ATLANTIS PRESS

Advances in Intelligent Systems Research, volume 166

7th Scientific Conference on Information Technologies for Intelligent Decision Making Support (ITIDS 2019)

Development of Information and Technology Platform for Optimal Design of Heating Systems

Valery Stennikov
Pipeline energy systems department
Melentiev Energy Systems Institute of Siberian Branch of the Russian Academy of Sciences, ESI SB RAS
Irkutsk, Russia
sva@isem.irk.ru

Evgeny Barakhtenko
Pipeline energy systems department
Melentiev Energy Systems Institute of Siberian Branch of the Russian Academy of Sciences, ESI SB RAS
Irkutsk, Russia
barakhtenko@isem.irk.ru

Dmitry Sokolov
Pipeline energy systems department
Melentiev Energy Systems Institute of Siberian Branch of the Russian Academy of Sciences, ESI SB RAS
Irkutsk, Russia
sokolov_dv@isem.irk.ru

***Abstract*—The paper presents a new methodological approach to development of information and technology platform for optimal design of heating systems. The methodological approach consists of the following components: basic principles of platform development; an architecture of the information and technology platform; a technique of automated software construction on the basis of the Model-Driven Engineering conception and modern metaprogramming technologies. The knowledge about heating systems, applied problems, and the applied software is stored in ontologies. The automated construction of the software system is performed on the basis of a heating system model, ontologies and modern metaprogramming technologies. The proposed approach allows us to successfully solve the problem of separation of methods (algorithms) for solving applied problems and mathematical models of heating system elements. To this end, the methods are implemented in the form of software components that are not related to properties and mathematical models of specific equipment. And the models of heating system elements are automatically integrated into the software system. As a result, the software system oriented to solving a specific applied problem is created in an automated mode. The developed approach has been used for the software implementation to design heating systems.**

Keywords—methodological approach; Model-Driven Engineering; ontology; metaprogramming; heating system design

I. INTRODUCTION

The problem of optimal heating system design covers a wide scope of tasks and consists in search for an optimal direction in change of system structure and parameters, determination and prevention of "bottlenecks", replacement of outdated technologies and equipment with new energy efficient ones, provision of reliable heat supply and system controllability subject to fulfillment of physical and technical conditions of system operation and constraints on operating parameters [6]. The existing approaches to solving this problem design their algorithms from different sets of subtasks (optimization of structure, optimization of parameters, analysis of system reliability, thermal and hydraulic calculations, calculation of heat source parameters and others) that are specific for a set of considered heating systems subject to their real features. As a rule, this is a complex iterative computation process, during which the subtasks for different heating systems may be solved in a varying sequence and by different methods depending on the stated goal.

Large sizes of heating systems and high computational complexity of the applied methods, algorithms and mathematical models make it impossible to solve problems of optimal heating system design without specialized software. Therefore it is necessary to work out new principles and approaches to software development that will automatically create complex software systems intended for solving applied problems by reference to inherent characteristics of modeled heating systems. In these approaches the knowledge on the subject domain must be formalized in the form of ontologies, which will allow their reuse in automatic construction of applied software systems. The automatic construction of a software system aimed at solving a particular applied problem is possible on the basis of the unified information and technology platform to be developed. The paper presents a new methodological approach to construction of this platform.

The existing approaches to software development are presented in [1-3]. The principles of the development of universal software components are considered in [1]. In [4,5], the focus is made on general principles of developing complex software systems. The authors of [6-10] propose a Model-Driven Engineering (MDE) conception to automate the stages of software development. The MDE conception represents a set of methodological approaches to the automated construction of complex software systems based on the preliminarily constructed models. In [11-13], the UML language is proposed as a universal tool for description of subject domains.

The authors of [10,17,18] consider the approaches intended to automate the software development stages based on metaprogramming. Metaprogramming represents a programming technique capable of creating the programs that generate other programs when running, or the programs that change themselves during execution. In Russia, the fundamental research in this area was performed by A.P. Ershov. In 1958 he published his monograph "Programming Programme for the BESM Computer" [19]. Reflective programming is one of the metaprogramming types. It represents an extension of the object-oriented programming paradigm [17,18,20].

Currently, there are approaches that suggest the use of ontologies as a tool for description of the subject domain [16-18]. The automatic construction of a software system involves ontologies that make it possible to formalize the description of objects of a subject domain, their properties, and interactions

between these objects. The ontologies are getting increasingly wider application in engineering [21]. Some researchers describe the experience of applying ontologies to solve power engineering problems [22, 23]. General issues related to the development and use of ontologies are discussed in [24, 25].

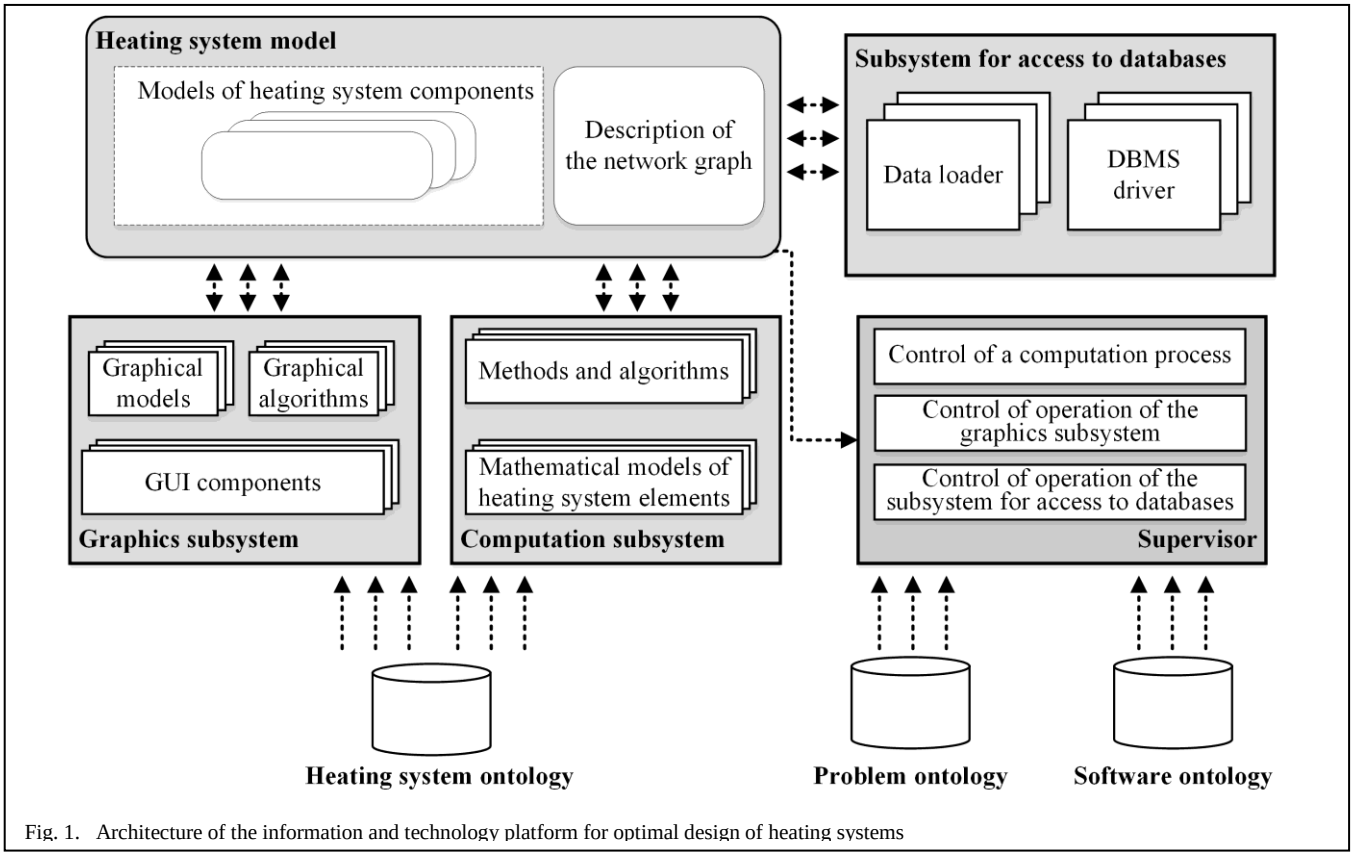

Fig. 1. Architecture of the information and technology platform for optimal design of heating systems

This paper describes a developed methodological approach to construction of the information and technology platform for optimal design of heating systems. The approach is based on the MDE conception. The automated software construction rests upon the up-to-date technologies of metaprogramming and ontology which allow a formalized description of objects of the subject domain, their properties and interrelations between these objects [26-28].

Based on the literature review and the analysis of the existing approaches to software implementation, we can conclude that the issues addressed in this paper are relevant.

II. Components of The Methodological Approach

We propose a methodological approach to development of the information and technology platform for optimal design of heating systems. The approach consists of the following components:

- basic principles of the platform construction;
- an architecture of the information and technology platform;
- a technique of automated software development based on the MDE conception and metaprogramming.

The principles of the platform development are the following:

- A complex of problems should be solved by the unified software tool that integrates the whole complex of problems on modeling, calculation and optimization of heating systems into the unified information environment.
- Software should be developed on the basis of the current paradigms of object-oriented and component programming which will allow the reuse of methods, algorithms and mathematical models for heating system elements.
- Software components should be arranged in the form of libraries and possess standardized interfaces.
- Software should be implemented on the basis of advanced information technologies and approaches to programming.

III. Platform Architecture

We have devised an original approach to architecture development of the information and technology platform for optimal design of heating systems. The platform architecture includes the following components:

- platform structure and composition of its subsystems;
- technologies and software tools for platform implementation;
- basic structure of the library of software components, their interfaces and mechanisms of integration into the platform.

The suggested architecture of the information and technology platform for optimal design of heating systems is presented in Fig. 1.

The information and technology platform for optimal design of heating systems comprises the following components.

Graphics subsystem. This subsystem is intended for work with graphical representations of heating system schemes and data on scheme elements on a site plan. The subsystem is responsible for user work with an active multilevel heating system model on a site plan, and allows him to browse data in a convenient form and make desired modifications. The graphics subsystem includes the software components that implement user-system interaction, and software components of models that implement graphical models of heating system elements. During work with a particular heating system the graphical software components of models are dynamically connected to the software system by the reflective programming (metaprogramming). The connection process is controlled by the knowledge stored in the heating system ontology and problem ontology [29,30].

Computation subsystem. This subsystem is intended for solving computational tasks and consists of the library of software components. The library includes the following components:

- universal software components that implement methods and algorithms for solving applied problems;
- software components that implement models of heating system elements.

Supervisor. This device performs the following functions:

- loads software components for computer modeling, calculation and optimization of heating systems using the metaprogramming;
- controls operation of the graphics subsystem;
- controls a computation process, when a sequence of calls of the required software components is formed;
- controls operation of the subsystem for access to databases.

The supervisor uses the knowledge stored in the problem and software ontologies.

Subsystem for access to databases. This subsystem is designed for storage and reuse of heating system model, initial data and calculation results, and graphical data on urban development.

The authors develop methodological approaches and implement software on the basis of the open source and free components. Java is chosen as a core language owing to the following advantages: 1) support of advanced technologies of object-oriented, component and functional programming; 2) built-in support of metaprogramming technologies; 3) wide range of technologies and tools for distributed and parallel calculations. The XML language is used as a tool for formal representation of ontologies. This language was used to develop the domain-oriented language. MathML is applied for storage of mathematical models of the heating system components.

We propose an original approach to design of interfaces for software components, whose function is standardization of their interfaces using design patterns. Depending on the functions performed, the platform components can be divided into groups. Interfaces of interaction between components, including the rules of their call, transfer of parameters and calculation results, are developed for each group of components. The groups of components are the following:

- computation components containing the software implementation of algorithms for mathematical modeling and optimization of energy systems;
- system components intended for data preparation and examination (interaction with the database, control of input data correctness, etc.);
- controlling components intended to arrange a sequence of calls of computation and system components.

IV. Technique For Automated Software Development

Based on the conducted research, we propose a new technique for automated software development based on the MDE conception and metaprogramming [30]. Let us present the main principles of this technique:

- A computer model of a specific heating system, models of heating system components (i.e. individual subsystems) and methods (algorithms) are integrated only in the context of an applied problem.
- Software components depending on the properties of the modeled heating system (a set of models of heating system components) are created automatically using metaprogramming based on the computer model of this heating system and ontologies that contain the description of equipment and applied problems to be solved.
- The software intended for modeling a certain heating system is developed in the context of an applied problem with the help of metaprogramming, based on automatically constructed software models of heating system components, and components implementing the methods and algorithms for solving the applied problems. The process of software construction is controlled by the knowledge stored in ontologies: a heating system ontology, a problem ontology and a software ontology [29].

The heating system ontology consists of: a description of the hierarchical structure of heating system, classification of the equipment used in the heating networks and its technical characteristics, a description of heating system components and their parameters (technical characteristics, hydraulic parameters and boundary conditions), and classification and description of the applied mathematical models (for example, equations defining the laws of head loss in the network branches, formulas for calculation of resistance, etc.).

The problem ontology contains a description of the applied problems (for example, optimization of multiloop network parameters) and the methods for their solution (for example, the multiloop optimization method), a description of algorithms, enumeration of parameters that represent initial data and parameters obtained by solving the problem.

The software ontology is intended to store the knowledge necessary to automate the construction and use the software. This ontology contains a description of software components and their properties, metadata (input and output parameters, description of data formats), a description of technologies and interfaces for access to the software components.

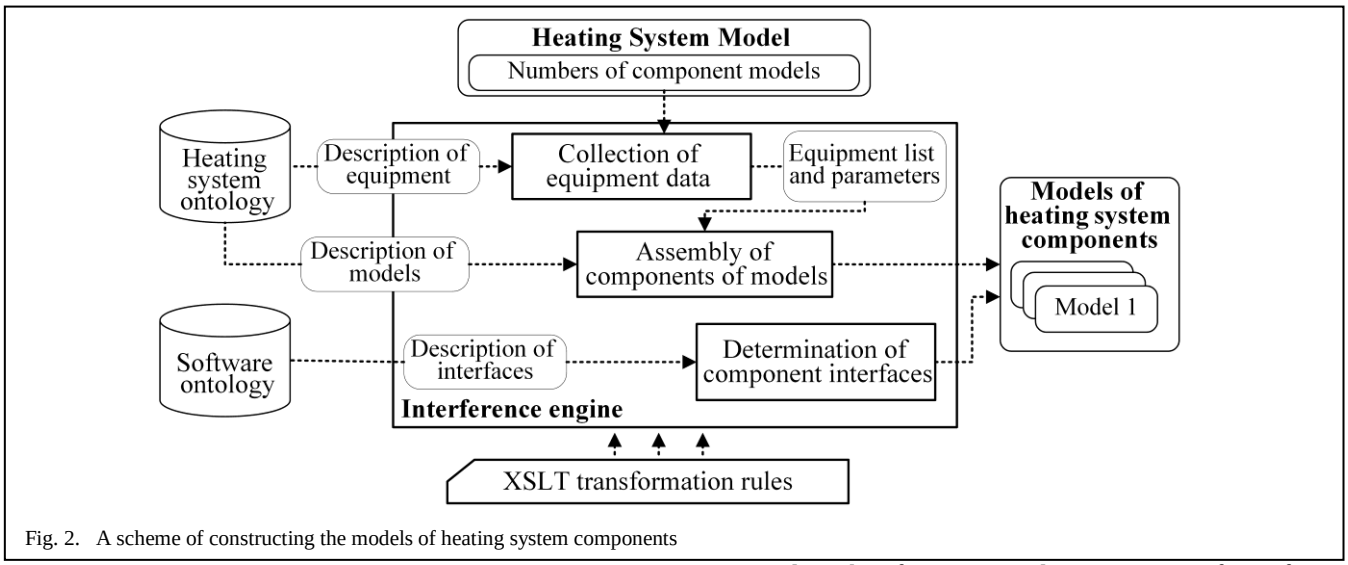

Fig. 2. A scheme of constructing the models of heating system components

The technique for automated software development includes four stages.

Stage 1. Construction of a heating system model. At this stage, an engineer constructs a hierarchical model of a specific heating system. This model represents a directed graph with its vertices corresponding to nodes (sources, consumers, connection nodes) and edges corresponding to branches (passive branches, active branches with pumping stations). For each component of the model, there is information on the installed equipment, its characteristics and methods of construction. The obtained model is stored in a database for repeated application. A graphical schematic editor is used to construct the models of real heating systems consisting of thousands of components. This editor allows construction and use of calculated schemes of heating systems on a site plan.

Stage 2. Formal description of an applied problem. At this stage an engineer formally describes an applied problem. This implies setting such conditions and constraints as: 1) a set of standard pipelines and their characteristics; 2) constraints on pipeline construction (reconstruction) technologies; 3) constraints on pressure at nodes; 4) constraints on speed of heat carrier flow; 5) design values of heat carrier sinks for consumers and heat carrier sources for sources; 6) a set of standard pumping stations and their characteristics.

Stage 3. Automatic construction of a software model. This stage suggests the creation of a software model intended for solving the applied problem. This model represents an aggregate of data structures that reflect the following software properties: a structure and composition of software components, a description of a computation process in the form of an oriented graph, a description of mathematical models of equipment, a list of applied methods and algorithms. The software model is constructed by an inference engine which, for its operation, applies a heating system model, formal description of the problem and ontologies. The inference rules implemented in the XSLT and XPath languages are applied for logical inference from the ontologies represented in XML.

An algorithm for automated construction of a software model includes the following steps.

Step 1. A list of equipment used to construct a heating system and technologies of its construction is compiled based on the heating system model.

Step 2. A list of new equipment allowed to be installed for network reconstruction and technologies of its installation is compiled based on the formal description of the problem.

Step 3. Based on the heating system ontology, the inference engine constructs the data structures that contain a description of heating system component models necessary to solve the problem based on the lists of equipment and technologies of its installation that were compiled at the previous steps.

Step 4. Based on the heating system model, formal description of the applied problem and problem ontology, the specific methods are chosen to solve the problems, a list of subproblems is compiled, and the methods and algorithms are chosen.

Step 5. An oriented graph defining the computation process is constructed to solve the problem based on the problem ontology. The nodes of this graph correspond to the steps of problem solution, and edges - to the links between them.

Step 6. Based on the list of methods and algorithms, the inference engine using the software ontology creates a list of software components necessary to solve the problem.

Step 7. The data structures describing the relations “problem – method – component” are formed.

Step 8. Based on the description of components (implementing the methods and algorithms), and software ontology that was given at the previous steps, the component interfaces implementing mathematical models of heating system components are determined.

Step 9. Based on the problem and software ontologies, the data structures describing input and output parameters for each

software component, types of data and ways of their transfer are formed.

The construction process of the data structures describing models of heating system components is presented in Fig.2.

Stage 4. Automatic software construction based on its model

At this stage, the software is automatically developed based on its model with the aid of metaprogramming techniques. The following subproblems are solved in the course of its development.

Subproblem 1. Formation of a set of software components implementing mathematical models of heating system components. Below is an algorithm of constructing a set of software models of heating system components. The algorithm consists of three steps.

Step 1. The development control component calls the XSLT-processor to construct models of heating system components, necessary to solve an applied problem. The XSLT processor, based on a list of equipment from the software model, mathematical models in MathML format and transformation rules in XSLT format, creates a set of data structures that contain a description of models of heating system components.

Step 2. The development control component calls the code builder which, based on the data structures defining the models of heating system components, and the description of interfaces of software components from the software model, builds a software code in the Java language.

Step 3. The development control component calls the Java compiler which compiles a software code built at the previous stage to construct a set of necessary models.

Subproblem 2. Loading of software components that implement the methods and mathematical models in the memory by the tools of reflective programming. These components are integrated into a unified software system using the standardized interfaces ensured by design patterns [1].

Reflective programming makes it possible to perform operations that cannot be performed within the object-oriented programming. Some of the most important operations are: examination of classes during the program run, determination of their attributes, methods and constructors; development of new copies of objects based on the class name, with the use of constructors; assignment and calculation of values of attributes by their names; call of methods by their name and description of arguments; flexible work with arrays and containers (collections). Reflective programming allows inspecting the software system structure, and dynamically changes the set of software components during its operation.

The system loader that implements the Factory design pattern does the following: 1) it receives a description of software components from the software model; 2) finds the components that correspond to the problem to be solved; 3) prepares data structures necessary for their call; 4) loads the components in the memory using the reflective tools of the Java language; 5) transfers references to the components to the software integration environment.

Subproblem 3. Filling of the hash tables with pairs "number – reference". These hash tables are used by software components represented by methods that receive references to the relevant software components represented by models, according to the numbers of mathematical models of components included in the heating system model.

V. Results

The software obtained as a result of automated development according to the proposed methodology consists of three architectural layers: 1) a subsystem of computation control (supervisor) that contains the components controlling the computation process; 2) a computation subsystem that solves an applied problem using the software components that implement methods, algorithms and models; 3) a subsystem of data storage that provides data exchange between database and local memory.

The developed SOSNA software was successfully applied at Melentiev Energy Systems Institute of Siberian Branch of the Russian Academy of Sciences to determine optimal parameters of heating systems when solving the problems of heating system design, expansion and reconstruction. The proposed methodological approach to automated software development enables its flexible application in modeling of the heating systems with a large set of equipment.

The SOSNA software was used to perform multivariate calculations that made it possible to determine the optimal parameters and make recommendations on rational reconstruction and expansion of the heating systems in the Central and Admiralteisky districts of Saint-Petersburg, the town of Bratsk and Magistralny urban-type settlement and others. The proposed recommendations on the heating system reconstruction ensure their adaptation to the growing heat loads and foster their optimal operation.

VI. Conclusions

The paper presents a new methodological approach to construction of the unified information and technology platform that will automatically develop a software system for solving a specific applied problem of optimal heating system design. The suggested approach includes basic principles of platform construction, an architecture of the information and technology platform, a technique for automated software development on the basis of the MDE conception and metaprogramming. The platform-based software can be used by research, design and operation organizations dealing with heat supply. It makes it possible to generate reconstruction and expansion recommendations which can enhance heating system efficiency and provide the required quality of heat supply to consumers.

Acknowledgment

The research was performed at Melentiev Energy Systems Institute of Siberian Branch of the Russian Academy of Sciences under the support of Russian Science Foundation (Grant No. 17-19-01209).

References

[1] R.C. Martin, Agile Software Development: Principles, Patterns and Practices. New York: Pearson Education, 2002.

[2] K. Beck, Extreme Programming Explained: Embrace Change. Boston: Addison-Wesley, 1999.

[3] G. Booch, Object-Oriented Analysis and Design with Applications. Boston: Addison-Wesley, 2007.

[4] M. Fowler, D. Rice, M. Foemmel, E. Hieatt, R. Mee, and R. Stafford, Patterns of Enterprise Application Architecture. Boston: Addison-Wesley, 2002.

[5] M. Fowler, and R. Parsons, Domain-Specific Languages. Boston: Addison-Wesley, 2010.

[6] D.C. Schmidt, "Guest Editor's Introduction: Model-Driven Engineering," Computer, vol. 39, no. 2, pp. 25-31, Feb. 2006.

[7] M. Volter, T. Stahl, J. Bettin, A. Haase, and S. Helsen, Model-Driven Software Development: Technology, Engineering, Management. New York: Wiley, 2006.

[8] M. Brambilla, J. Cabot, and M. Wimmer, Model Driven Software Engineering in Practice. Synthesis Lectures on Software Engineering. San Rafael: Morgan & Claypool, 2012.

[9] A.R. Silva, "Model-driven engineering: A survey supported by the unified conceptual model," Computer Languages, Systems & Structures, vol. 43, pp. 139-155, 2015.

[10] V. Štuikys, and R. Damaševičius, Meta-Programming and Model-Driven Meta-Program Development. London: Springer-Verlag, 2013.

[11] G. Booch, I. Jacobson, and J. Rumbaugh, The Unified Software Development Process. Upper Saddle River, New Jersey: Prentice Hall, 1999.

[12] G. Booch, J. Rumbaugh, and I. Jacobson, The Unified Modeling Language User Guide, 2nd edn. Boston: Addison-Wesley, 2005.

[13] C. Larman, Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development, 3rd edn. Upper Saddle River, New Jersey: Prentice Hall, 2004.

[14] N.M. Goldman, Ontology-Oriented Programming: Static Typing for the Inconsistent Programmer, The Semantic Web - ISWC 2003. ISWC 2003. Lecture Notes in Computer Science, vol. 2870, D. Fensel, K. Sycara, J. Mylopoulos, Eds. Berlin, Heidelberg: Springer, 2003

[15] J.Z. Pan, S. Staab, U. Aßmann, J. Ebert, and Y. Zhao, Ontology-Driven Software Development. Berlin: Springer-Verlag, 2013.

[16] H. Paulheim, Ontology-based Application Integration. New York: Springer-Verlag, 2011.

[17] K. Hazzard, and J. Bock, Metaprogramming in .NET. Greenwich: Manning Publications, 2012.

[18] I. Forman, and N. Forman, Java computection in Action. Greenwich: Manning Publications, 2005.

[19] A.P. Ershov, Programming Programme for the BESM Computer, London: Pergamon Press, 1959.

[20] B.C. Smith, Procedural Reflection in Programming Languages, PhD Thesis. Massachusetts: MIT, 1982.

[21] G. B. Evgenev, Intelligent Design Systems, Moscow: Bauman MSTU, 2009. [in Russian].

[22] Y.A. Zagorulko, and G.B. Zagorulko, "Ontological approach to the development of the decision support system at the oil and gas producing enterprise," Vestnik NGU, vol. 10, 2012, pp. 121-129. [in Russian].

[23] T.N. Vorozhtsova, and S. K. Skripkin, "Using ontologies in modeling of software," Vychislitel'nye tekhnologii, vol. 13, 2008, pp. 376-381. [in Russian].

[24] T.R. Gruber, What is an Ontology?, Stanford University, Nov. 9, 2009. [Online], http://www-ksl.stanford.edu/kst/what-is-an-ontology.html.

[25] N. Guarino, Understanding, Building, and Using Ontologies. Nov. 1, 2010. [Online]. http://ksi.cpsc.ucalgary.ca/KAW/KAW96/guarino/guarino.html.

[26] T.R. Gruber, "A translation approach to portable ontology specifications," Knowledge Acquisition, vol. 5, no. 2, pp. 199-220, 1993.

[27] S. Staab, and R. Studer, Handbook on Ontologies, 2nd edn. Heidelberg: Springer-Verlag, 2009.

[28] S. Staab, T. Walter, G. Gröner, and F.S. Parreiras, "Model Driven Engineering with Ontology Technologies," in Reasoning Web. Semantic Technologies for Software Engineering. Lecture Notes in Computer Science, vol. 6325, U. Aßmann, A. Bartho, C. Wende, Eds. Berlin, Heidelberg: Springer, 2010, pp. 62-98.

[29] V. Stennikov, E. Barakhtenko, and D. Sokolov, "The Use of Ontologies in the Integrated Graphical Environment," Advances in Intelligent Systems Research, vol. 158, pp. 152–157, 2018 [Proc. of Vth International workshop "Critical infrastructures: Contingency management, Intelligent, Agent-based, Cloud computing and Cyber security" (IWCI 2018)].

[30] V. Stennikov, E. Barakhtenko, and D. Sokolov, "Automation of the Integrated Graphical Environment Construction," Advances in Intelligent Systems Research, vol. 158, pp. 152–157, 2018 [Proc. of Vth International workshop "Critical infrastructures: Contingency management, Intelligent, Agent-based, Cloud computing and Cyber security" (IWCI 2018)].