

Project VIDE – Challenges of Executable Modelling of Business Applications

Radoslaw Adamus^{*†}, Grzegorz Falda^{*}, Piotr Habela^{*}, Krzysztof Kaczmarek^{##},
Krzysztof Stencel^{*†}, Kazimierz Subieta^{*}

^{*} Polish-Japanese Institute of Information Technology, Warsaw, Poland

[†] Computer Engineering Department, Technical University of Lodz, Lodz, Poland

[#] Warsaw University of Technology, Warsaw, Poland

⁺ Institute of Informatics, Warsaw University, Warsaw, Poland

{gfalda, habela, stencel, subieta}@pjwstk.edu.pl, radamus@kis.p.lodz.pl,
k.kaczmarek@mini.pw.edu.pl

Abstract. The VIDE project investigates the potential of the Model Driven Architecture in terms of increasing the productivity, improving complexity management and raising the non-IT stakeholders involvement in the process of developing business software. The main focus of the research is the Platform Independent Model (PIM) level language designed to comply with Object Management Group modelling specifications. The project attempts to adapt techniques from the programming world to that level, including seamless integration of a query language, aspect-oriented composition and quality defects detection. To reduce the gap from the business modelling perspective, the project also adopts a Computation Independent Modelling language and supports its' transition to the PIM and traceability thereafter. The direct benefits of such work are strongly dependent on the ability to make such PIM models easily transformable to execution platforms. The experience of project participants in application server platforms, as well as with experimental database systems development has been useful. In this paper, the work on refining and implementing the Unified Modelling Language and Object Constraint Language to allow executable modelling will be described.

1. VIDE Project Overview

The VIDE project (Visualise All Model-Driven Programming) is aimed at exploiting the concepts of model driven development (and the Object Management Group (OMG) Model Driven Architecture (MDA) particularly) in order to make the development of data-intensive applications more productive, as well as approachable for non-IT stakeholders. The project is designed to find the optimum solution by exploring ways of raising the level of abstraction with the use of appropriate models. When a higher level of abstraction is considered, one may expect sometimes contradictory features of the solution. On one hand, the higher level might assure an easy and intuitive means of specifying the domain by non-programmers. On the other hand, the aim could be for more expressive power and scalability. This would be achieved at the cost of language sophistication and would hence require expertise to take advantage

of it. A similar issue occurred when database query languages were proposed. At this time both these goals were considered, whether to go for greater accessibility or greater expressiveness. The greater expressiveness view prevailed. Interestingly, as a whole, the VIDE project has both these goals and to aid clarity we have defined the following priorities.

The primary aim is the development of a productive and powerful tool for IT specialists, which takes advantage of the findings in the area of programming and query languages, and shifts the results to the platform-independent level. In general, models can be applied at various levels of detail and formality. Usually, the following names are used to describe those levels:

- *Sketches*, which represent the traditional use of UML and similar language as a help in understanding the problem and communicating ideas and solutions to other developers. Those kinds of models do not need to be complete nor fully formalized.
- *Blueprints*, which follow the traditional distinction between design and its realization, as in case of other engineering domains. The distinction of design and coding is maintained in terms of artifacts and is also reflected in assignment of those tasks to different groups of developers.
- *Executable models*, which require the presence of precise semantics and, by the automation of executable code production, blur the distinction between modelling and programming.

With the above assumptions in mind, we can state that VIDE attempts to follow the last of those levels, that is, executable modelling at the platform-independent level.

With executable modelling as a foundation, the secondary aim is to realise the other goal, that is, making the tools more accessible to non-IT users thus reducing the gap between the business and the software development views. This is realised by developing a layer in which a less complex (and also thus less precise) model can be developed by business analysts and stakeholders themselves in order to capture the domain description. Following the MDA terminology, this is provided by a Computation Independent Model (CIM) layer. There are numerous tools dealing with business process modelling and related views at this level (such as the conceptual data view, the organisational view or the business rules view). But the project expects to demonstrate gains by supporting a consistent and gradual transition from CIM to PIM and completing it with even more informal modelling notions as the entry step to CIM modelling.

Such a general formulation of the research goal implies a broad range of modelling topics that need to be addressed:

- Evaluating and adapting existing, yet immature, modelling specifications. In particular, the units responsible for detailed behaviour specification in UML (UML Actions and Structured Activities) [OMG2007], as well as the clarification of their underlying semantics. This will require a proposal for a concrete syntax (visual and textual) for constructs which are not defined in the UML specification.
- Providing the resulting language (due to the anticipated data-intensive nature of its applications) with a query language capability and assuring its seamless integration. This work involves adapting the OMG Object Constraint Language (OCL)[OMG2006b]. Although the choice may appear obvious because OCL is an existing, standardized and powerful language which allows the formulation of ex-

pressions against UML models and their instances. The application of OCL to this role required overcoming several problems not foreseen by the language creators.

- Enriching the modelling language with the means of dealing with crosscutting concerns, based on an aspect-oriented composition approach.
- Providing the tool with an extendable specification of undesirable structural and behavioural model characteristics, and with the ability to visualize those defects on demand inside a model editor.
- Providing the model editor functionality which allows the efficient switching from structural modelling to dynamic (behavioural) modelling thus taking advantage of the assumed development process.
- Providing the toolset with CIM level modelling functionality. The OMG standard Business Process Modelling Notation (BPMN) was selected as a base [OMG2006c] and implements a chain of transitions from natural language descriptions. This chain extends through the adapted BPMN models towards Unified Modelling Language (UML) compliant incomplete PIMs. These gradual refinements of the model assure its traceability towards the requirements. A transition from CIM towards executable process definition is also supported.
- Assuring a straightforward way of running the detailed PIM in order to validate that the constructed functionality matches the actual requirements.
- Creating model compilers that automate code generation for the target platforms to raise productivity and avoid errors occurring during manual code elaboration.

Whilst some of these features are able independently to offer improvement in modelling tools, much of the effort comes to fruition only when the PIM is executable and code generation is successfully achieved. Otherwise, the effort needed for the precise behaviour modelling could be questionable because more or less effort depends on how much code can be actually automatically created for the target execution platform.

MDA popularises the idea of shifting software development from the programming environment towards modelling environments reflected in an earlier abstraction from assembly languages to programming languages [MSUW2004]). These promises create an impression that the problem of model execution and code generation is solved. However, this is not the case – especially for data oriented applications that involve complex software technologies in addition to complex language constructs. The difficulty of model transformations and the complexity involved have caused reservations towards the overall advantage of MDA approach [HT2006].

In this paper we focus on the model execution as this is the area where fairly complete results are already available. Currently, we can confirm, that a sufficiently flexible database programming environment is capable of executing UML models of constructs outlined in the VIDE project requirements. They involve a significant part of UML Actions, UML Structured Activities and OCL expressions integrated under a common concrete syntax.

In the remaining sections of this paper we briefly describe the design of the “VIDE PIM language” that involves the abovementioned constructs. We then explain the solution of the model execution engine, and finally provide conclusions and outline the future work.

2. PIM-level programming and query language

Being focused on handling data collections, the VIDE project assumed in advance the integration of a query language functionality into the UML behaviour. In order to reduce the number of solutions the VIDE toolset would need to develop from scratch and to enable promotion among developers OCL was chosen. It is the only language available as a standard that is dedicated to work with UML model instances. The original purpose for which the OCL was designed (specifying constraints over UML models) differs significantly from the role it is to play in VIDE. Hence, this required significant work to design its integration into UML behaviour. The issues like the visibility of local variables (not relevant for specifying constraints but essential for a query language) and alignment of result type systems between OCL and UML needed to be addressed. From the point of view of the current development, this way of unifying UML behaviour and OCL (currently provided as two separate and not fully synchronised specifications) into a single, not mutually redundant set of constructs seems really natural and desirable. It also allows reducing the number of UML action elements by removing those that deal with data retrieval and are covered by OCL. This includes *ReadStructuralFeatureAction*, *ReadVariableAction*, *ReadSelfAction*, *ReadLinkAction*, *ReadSelfAction* and *ReadExtentAction*.

This integration performed in VIDE makes UML behaviour capable of specifying high level expressions. This involves selection, navigation, quantifiers, ordering, operation invocations, converting collections, collecting values with aggregating functions and constructing complex result structures using tuples.

In terms of the concrete syntax, its design for imperative constructs (Structured Activities and Actions) was performed so as to avoid a mismatch with the already established OCL syntax. Moreover, the concrete syntax solution closely follows the underlying abstract syntax (UML metamodel) so as to make the mapping onto the metamodel more straightforward. Several useful shortcuts were introduced though, to make the coding simpler in case of updates and handling of collections.

An important consideration for the concrete syntax is the fact that the language will extend the visual notation of UML diagrams. In particular, the behaviour specification will be available by selecting the respective operation on a class diagram and will involve a subset of UML Activities. For this reason, an optimum balance between textual and visual syntax for the UML constructs needed to be found and in VIDE some elements are given both textual and visual representations that may be chosen by the developer.

3. Developing a model execution engine

There are important pragmatic reasons that imply the demand for making the model executable. This involves:

- Software construction because if the resulting executable code can be directly adopted on the target platform, this provides important gains in terms of development productivity and maintenance.

- Testing and debugging because if the subsequent code transformations are guaranteed not to distort the model semantics, the testing can be shifted to the PIM level.
- Prototyping and validating with final users because even if the transition toward the target execution platform is laborious, the possibility of running platform-independent models is still valuable; e.g. in order to verify with the domain experts the model behaves as intended, before further work on the implementation is performed.

Apart from that, the executable implementation is also needed for the purpose of disambiguating the UML specifications. The requirement for a reference implementation and a human-readable operational semantics has been emphasised since the early phases of UML 2 [Thom2004].

When developing the UML/OCL model execution engine for VIDE, we encountered several shortcomings of the current UML and OCL specifications. Firstly, there were a number of inconsistencies that may be easily overlooked when writing a specification, and which become visible when actual implementation is approached. This involved some ambiguities in the OCL metamodel specification and several errors in UML 2.1 ‘well-formedness’ constraints on Actions and Structured Activities. Secondly, some open questions occur regarding the handling of class extents and on providing an entry point and a kind of global environment for model execution. These issues are only partially answered by the current UML executable semantics document [OMG2006a]. From the point of view of VIDE the main shortcoming of that specification is the fact it excludes the treatment of expressions and does not consider the integration with OCL.

The implementation confirmed that the amount of information needed for model execution needs to significantly exceed what is usually available from just parsing code. For example, updating variable and object attributes, or navigating through links or through plain attributes. Although these can be represented identically in the language syntax, they may require distinguishing and proper handling when the model is executed or transformed into the source code of another language.

Our model execution engine is based on the ODRA (Object Database for Rapid Application Development) object database platform [LS2007]. It actually realises a complete model compiler producing the code of the SBQL language that is used by ODRA. This can be treated as a model execution engine because:

- It allows integration into the development environment and provides the functionality that makes the step of code generation transparent for the developer who runs the model.
- It provides a comparable level of abstraction and the availability of needed constructs at the side of SBQL. Thanks to this, runtime errors, if they occur, can be meaningful in the context of the original model.

In addition, the runtime allows the model to consume and provide Web service interfaces (according to declarations inside model), so that other executable components can be invoked from and can invoke the model.

The development of this model compiler and execution engine allowed the comparison of the capabilities of UML/OCL subset used by VIDE with the SBQL language used by ODRA. Particularly, the overall expressive power and the suitability of concrete syntaxes are to be evaluated.

4. Conclusions

Despite the MDA promise of raising the level of abstraction by shifting software development towards executable PIMs, many aspects of the respective modelling language specifications and code generation technologies remain immature, and important open questions exist.

Only after concluding the VIDE project's research in the areas of model compiler and executable model semantics, will it be possible to answer the foundational questions:

- What are the limitations of the executable modelling and automatic code generation approach in this area of applications?
- To what extent are those limitations inherent to code translation in general, and to what extent are they the result of the heterogeneity of technologies at the target platforms?

Currently, we have been able to confirm that an engine for running the detailed UML behavioural models involving OCL queries can be built and integrated with a modelling tool. The research in this area revealed several issues that need to be considered when specifying a precise executable semantics for UML models. The results will inform the process of further development of the respective OMG specifications [OMG2006a]. Integrating OCL with UML Actions allowed to significantly increase the expressive power of the resulting UML-based language. However, we are going to compare its capabilities with query languages which were originally designed as such and particularly, with SBQL.

References

- [HT2006] B. Hailpern, P. Tarr: Model-driven development: The good, the bad, and the ugly. IBM Systems Journal: Model-Driven Software Development. Volume 45, Number 3, 2006
- [LS07] M.Lentner, K.Subieta: ODRA: A Next Generation Object-Oriented Environment for Rapid Database Application Development. ADBIS 2007: 130-140.
- [OMG2006a] Object Management Group: Semantics of a Foundational Subset for Executable UML Models (Initial Submission), May 2006. ad/06-05-02
- [OMG2006b] Object Management Group: Object Constraint Language version 2.0, May 2006. formal/06-05-01.
- [OMG2006c] Object Management Group: Business Process Modeling Notation Specification, February 2006. dtc/06-02-01.
- [OMG2007] Object Management Group: Unified Modeling Language: Superstructure version 2.1.1, February 2007. formal/2007-02-05.
- [MSUW2004] S.J.Mellor, K.Scott, A.Uhl, D.Weise: MDA Distilled: Principles of Model-Driven Architecture. Addison Wesley 2004
- [Thom2004] D.A.Thomas: MDA: Revenge of the Modelers or UML Utopia?, IEEE Software 21, No. 3, 15–17 (May/June 2004).

[WK2003] J.Warmer, A.Kleppe: Object Constraint Language, The: Getting Your Models Ready for MDA. Addison Wesley 2003.