About the Specialization of Model-Driven Approach for Creation of Case-Based Intelligent Decision Support Systems

Dorodnykh N.O., Yurin A.Yu.
Matrosov Institute for System Dynamics and Control Theory,
Siberian Branch of the Russian Academy of Sciences (IDSTU SB RAS),
Irkutsk, Russia
Email: tualatin32@mail.ru
Email: iskander@icc.ru

***Abstract*—The paper describes the specialization (modification) of the model-driven approach in the context of the development of case-based intelligent decision support systems. The specialization includes: the redefinition of the basic steps of the process of the software creation (their structure and content) and the models used. It is proposed to use: ontology as the computation-independent model (CIM), which will provide a complete and consistent description of the concepts and relations of the subject domain and their visual representation in the form of a graph; XML-like specification as a platform-specific model (PSM), which takes into account features of the description of cases.**

Keywords—model-driven approach, cases, knowledge bases, decision support systems, code generation.

I. Introduction

The problem of improving the efficiency and quality of development of problem-oriented intelligent systems (including knowledge bases (KB) and expert systems (ES)) is still relevant. This problem can be solved in different ways: from improving the development approaches (methodologies) up to creating and using specialized software [1]. Thus, there are some tendencies in this area:

- the development and application of software for ontological and cognitive modeling and CASE-tools (Protégé, FreeMind, Xebece, TheBrain, XMind, IBM Rational Rose, StarUML, ect.). These systems allow to create graphical conceptual models corresponding to the key abstractions of software;
- the development and application of specialized editors and shells for ESs (Expert System Designer, Expert System Creator, ARITY Expert Development Package, CxPERT, Exsys Developer, ect.). These systems allow to implement a formalized description of domain concepts and KB structures on a certain programming language (knowledge representation language), but have the low integration ability with CASE-tools, in most cases they support the one a particular language;
- the use of integrated development environments (frameworks) and unified approaches, which provide coverage of all stages of the life cycle of knowledge based systems and integration of the first two tendencies [2], [3], [4], [5].

In general, the existing solutions (results of mentioned directions) allow to improve the effectiveness of the intelligent systems development process, but don't solve the problem of portability of the developed systems to another technology or software platform (the operating system or the programming language).

One of the solutions for this problem is to use approaches based on Generative Programming [6], in particular, a model-driven approach, which provides the ability to develop multi-platform software.

II. Model-Driven Approach

A Model-Driven Approach (Model Driven Software Development or Model-Driven Engineering, MDE) is a software design approach which uses the information models as the major artifacts that, in turn, can be used for obtaining another models and generating programming codes [7].

Thus, the core ideas of the model-driven approach are:

- a model is the key artifact during the development process of software (a formal specification of the function, structure and behavior of a system within a given context);
- the software development process is a sequence (a chain) of transformations of models (from the more abstract to the less abstract).

Some of the better known MDE initiatives are the following:

1) Model-Driven Architecture (MDA), which is a registered trademark of Object Management Group (OMG) [8], [9], [10]. The main idea of the approach is to build an abstract meta-model for the management and exchange of metadata (models) and setting the ways of their transformation into a software-supported technology (Java, CORBA, XML, etc.). MDA specifies three default viewpoints on software: computation independent, platform independent and a platform specific. The viewpoint is an abstraction

technique for focusing on a particular set of concerns within a system while suppressing all irrelevant detail. The viewpoint can be represented via one or more models;

2) Eclipse Modeling Framework (EMF) is an Eclipse-based modeling framework and code generation facility for building tools and other applications based on a structured data model [11]. EMF provides the foundation for interoperability with other EMF-based tools and applications. The heart of EMF is Ecore. Ecore is the special language for description of meta-models (implementation of OMG's Essential Meta-Object Facility, EMOF). The basic tools to work with meta-models and skeletal code generation of software (programming skeletons) are EMF.Core, EMF.Edit, EMF.Codegen;

3) Model-Integrated Computing (MIC) has been developed over two decades at ISIS, Vanderbilt University for building a wide range of software systems. MIC focuses on the formal representation, composition, analysis, and manipulation of models during the design process. It places models in the center of the entire life-cycle of systems, including specification, design, development, verification, integration, and maintenance [12]. MIC providing three core elements: the technology for the specification and use of the domain-specific modeling languages (DSML); the fully integrated metaprogrammable MIC tool suite, and an open integration framework to support formal analysis tools, verification techniques and model transformations in the development process; the three-level representation of the system development process (Application Level, Model-Integrated Program Synthesis Level, Meta-Level).

In the context of the development of case-based intelligent decision support systems, MDA is selected as the primary approach, as the most standardized version (initiative) of the MDE.

III. Model-Driven Architecture

A. The main MDA elements (specifications)

MDA itself is not a new OMG specification but rather an approach to software development [9] which is enabled by existing OMG specifications such as:

- MOF (Meta-Object Facility) – OMG's standard for model-driven engineering. Its purpose is to provide a type system for entities in the CORBA architecture and a set of interfaces through which those types can be created and manipulated. MOF is designed as a four-layered architecture (a conceptual modeling space). Every model element on every layer is strictly in correspondence with a model element of the layer above. MOF provides a means to define the structure, or abstract syntax of a language or of data [13];
- UML (Unified Modeling Language) – the OMG's standard for an object modeling in software development. UML is not the programming language, but the UML-based models allow to generate codes of software [14];
- XMI (XML Metadata Interchange) – the OMG's standard for exchanging metadata information (UML-based models, etc.) via Extensible Markup Language (XML). XMI is also commonly used as the medium by which models are passed from modeling tools to software generation tools as part of MDE [15];
- Query/View/Transformation (QVT) – the standard set of languages for model transformation (model to model transformation, M2 [6]) defined by the OMG. The QVT standard defines three model transformation languages (QVT-Operational, QVT-Relations and QVT-Core). All of them operate on models which conform to MOF meta-models. The QVT standard integrates the OCL (Object Constraint Language) 2.0 standard and also extends it with imperative features [16];
- MOF Model to Text Transformation Language (MOFM2T) – the OMG's specification for a model transformation language [17]. Specifically, it can be used to express transformations which transform a model into text (M2T) [6].

B. Software development stages

MDA specifies three default models of a system corresponding to the three MDA viewpoints defined above. These models can perhaps more accurately be described as layers of abstraction, since within each of these three layers a set of models can be constructed, each one corresponding to a more focused viewpoint of the system (user interface, information, architecture, etc.). Thus, MDA defines the software development process as a sequential transformation of these models. Consequently, the development process can be represented as a set of following stages:

- building a computation-independent model (CIM). It presents the system requirements, i.e. exactly what the system is expected to do, but hides all information technology related specifications to remain independent of how that system will be (or currently is) implemented. CIM plays an important role in bridging the gap which typically exists between these domain experts and the information technologists responsible for implementing the system;
- building a platform-independent model (PIM). PIM is a model of software that is independent of the specific technological platform used to implement it;
- building platform-specific models (PSM). PSM is a model of software that is linked to a specific technological platform (for example, a specific programming language, operating system, database, etc.). Thus PSM specializes PIM for a specific platform based on its model (Platform Description Model, PDM);
- generating program codes and/or specifications (using the generator) based on PSM.

As a result, we propose to adapt (modify) and apply the MDA/MDD approach for building the case-based intelligent decision support systems, including clarification and redefinition of the main elements and stages taking into account specifics of the developed software.

IV. Problem statement

MDA can be formalized as follows:

$$MDA = \left\langle \begin{array}{l} L, CIM, PIM, PSM, PDM, \\ F_{CIM-to-PIM}, F_{PIM-to-PSM}, \\ F_{PSM-to-CODE} \end{array} \right\rangle \quad (1)$$

where an L – visual modeling languages, $L = \{UML\}$; CIM, PIM, PSM, PDM – corresponding models; $F_{CIM-to-PIM} : CIM \rightarrow PIM$, $F_{PIM-to-PSM} : PIM \rightarrow PSM$, $F_{PSM-to-Code} : PSM \rightarrow Code$ – model transformation rules.

Thus, the main purpose of this paper is to adapt (to specialize) the MDA methodology in the context of the development of case-based intelligent decision support systems, i.e. to define an MDE^{CBR}:

$$MDA^{CBR} = \left\langle \begin{array}{l} L^{CBR}, CIM^{CBR}, PIM^{CBR}, \\ PSM^{CBR}, PDM^{CBR}, \\ F^{CBR}_{CIM-to-PIM}, F^{CBR}_{PIM-to-PSM}, \\ F^{CBR}_{PSM-to-Code} \end{array} \right\rangle \quad (2)$$

V. MDA specialization

The MDA specialization includes:

- to use the original author's notation - RVML (Rule Visual Modeling Language [18]) as the additional visual modeling language: $L^{CBR} = \{UML, RVML\}$;
- to use the ontologies as computation-independent models (CIM), which provide a complete and consistent description of the domain concepts and relations, and also their visual representation in a graph;
- to use the original XML-like specification as a platform-specific model (PSM), which provide the description of cases (particularly: possible fuzzy property values and description for a membership function) and the formation of an operational level to work with the Data Based Management Systems (DBMS).

The development process of the case-based intelligent decision support systems is presented by the following sequence of stages (Figure 1):

Stage 1. Description of the subject domain that contains the main concepts and relations. At this stage, the user creates a computation-independent model (CIM). This model can be implemented in the form of an ontology or an UML-model (in particular, as a class diagram). In addition, the main architectural elements of system (such as the "input form", the "output form", the "inference engine" and etc.) is also produced at this stage.

The efficiency of this stage can be improved by reusing (transformation) the existing conceptual models created using various ontological and cognitive editors, such as CASE-tools (e.g., IBM Rational Rose, etc.) [19].

Most of the software that supports the MDA/MDD approach does not realize this stage and only allows one to develop the software starting at the next stage. In this case, the conceptual model of a subject domain (even presented in the form of an ontology) is considered as a platform-independent model (PIM) that describes the main concepts and business logic (that is acceptable for databases).

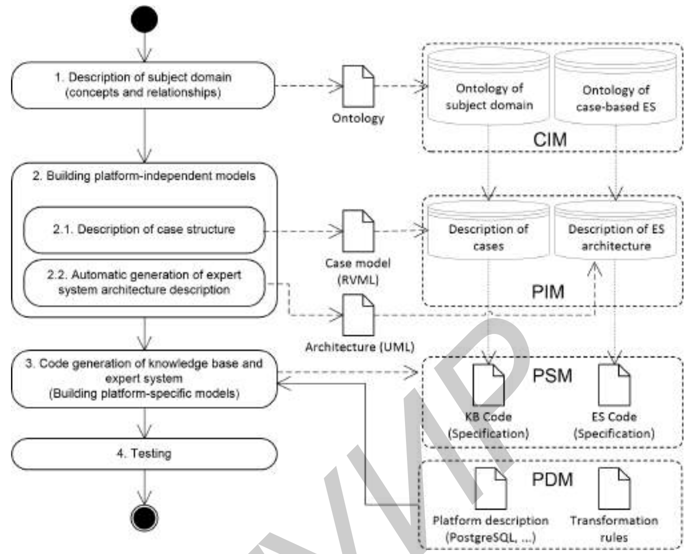

Figure 1. Stages and models of the specialized MDA

In the case of developing intelligent systems, this stage is necessary and corresponds to the stage of the conceptualization of knowledge. This stage allows one to pass from a general conceptual model of a subject domain to a knowledge representation model (with logical rules).

Stage 2. Building a platform-independent model (PIM) that describes the case structure with a clear separation of the "problem description" and "solution" parts. This model is obtained as a result of the automated transformation of a CIM with the subsequent clarification.

Visual modelling is one of the main aspects of the MDA approach. MDA traditionally uses a unified modelling language (UML) for building models. It should be noted that applying the MDA approach to develop specific software requires the use of UML extensions [20] that allow one to take into consideration some features of a subject domain (e.g., telecommunication or health.), architectures (e.g., real-time access and reliability), and programming languages and formalisms (e.g., CORBA or Prolog). So, it is proposed to use RVML as an additional visual modeling tool.

Stage 3. Building a platform-specific model (PSM) that is linked to a specific technological platform, in particular, DBMS.

Stage 4. Generating the codes for a KB and the case-based intelligent decision support system. At this stage, the interpretation of the UML-class diagram (that describes the software architecture) and RVML diagrams is performed. The main results of the interpretation are the program codes and specifications for an interpreter. In the process of interpretation and code generation, the platform description model (PDM) and rules for the transformation of models are used. In this case, a PDM describes the syntax and semantics of the

programming languages for which program code is generated.

Stage 5. Testing. At this stage, the obtained program codes are tested in special software (in the interpreter).

It should be noted that the end user (an expert or a system analytic) only designs a CIM, a PIM and part of a PSM. All of the transformations of the models and the generation of program codes (with the possibility of modifications) are implemented with specialized software that includes a PDM.

The described sequence of stages almost coincides with a "standard" MDA approach, but the stage's content is redefined based on the designs of the case-based intelligent decision support systems.

VI. Conclusion

Efficient creation of KBs for solving problems in various areas requires the development and use of specialized methodical, algorithmic and program means. This paper describes the modification (adaptation) and application of the MDA/MDD approach for the development of the case-based intelligent decision support systems.

The modification includes: the use of the ontology as the CIM, the use of the RVML notation to create the PIM, and the use of XML-like specification as a platform-specific model (PSM) that takes into account the particular case descriptions, including description of fuzzy property values and a membership function.

The proposed approach will be the basis for the development of software.

Acknowledgment

The reported study was partially supported by RFBR (research projects No. 16-37-00122).

References

[1] Gavrilova T.A., Kudryavtsev D.V., Muromtsev D.I. *Knowledge Engineering. Models and methods.* SPb.: Lan, 2016. 324 p. (In Russ.)

[2] Gribova V.V., Kleshchev A.S., Krylov D.A., Moskalenko F.M., Smagin S.V., Timchenko V.A., Tyutyunnik M.B., Shalfeeva E.A. *IACPaaS project. Complex for intelligent systems based on cloud computing* // Artificial Intelligence and Decision Making. 2011. No.1. P.27-35. (In Russ.)

[3] Golenkov V.V., Gulyakin N.A. *Design principles of mass semantic technology component of intelligent systems* // Proceedings of the I International Scientific and Technical Conference - Open Semantic Technologies for Intelligent Systems (OSTIS-2011). – Minsk: BSUIR, 2011. P.21-58. (In Russ.)

[4] Zagorulko Yu.A. *Semantic technology for development of intelligent systems oriented on experts in subject domain* // Ontology of Designing. 2015. Vol.15, No.1. P.30-46. (In Russ.)

[5] Rybina G.V. *Instrumental tools for constructing of dynamic integrated expert system: developing of complex AT-TECHNOLOGY* // Artificial Intelligence and Decision Making. 2010. No.1. P.41-48. (In Russ.)

[6] Czarnecki K., Helsen S. *Feature-based survey of model transformation approaches* // IBM Systems Journal. 2006. Vol.45, No.3. P.621-645.

[7] Stahl T., Voelter M., Czarnecki K. *Model-Driven Software Development: Technology, Engineering, Management*, 1rd ed. John Wiley & Sons, 2006. 446 p.

[8] Frankel D. *Model Driven Architecture: Applying MDA to Enterprise Computing.* New York: Wiley, 2003. 352 p.

[9] *OMG Model Driven Architecture (MDA).* Available at: http://www.omg.org/mda/ (accessed 25.11.2016).

[10] Kleppe A., Warmer J., Bast W. *MDA Explained: The Model-Driven Architecture: Practice and Promise*, 1rd ed. New York: Addison-Wesley Professional, 2003. 192 p.

[11] *Eclipse Modeling Framework (EMF).* Available at: http://www.eclipse.org/modeling/emf/ (accessed 25.11.2016).

[12] *Model Integrated Computing (MIC).* Available at: http://www.isis.vanderbilt.edu/research/MIC (accessed 25.11.2016).

[13] *Meta-Object Facility (MOF).* Available at: http://www.omg.org/mof/ (accessed 25.11.2016).

[14] *Unified Modeling Language (UML) Version 2.5* // OMG Document formal/15-03-01. Available at: http://www.omg.org/spec/UML/2.5/ (accessed 25.11.2016).

[15] *XML Metadata Interchange (XMI).* Available at: http://www.omg.org/spec/XMI/ (accessed 25.11.2016).

[16] *Query/View/Transformation (QVT).* Available at: http://www.omg.org/spec/QVT/ (accessed 25.11.2016).

[17] *MOF Model To Text Transformation Language (MOFM2T).* Available at: http://www.omg.org/spec/MOFM2T/ (accessed 25.11.2016).

[18] *Rule Visual Modeling Language.* Available at: http://www.knowledge-core.ru/index.php?p=rvml (accessed 25.11.2016).

[19] Dorodnykh N.O., Yurin A.Yu. *Using UML class diagrams for design of knowledge bases of rule-base expert systems* // Software Engineering. 2015. No.4. P.3-9. (In Russ.)

[20] De Miguel M., Jourdan J., Salicki S. *Practical experiences in the application of MDA* // LNCS. 2002. Vol.2460. P.128-139.

О СПЕЦИАЛИЗАЦИИ МОДЕЛЬНО-УПРАВЛЯЕМОГО ПОДХОДА ДЛЯ СОЗДАНИЯ ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ ПОДДЕРЖКИ ПРИНЯТИЯ РЕШЕНИЙ ПРЕЦЕДЕНТНОГО ТИПА

Дородных Н.О., Юрин А.Ю.

Рассмотрена специализация (конкретизация) модельно-управляемого подхода в контексте создания интеллектуальных систем поддержки принятия решений прецедентного типа. В частности: уточнены основные этапы процесса построения программных систем (их состав и содержание) и используемые модели. В рамках специализации предложено использовать: онтологии в качестве вычислительно-независимой модели (CIM) и XML-подобную спецификацию в качестве платформо-зависимой модели (PSM). Использование онтологии обеспечивает полное и согласованное описание понятий и отношений предметной области и их визуальное представление в виде графа; а предложенная спецификация учитывает особенности описания прецедентов.