Knowledge sifters in MDA technologies

Yuri Kravchenko, Ilona Kursitys, Victoria Bova

Southern Federal University, 105/42 Bolshaya Sadovaya Str., Rostov-on-Don, 344006, Russia

E-mail: ykravchenko@sfedu.ru, i.kursitys@mail.ru, vvbova@yandex.ru

Abstract. The article considers a new approach to efficient management of information processes on the basis of object models. With the help of special design tools, a generic and application-independent application model is created, and then the program is implemented in a specific development environment. At the same time, the development process is completely based on a model that must contain all the information necessary for programming. The presence of a detailed model provides the automatic creation of typical parts of the application, the development of which is amenable to automation.

1. Introduction

The new approaches for effective management of business processes are based on the idea of automatic development of software applications using the models of business objects. Specific design tools allow us to create the general model of the application regardless of the ways of implementation. The application is implemented with the use of a concrete programming environment. In this case, the development process is based entirely on the model containing all information required for programming. The clear benefits of such approach are [1-3]:

• The independent model allows us to implement software using any programming platform.
• Software implemented in accordance with Model Driven Architecture (MDA) can be easily transferred between different operation systems.
• Cross-platform implementation provides substantial saving of resources.

Model Driven Architecture allows us to automate the programming process to a certain degree. The idea of such model provides automated creation of software typical parts, able to be developed automatically, such as: user interface creation, programming of standard operations, database development and providing data access. The model gives an abstract description of software and hides information about certain aspects to represent simplified descriptions of others. The model captures information in a form suitable for human interpretation and machine processing. For this purpose, the model defines a notation and a metamodel. The notation is a complex of graphic elements, used in the model and understandable by people. The metamodel describes concepts used in the model and captures information in the form of metadata, which can be processed by machine tools.

2. MDA models

Let us consider the main models used in MDA:

Computation Independent Model (CIM) describes general system requirements, conceptual dictionary and operating conditions (environment). The model does not contain any technical details
or structures and system features descriptions. CIM is the most common model independent of the system implementation. MDA specification highlights that CIM should be built to be transformed into a platform independent model. Thus, CIM is recommended to be implemented on the basis of the unified modeling language UML.

Platform Independent Model (PIM) describes contents, structure and features of the system. The model can contain more detailed information, but it does not include any data about system implementation in terms of concrete platforms. The PIM model is based on CIM. The unified modeling language UML is used to build the PIM model.

Platform Specific Model (PSM) describes contents, structure and features of the system related to its implementation in terms of a concrete platform. According to the purpose, the model can be detailed to a greater or lesser degree. The model is built on the basis of two models. The basis of the PSM model is PIM. The platform model is used to improve PSM in accordance with the platform requirements [4].

The development process includes 3 stages. The CIM model is built at the first stage. This kind of model is often called ‘domain’ or ‘business’ model. The purpose of this stage is to develop common system requirements, to create a conceptual dictionary and to describe the system environment. Entities described in CIM at this stage should be carefully analyzed and debugged. Only elements used and evolved at further stages can be included in the model. At this stage, one can use any tool to build the CIM model. However, to provide compatibility with further stages, the UML language is highly recommended to use for the model building. It should be considered that the CIM model at the first stage is represented as a general system’s conception and is not essential for the software development process. In terms of relatively small software systems, this stage can be neglected, but it is quite necessary in the context of complex projects. For instance, developing textual terms of reference does not seem to help the actual programming process, but it significantly contributes to understanding the problem and allows us to avoid further design mistakes.

At the second stage, the PIM model is built. It can be developed from the very beginning if we skipped the first stage or it can be based on the CIM model. Transforming CIM into PIM is implemented with the use of UML descriptions developed at the first stage. At this stage, we add elements describing business logic, contents and common structure of the system, subsystem contents and interaction, function allocation, user interface common description and requirements. The PIM model is always included in all automated application development environments based on MDA.

The PSM models are built at the third stage. The number of platforms for the software defines the number of the PSM models. In some cases, software or its components functions on several platforms simultaneously. The PSM model is created by the PIM model transformation in terms of the platform requirements. The transformation process is described further. According to MDA, the software development process ends at the stage of the PSM model building. It is considered that properly built PSM contains technical information sufficient for source generation (if possible) and for desired software resources.

Let us consider the first stage of MDA development including the CIM model building. The most difficult problems here are creation of common conceptual dictionary and environment description. Solving these problems allows us to create a multidisciplinary knowledge area required for the processes of search and processing dependencies and patterns of knowledge elements which composes the basis of designed system business logic [5-7].

Let us assume ontology as the basis for development of terms and definitions dictionary with the use of semantic similarity estimation between business objects with context. Ontological knowledge structure should include paradigmatic concepts relations, independent of the problem context, and variable syntagmatic concepts relations, appearing in a context of the problem solution. Ontology formalization is implemented with the use of semantic nets assigning relations on the set of knowledge objects. Relations represent the main building elements of the knowledge structure [8].

The main conceptual categories in creating ontological structure are active entities (actors), which realize the knowledge filter functions, passive entities (business objects), which have their own
properties (attributes). Actors are the knowledge filters, representing the active elements (intelligent agents), creating the knowledge area based on searching multidisciplinary relations between different subject areas. Business objects are passive entities containing main basic knowledge having their own attributes and relation system.

The paper suggests building the actors on the basis of Multi-Agent Systems (MAS), which provides organization of effective communities of the intelligent agents. The agents are considered as autonomous subjects, which have intelligence to support deciding their individual or common problems and achieving the desired aims. However, autonomous, responsive and active character of the agents makes software development based on agents more complex in comparison with other methods. Moreover, MAS becomes more difficult to design and to implement in terms of new requirements and interactions of multi-agent environments, such as Semantic Web. Intelligent multi-agent systems demand description tools for procedures of intelligent agents information processing. Web service includes pre-processor that uses the markup language TEMPLET.

The Templet model is a type of the actors model [9], but let us implement actor semantics by means of procedure language rather than special language, such as Erlang. This allows us to use programming systems of available languages. The Templet system implements the idea of model-driven development. This is helpful for automated code transformation in terms of parallel programming. The special feature of the suggested model is combining classic and subject languages, where classic language describes the semantics. The Templet pre-processor can be used in metaprogramming mode [9]: the original program is transformed into another one, which performs a more complex conversion than the pre-processor can provide. For example, deep semantical analysis and code optimization can be implemented in such way.

In addition to classic aspects of MAC, the suggested approach involves new principles to support development of programming agents working in Semantic Web (fig.1). Combination of multi-agent technologies and meta-programming capabilities based on procedure language allows us to provide conversion of business information in a form of “process-object” model implementing transfer from...
CIM to PIM (fig.1). Organizing knowledge processing as a sequence of operations estimating importance of system relations between business objects allows us to create business logic descriptions, which is essential for functions allocation among the elements of the designed system [10, 11].

3. Abstract multi-level ontological architecture
Let us consider the abstract multi-level ontological architecture of semantic relations between actors, subject areas and passive entities (business objects) (figure 2).

![Figure 2. Abstract multi-level ontological architecture of semantic relations between actors, subject areas together and passive entities (business objects)](image)

The paper suggests the power relaxation principle to determine stable multidisciplinary relations. Let us introduce the concepts of multi-level neighborhood rate and multi-level search depth. The multi-level neighborhood rate denotes the local degree of relations between business objects on the first level and between subject areas on the second one. The multi-level search depth denotes a possible number of entities on the first level and a possible number of subject areas on the second one in a sequence of semantic relations. Each relation in the sequence on the second level is weighted, and the weight coefficient takes the value equal to a number of obtained relations between subject areas.

The method includes the following stages. The first stage is to determine a structure element of the second level (a subject area) with the largest value of neighborhood rate and to take it as the "center of attraction". If there are several of them, one is to be selected randomly. Then, depending on the purpose of integration, let us set the search depth value of the second level. It is necessary to search other subject areas related to the "center of attraction", and the number of elements in the sequence cannot exceed the search depth value of the second level. In the obtained subset of functional areas, we determine the relations weights. The maximum weight is selected as the objective value of the selected subset. The researcher can introduce a variable, which value sets the acceptable deviation to the lower side from the objective value of the weight. When the subject areas are determined and included in the subset in accordance with the criterion of weight objective value and acceptable deviation, they are set as the areas to search stable multidisciplinary relations.

The second stage is to determine the first level structure element (an object) with the largest value of neighborhood rate inside of the selected subject areas and to take it as a "center of attraction". If there are several of them, one is to be selected randomly. Then, depending on the purpose of
knowledge integration, let us set the value of the search depth of the first level.

Let us introduce formalized description of the subject area in terms of the suggested method. A set of areas is denoted as $F = \{F_1, F_2, ..., F_z\}$, where $z = 1, m$, $m$ is an integer constant (a number of subject areas), and $F_z = \{O_{z1}, O_{z2}, O_{z3}, ..., O_{zk_z}\}$, $k_z$ is an integer constant (a number of business objects in Fz). If the relations between entities are set in terms of a single subject area, then $\forall i, j \rightarrow i = 1, k_z; j = 1, k_z; i \neq j \exists C_z[i][j] = n$ where $n$ is a number of relations between $O_{zi}$ and $O_{zj}$. $z$ is an integer constant (an order number of the studied area). If all relations between entities of subject areas, including multidisciplinary ones, are assigned, then $\forall a, b \rightarrow a = 1, \sum_{z=1}^{m} k_z; b = 1, \sum_{z=1}^{m} k_z; a \neq b \exists C_{\text{interdisc}}[d][b] = p$, where $p$ is a number of all relations between entities of multi-level ontological architecture [2]. Let us introduce a set of all objects of the architecture as follows:

$$0[d] = \{O_{z1}, O_{z2}, O_{z3}, ..., O_{zk_z}, O_{(z+1)1}, O_{(z+1)2}, ..., O_{(z+1)k_{z+1}}, O_{(z+2)1}, O_{(z+2)2}, ..., O_{(z+2)k_{z+2}}, ..., O_{m1}, O_{m2}, ..., O_{mk_m}\}$$

where $d = \sum_{z=1}^{m} k_z$.

4. Conclusion

The results of search in a subject area are represented as a subset of business objects from different subject areas with stable multidisciplinary relations. Thus, the researcher obtains the formalized procedure of knowledge integration in the conditions of uncertainty.

References

[1] Kherraf S, Lefebvre E, Sury W 2008 Transformation from CIM to PIM using patterns and archetypes. 19th Australian Conference on Software Engineering

[2] Wu J H, Shin S S, Chien J L, Chao W S, Hsieh M C 2007 An extended MDA method for user interface modeling and transformation. The 15th European Conference on Information Systems, pp. 1632–1641

[3] Sharifi H R, Mohsenzadeh M 2012 A new method for generating CIM using business and requirement models. World Comput. Sci. Inf. Technol. J. (WCSIT) 2(1) 8–12

[4] Osis J, Asnina E, Grave A 2007 Computation independent modeling within the MDA. In: IEEE International Conference on Software-Science, Technology and Engineering (SwSTE 2007) pp. 22–34

[5] Xavier B 2005 MDA en Action: Ingénierie Logicielles Dirigée par les Modèles. (Eyrolles, Paris)

[6] Castro V D, Marcos E, Vara J M 2011 Applying CIM-to-PIM model transformations for the service-oriented development of information systems, 2nd Information and Software Technology, pp. 87–105

[7] Hahn C, Dmytro P, Fischer K 2010 A model-driven approach to close the gap between business requirements and agent-based execution, Proceedings of the 4th Workshop on Agent-based Technologies and applications for enterprise interoperability (Toronto, Canada) pp. 13–24

[8] Kravchenko Y A, Kureichik V V 2012 Knowledge management based on multi-agent simulation in informational systems. Conference proceedings. 8th IEEE International Conference Application of Information and Communication Technologies AICT 2014. (Astan, Kazakhstan) pp. 264-267

[9] Bova V V, Kravchenko Y A, Kureichik V V 2015 Decision Support Systems for Knowledge Management. Software Engineering in Intelligent Systems. Proceedings of the 4th Computer Science On-line Conference 2015 (CSOC2015), Vol. 3. (Springer International Publishing AG Switzerland pp. 123-130

[10] Bova V V, Kravchenko Y A, Kureichik V V 2015 Development of Distributed Information Systems: Ontological Approach. Software Engineering in Intelligent Systems. Proceedings of the 4 th Computer Science On-line Conference 2015 (CSOC2015), Vol. 3. (Springer International Publishing AG Switzerland pp. 113-122