by software companies these days on developing complex software for reuse and protecting the more and more evolving applications against technological obsolescence.

The Model-Driven Architecture (MDA) project from OMG promotes the use of meta-modeling in order to drive the system's design and implementation. In this context the paper presents:

- *an approach: it reviews the SmartModels approach briefly introducing its principles, basic entities and main elements when defining a business-model;*
- *a prototype: SmartFactory which is based on Eclipse platform and its role is to validate the new approach. It addresses the paradigm of how to practically implement MDA principles and rules for software engineering. Therefore, it deals with important implementation issues based on Eclipse Platform.*

The examples in this paper target the process of developing e-learning deployment tools.

1. Smartmodels – An Approach Based on Models

SmartModels aims to address in a practical way the MDE [6] principles. It is an approach which integrates these concepts and proposes a way of developing domain-specific software based on models as a more flexible option to the MOF [6] plus UML aproaches. It gains know-how from a previous research also on meta-modeling [5] and together with it we developed a prototype called SmartFactory in order to validate and get feed-backs from possible users.

MDA approach, as OMG[6] established it, has the advantages of stability and platform-independence through defining business functionality and behavior in a base PIM technology-independent way. This means that an approach based on models has many advantages primarily from the design point of view, but also from the implementation point of view (application coding, management and maintenance). There are many proven examples on developing standards like SQL, GUI builders, HTML or regular expressions.

One of the main problem the companies face today is that even if we have a perfect model the programmers have to make a lot of compromises when trying to implement the model using a specific programming language and mapping to a platform. This problem applies to the process of developing e-learning deployment tools when trying to encapsulate all type of possible knowledge presentations or questions from an assessment.

SmartModels tries to reduce this gap between the design and implementation and to ensure the independence between the model, future family of applications and the technology (which evolves so rapidly). Through its small kernel and a set of basic

entities, it provides a framework to describe models and a software factory to automatically generate code as much as possible. This means that the applications may be re-generated at any time if the model or the technology evolves and also the model instances can drive the behavior of the application at code generation time or at run-time. Therefore, SmartModels applies MDA through reusing the know-how of promising platforms for building integrated development environments (IDEs) like SmartTools [1] or Eclipse [10].

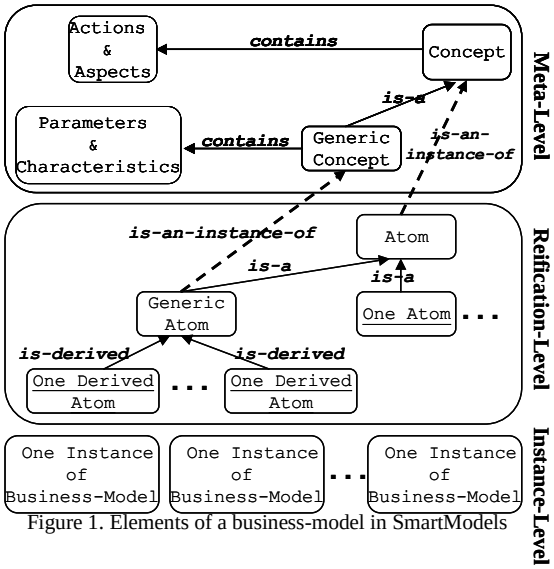

Figure 1. Elements of a business-model in SmartModels

The main objective of SmartModels is on the one hand, to clearly identify, thanks to a meta-level, the semantics of concepts used for the modeling of a given domain, and on the other hand, thanks to approaches by separation of concerns and generative programming [2], to equip, in a modular way, the applications related to this domain ([9]).

Next sections will present each entity of SmartModels with respect to the level it manifests. Figure 1 distinguishes between the different levels of the architecture of our meta-model: the main elements proposed by SmartModels in order to define business models. The second chapter presents the process of deploying these models built around e-learning tool examples using our SmartFactory prototype.

1.1. SmartModels Meta-Level

First of all, the *meta-level* is the top level of SmartModels business-model reification and it handles the meta-information through concepts. A concept participates to the definition and the management of the meta-information of a business-model. A separation between the meta-level and the reification-level is important in order to clearly identify the relevant common and variable concerns of a family of entities which compound a domain and even identify inter-related concerns of a set of closely-related domains. The meta-level contains the model's entities information (abstract) and rules which define their behavior and set the foundation for describing their infrastructure requirements.

The definition of the meta-information of a model is under the responsibility of an expert of the business domain to which the model is dedicated. Therefore it encapsulates the semantics of entities and their treatments. It can be related to one or a number of *atoms* and drives their behavior. Just as a forward-looking we mention that in our approach an atom is the structure which encapsulates the description of an entity.

The main entity which points out the clear separation between the semantics (meta-level) and their reification (reification-level) of the SmartModels model entities is the *Concept*. The choice to encapsulate the meta-information at the meta-level has a couple of very important advantages:

- The support for reuse of the semantics in other (closely related) models;
- The maintenance of the semantics (updating and redefining of the semantics) deals only with the meta-level (concepts);
- The model transformation which is one of the goals of our approach.

The semantics of a business-model stored in a concept are reified through a set of hypergeneric parameters and characteristics [5] (which form the meta-information) and a set of actions (which perform treatments on the entities according to their meta-information). The identification of the parameters and characteristics and their possible values is the job of the meta-programmer which addresses the know-how of the business-domain. Based on the set of parameters corresponding to a concept we can make a differentiation between the families of entities of a domain and based on their values we can distinguish the entities within a family.

The hypergeneric parameters customize the behavior of the entities (it refers to generic atoms – see section 2, and not their instances) of a business-model.

Their role is to capture and express the properties which compound the definition of the generic entities. A parameter expresses a basic type property, e.g. a boolean or an integer value, an enumeration, a tuple type or a collection of those values. A characteristic expresses a property whose value is defined by atom(s) which are defined within the model (enumeration, tuple or collection). The programmer has to set those values to describe the behavior of a generic entity. For example, a business-model built to encapsulate the structures (entities) and semantics of a tool to create on-line assessment (quizzes) solutions may present parameters like:

- *MultipleAnswerCardinality* which tells if a question corresponds to a single (1) or multiple possible correct answer (2..*) or
- *ForceExactAnswer* which expresses the requirement to accept only precise answers (TRUE) in case of expecting a name or checking for spelling mistakes, or if it is interpreted together with the first parameter it can have the meaning of accepting an answer only if all choices are correctly set, or
- *TimeLimit* which adds the aspect of time limitation for the specified assessment or question,

and characteristics like:

- *PossibleImageTypes* which indicates the list of accepted picture file types (let's assume that images are reified through basic atoms in our model).

Actions are "first-class" entities described by concepts in order to dynamically manage the behavior of atoms according to their meta-information. The body of an action encapsulates the execution which can be performed by that action. This execution usually takes into account the set of parameters and characteristics of the generic atom to which the action is attached and optional can present a set of aspects [4], invariants, preconditions and postconditions. For example, an action can check the remaining time to limit the work on a question or can verify the image links that we try to import in the project.

We arrived at the line of demarcation between semantics (*the meta-level*) and data of a business-model (*the reification-level*). As we anticipated in the previous section, an atom is the reification of entities of a business-model. Identifying the atoms of a domain is an important task of a programmer.

1.2. SmartModels Reification-Level

The entity of a business-domain is encapsulated in a model hierarchy through *atom* – a structure which holds the entity data and which is similar to the MOF [6] "class" notion and available in most of the object-oriented programming languages (OOPL). It can be used to factorize the data of a domain and has instances within the applications which rely on the given business-model.

In this context it is important to learn about the SmartModels distinction between *basic and generic atoms*. An atom is generic if its meta-information presents parameters and/or characteristics. If an atom does not need additional semantics besides its data-model, we call it *basic* and it will have direct instances within applications.

Now is the time to introduce also the notion of *derived atom* (see figure 1) which is an instance of a generic atom obtained through relevant combination of values associated with the sets of characteristics and parameters which participate to the definition of its *generic* atom.

If we go back to the previous example of a tool to create on-line assessment solutions we can identify:

- *basic atoms*: image types, answers
- *generic atoms*: a question with hyper-generic parameters, characteristics and actions like those presented at section 1.1. We can imagine its heirs being all sorts of question types: hot-spot (allow student to answer by selecting an image from a set), forms to entry text, drag and drop images (set up a draggable image over a list of possible correponding images), labelling (label a set of images to match their text descriptions), text identification (for example to identify each spelling mistake in a passage), true/false questions. The list of generic atoms can be designed so we can benefit by the advantages of polymorphism In this way we can continue to enrich the model by adding other heirs to describe new types of questions at the meta-level (to enhance their semantic information) and at the reification level (to enhance their structural properties).
- *derived atoms*: examples of questions having different properties: i.e., a free-form text edit derived atom is a text form question atom with the following meta-information (see section 2.4 for a detailed diagram and presentation):
    - forced exact answer: FALSE;
    - time limit: NONE.

or a type of image labelling can be described by:

- multiple-answer cardinality: 4 (the correct choices can be up to four);
- forced exact answer: TRUE;
- time limit: "00:05:00" (exactly five minutes);

- image types accepted: basic atoms like JPEGImage, PNGImage, BMPImage.

Actions will check the conformance of the derived atoms' parameter values during the creation of new questions or at run-time checking the constraints (time limit, mandatory exact answer).

1.3. SmartModels Instance-Level

At this point we can see the benefits of the SmartModels approach. A possible application to the example mentioned above can be a tool for easy developing and deploying flexible and reusable on-line learning software systems, which describes and presents knowledge and also which can generate assessments and provide assistance on the evaluation process.

This chapter summed up the design of our approach. Next chapter presents SmartFactory – a practical implementation of SmartModels principles. We will try to demonstrate its interest through building the above mentioned domain model and generating its applications.

2. SmartModel of the On-Line Learning Assessment System

In this chapter let's explore more the example of building a model through our approach for deploying a tool to develop on-line learning assessment system. Figure 2 presents the UML class diagram of the concepts and atoms which participate to the semantic and structural description of our model.

The *BasicConcept*, *BasicAtom* and the related generic atom and concept pointed out as supertypes for our model entities are presented just for the purpose to underline the way we attach each new model to our SmartModels kernel. They represent the abstract entities from the built-in metamodel.

Before going further it is important to see the SmartModels methodology to describe a business model. This is a five-step process:

1. to identify the basic atoms of the model,
2. to identify the generic atoms,
3. to define the criteria of genericity (the hypergeneric parameters) - typically this is a step that must be performed by an expert of the domain. It represents a part of the knowledge of the business model;

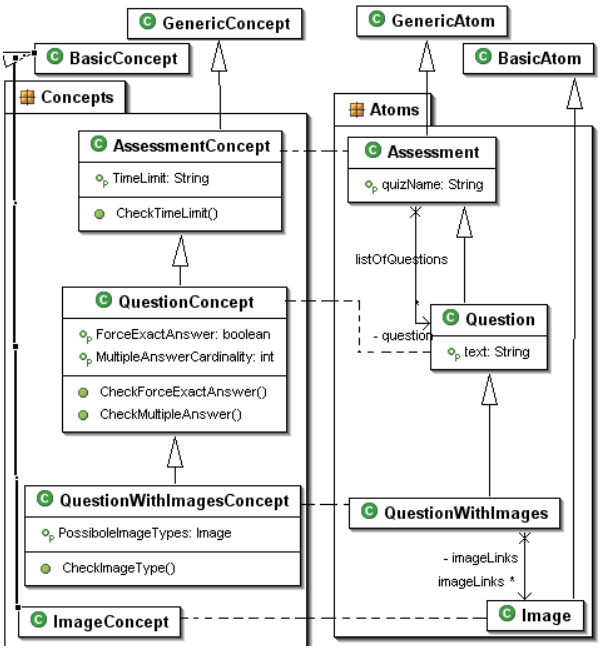

Figure 2. On-Line Assessment SmartModel Class Diagram

4. to specify the actions attached to generic and non-generic atoms,
5. to specify the instances of the generic atoms (derived atoms).

We will follow this methodology and the next sections will highlight a couple of important features of SmartModels and their advantages applied to this specific use-case.

2.1. Define Meta-Information of Complex Entity Families

One of the hardest part of creating a flexible tool for developing on-line assessment is that there are many ways a professor can imagine the evaluation of students. One can decide to create multiple-choice questions and require correct answers on all choices to mark all points. Another professor can decide to mark just the good answers and offer some points; others may even think of a weight for each answer and subtract points if a student makes wrong choices.

We can also imagine a requirement for evaluating a quiz to get an exact answer. This can mean checking for spelling mistake in case of a free-form text question or labelling correctly a set of images.

Should the structural entities of our model which describe the different types of questions of the quiz be

equipped with all the information about all possible ways to evaluate them? SmartModels is a framework which makes a clear differentiation between the semantic information and reification of the families of entities of a domain. In our case, we can unambiguously separate the description of the structural features of each question type from the concerns that deal with the process of evaluating them.

This means that the user will define atoms only to encapsulate the different structural features of each question type without having to concentrate on the way they will be avaluated or mixing each type of question with all its possible evaluation manners. On a meta-level, the user can concentrate on creating rules to evaluate a quiz. More than that, he can create rules which apply at the level of a single question or a set of question types (i.e., how are multiple-choice questions marked) or even rules for the whole quiz (i.e., setting a time limit).

2.2. Carry on all Benefits of Polimorphism

This paragraph underlines a simple truth about SmartModels, but very powerful: it is built in the context of object-oriented technology and therefore it makes use of the notion of polymorphism.

More than that, this statute of SmartModels applies both at the level of concepts and atoms. This is one of the main reasons we can create families of entities. As a consequence, for both the semantic information and the structure of the entities of a domain we can easily:

- extend or refactor the model;
- reuse entites and their properties in order to describe more specialized entities in the same model or even in other closely related models (inheritance at the level of models).

As we already mentioned it, a quiz may contain questions dealing with text and/or images. To keep things simple in this paper we considered only the case of questions with text and images, but if we look in the second diagram we can notice that with a little change we can encapsulate meta-information of the different question concepts in different trees.

Figure 2 presents the *QuestionConcept* which addresses the semantic information of a general text type question but also its specialization *QuestionWithImagesConcept* which concentrates on more specific characteristics dealing with image manipulation. In this way, the instances of this concept (the corresponding questions with images atoms) will use semantic information about the text column of the question through inheritance.

PossibleImageTypes characteristic indicates the collection of accepted types of images which can be handled by our tool. Notice that the type of information that this characteristic uses is also an entity of the model: the basic atom *Image*. Again, this is the distinction that SmartModels makes between parameters and characteristics (see section 1.1). For example, the parameter *MultipleAnswerCardinality* is of type integer with possible value between 1 and usually 4 (most of the quizzes we have seen have questions which include 4 choices, but it should not be limited); and parameter *ForceExactAnswer* of type boolean.

2.3. Insert Dynamic Aspects: Actions

What if after deploying the first version of the tool for developing on-line assessments we need to add a new mode of organizing the exam which was not planned at the model design stage: for example "to enforce a time-limit". Typically a professor should be able to stipulate when he creates the quiz even if it was not included in the original model because it should be checked at run-time. In a classical object-oriented approach, it would lead to considerable changes in the structure of entities and in their behavior in order to implement this enhancement.

Thanks to the aspect-oriented approach proposed in SmartModels, it provides the opportunity to attach actions (*CheckTimeLimit*) to each concept to dynamically control the behavior of entities. If we combine this opportunity with the fact that we can benefit from inheritance, the actions increase the level of flexibility of the model: a professor may think either to set a time limit on individual questions if he likes or a global timer for the whole exam if we place the time-limit parameter at the level of the assessment question (see figure 2).

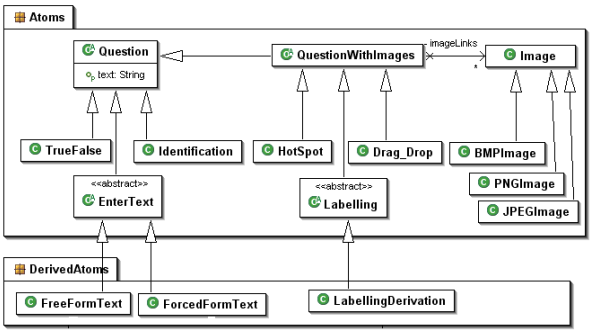

Figure 3. Atoms and Derived-Atoms

2.4. Derived-Atoms Explosion

Derived-Atoms are another mean provided by SmartModels to enrich the model and capture in the modeling phase as much as possible the commonalities and variabilities of the domain entities.

If we consider again our example, figure 3 presents a couple of possible types of text questions and text and image questions. Now we can equip our on-line assessment deploying tool with more and more question types in two ways:

- either to create new atoms (creating new hierarchies of atoms in case of new entities form the domain or creating heirs of existing atoms to obtain specialized atoms through inheritance) or
- to derive new atoms from generic atoms in order to create new entities through a relevant combination of parameter and characteristic values. The number of combination possibilities is therefore limited only by the richness of the semantic information described through parameters, their type ranges and relevance in relation with other parameters from the same conceptual tree.

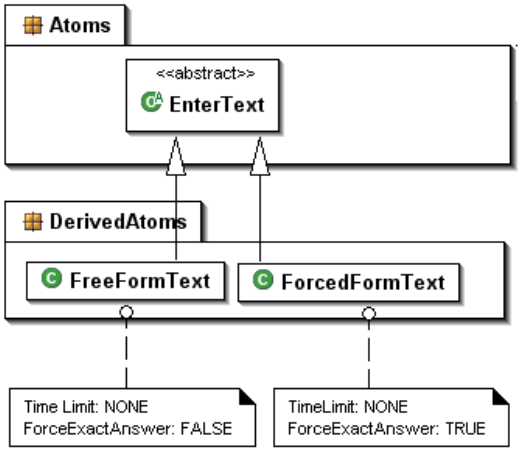

Figure 4 Edit-Text Derived Atoms

Now we can see the effects of setting the *ForceExactAnswer* parameter value to true or false when creating a new instance of *ForceFormText/ FreeFormText* derived-atom at run-time. Other variation of this edit-text type questions may have a time limitation which can also be specialized to be an exact period of time or a fixed date and time value (a professor may not want to set a timer on the exam editing, but to set date and time limit until the assignment may have to be submitted).

Another interesting illustration would be to consider a labeling type of question when the student has to associate to a set of images a set of names or statements. A derived labeling atom can be obtained through a combination of values of its concept parameters. If *MultipleAnswerCardinality* is set to "4" then this means that there will be a question where there will be four choices to be presented to the student. We can set a timer on this question and we can enumerate the types of images that can be displayed by this tool (*{JPEGImage, BMPImage, PNGImage}*).

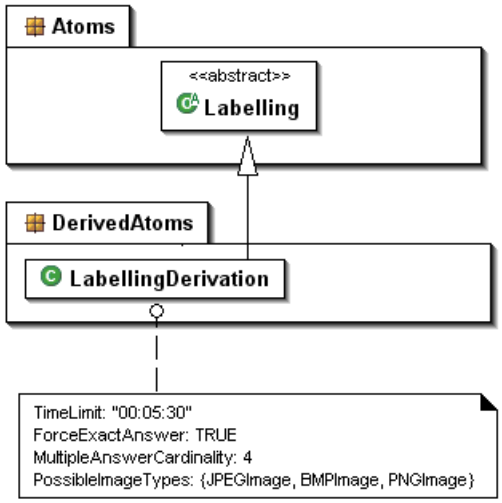

Figure 5. Labelling Derived-Atoms

Similar to the previous example, we can imagine other possible derivations. Adding a new parameter to specify the number of labels and the number of images to match, we can obtain an even more flexible way to manage the creation of labeling questions. In this way a professor may create a question with ten labels from which to chose only the four valid names for the right column images.

Of course the choice of the concept and/or atoms as well as the associated parameters and characteristics may be discussed by an expert of the domain.

The aim of these examples is only to show the expressiveness of SmartModels approach to capture within a model as much as possible information which can then be automatically generated on a specific platform and mapped to an up-to-date technology.

Conclusions

It is very important to understand that in the world of software everything evolves: technologies, methodologies and applications.

We believe that in order to provide an approach centered on models, which capture the know-how of a

domain, it is of primary importance to ensure the independence between both the model and the software platform and between the model and the possible applications. This article promotes the idea that model-oriented programming is a better approach to solve these new challenges.

These ideas have been around for a couple of years, but today there is no major vendor which gets behind OMG's MDA initiative and makes it happen in software development. Our approach together with the prototype wants to form a possible and feasible way to apply these principles. It can happen in JetBrains or Eclipse and based on this last experience we propose a way to address meta-modeling issues extending their know-how. SmartModels is a MDA approach which provides a framework to create models that capture information about a business-domain independent from a technology, platform or programming language.

The first chapter introduces SmartModels, and presents its principles, basic entities and main elements when defining a model, which aim to match the requirements of an approach centered on models. Next chapter is a second contribution which investigates in the context of a real example a couple of important advantages of developing through our approach.

Because of the growing interest around educational technologies we decided to experiment our approach for the description of various business models and their applications of this domain. Therefore, in this paper we started to investigate the business models of a development and deploying tool for on-line learning solution and evaluation. The objective is to get feedbacks in order to improve the expressiveness of SmartModels – how to ease the job of a meta-programmer to describe a model, as well as a better automation.

REFERENCES

[1] I. Attali, C. Courbis, P. Degenne, A. Fau, D. Parigot, C. Pasquier and C. Sacerdoti (May 2001), SmartTools: a development environment generator based on XML technologies, *XML Technologies and Software Engineering*, Toronto, Canada, ICSE'2001 workshop.

[2] K. Czarnecki, U. Eisenecker, June 2000, Generative Programming: Methods, Techniques and Applications, Addison-Wesley.

[3] K. Czarnecki, J. Vlissides (2003), Domain-Driven Development, Special Track *OOPSLA'03*, oopsla.acm.org/ddd.htm

[4] G. Kiczales, J. Lamping, A. Menhdhekar, C. Maeda, C. Lopes, M. Loingtier, J. Irwin (June 1997), Aspect-Oriented Programming, *ECOOP'97 – Object-Oriented Programming 11th European Conference*, Jyväskylä, Finland, vol 1241, *Lecture Notes in Computer Science*, pages 220-242, Springer-Verlag.

[5] P. Lahire, D. Parigot, C. Courbis, P. Crescenzo, E. Ţundrea, *An Attempt to Set the Framework of Model-Oriented Programming*, 6th International Conference on Technical Informatics (CONTI 2004), Timişoara, România, May 27-28, 2004, Proceedings Periodica Politechnica, *Transactions on Automatic Control and Computer Science*, Vol. 49 (63), 2004, ISSN 1224-600X.

[6] Object Management Group, Model-Driven Architecture (MDA) and Meta Object Facility (MOF) Specification, www.omg.org/mda

[7] J. Palsberg, B. Jay (August 1998), The Essence of the Visitor Pattern, *COMPSAC'98, 22nd Annual International Computer Software and Applications Conference*, Vienna, Austria.

[8] E. Ţundrea, P. Lahire, D. Parigot, C. Chirilă, D. Pescaru, (May 25-26, 2004), SmartModels – An Approach For Developing Software Based On Models, *1st Romanian - Hungarian Joint Symposium on Applied Computational Intelligence SACI'2004*, Timişoara, Romania, ISBN 963-7154-26-4, pages 231-240

[9] P. Crescenzo, P. Lahire, E. Ţundrea, SmartModels: la généricité paramétrée au service des modèles métiers, LMO 2006, pages 151-166, 22-24 March 2006, Nîmes, France

[10] Eclipse Foundation Community Inc., http://www.eclipse.org/