Automation of the Integrated Graphical Environment Construction

Valery A. Stennikov [1], Evgeny A. Barakhtenko [2], Dmitry V. Sokolov [3]

Melentiev Energy Systems Institute of Siberian Branch of the Russian Academy of Sciences,
Lermontov str., 130
Irkutsk, Russia
[1]E-mail: sva@isem.irk.ru

[2]E-mail: barakhtenko@isem.irk.ru

[3]E-mail: sokolov_dv@isem.irk.ru

Abstract

The paper presents a methodological approach to automated construction of integrated graphical environment for computer modeling of pipeline systems of various types. Automated construction of the integrated environment is performed using a computer model of the pipeline system and ontologies based on the Model Driven Engineering concept and reflective programming.

Keywords: Integrated graphical environment, automation of programming, Model-Driven Engineering, metaprogramming, reflective programming, ontology.

1. Introduction

It is impossible to effectively solve the problems of design and operation of energy pipeline systems (heat, water, oil, gas, etc.) without an integrated graphical environment. This environment represents a software system designed to solve information (work with a computer pipeline system model and its elements on a plan of the territory in an interactive mode), calculation (solving engineering problems) and analytical (overviews of graphs, tables and calculation results) problems within a single user interface.

The commonality of the topological properties of pipeline systems makes it possible to develop a software system being an integrated graphical environment for computer modeling of pipeline systems of different types. The design and operation organizations apply specialized software oriented to a certain type of pipeline systems and certain classes of problems to be solved. Currently, there is no description of methodological approaches to the construction of integrated graphical environments that are universal and capable to model a variety of pipeline system types within one software system.

It is highly difficult to maintain an actual state of the whole set of required software components, therefore, we can conclude that the implementation of the integrated graphical environment requires the involvement of programming paradigms oriented to the automation of software design stages.

The existing approaches to the development of software 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 [6-8] 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 design of complex software systems based on the preliminarily developed models [6]. In [9-11], the UML language is proposed as a universal tool for description of subject domains. Currently, there are approaches that suggest the use of ontologies as a tool for description of the subject domain [12-14]. The authors of [8,15,16] consider the approaches intended to automate the software development stages based on metaprogramming. Reflective programming is one of the metaprogramming types. It represents an extension of the object-oriented programming paradigm [15-17]. 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 change the set of software components during its operation.

The paper proposes a methodological approach to automated construction of an integrated graphical environment for computer modeling of pipeline systems. Based on this approach the software system is implemented. This system represents a software prototype of the integrated graphical environment for computer modeling of pipeline systems of different types.

2. A methodological approach and its specific feature

Based on the research results, we propose a methodological approach intended to integrate the MDE conception and reflective programming. In the proposed approach, the MDE conception is adapted to specific features of the subject domain of computer modeling of pipeline systems, while automated design of a software system is performed based on the computer model of pipeline systems and ontologies [18-20]. The construction of the integrated graphical environment involves three applied ontologies: a pipeline system ontology, a problem ontology, and a software ontology. From the perspective of long-term storage of ontologies, the preference is given to XML format, which is a universal tool for data exchange between software systems. Moreover, it offers an opportunity to store complex (hierarchical, network) data structures, and modern programming platforms have libraries designed to work with this format.

The proposed methodological approach includes:

- The principles of automated development of an integrated graphical environment for computer modeling of pipeline systems;
- An architecture of the integrated graphical environment, describing the arrangement of the software system developed in an automated mode;
- A technique for automated construction of the software system based on the MDE conception, reflective programming, ontologies and a computer model of pipeline system.

The main principles underlying the methodological approach are:

- A software system oriented to modeling of a certain pipeline system is designed in an automated mode in the context of constructing a computer model of this pipeline system (Fig.1).

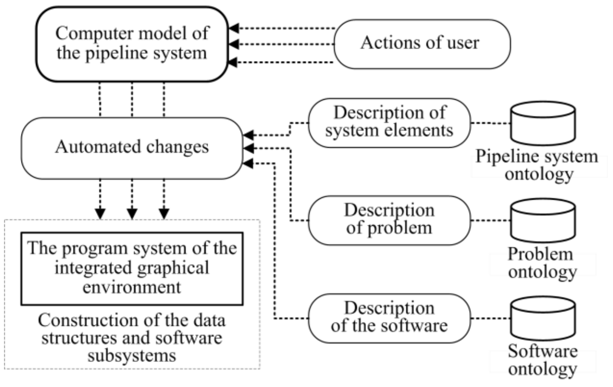
Computer model of the pipeline system
Actions of user
Automated changes
Description of system elements
Pipeline system ontology
Description of problem
Problem ontology
Description of the software
Software ontology
The program system of the integrated graphical environment
Construction of the data structures and software subsystems

Fig. 1. Illustration of the design principle of the integrated graphical environment.

- Automated design of a software system and its adaptation to certain features of the pipeline system to be modeled is performed by reflective programming.
- The knowledge describing the properties, problems and software for their solving that are common for all types of pipeline systems is stored in the form of ontologies, and specific features of the modeled pipeline system are described by its computer model.

We propose an architecture of the integrated graphical environment (Fig.2) which includes the following components:

- A graphical subsystem that ensures the work of the user with active graphical model of pipeline system on the location plan and allows the user to look through the data in a user-friendly form and make necessary changes.
- An instrumental subsystem that ensures solving the following problems: create a necessary graphical menu form that allows the user to solve the problems that correspond to the type of a modeled pipeline system; creates panels of instruments for drawing, editing, changing the states of elements of the active model of the pipeline system and makes them available for the user; forms dialog windows for the adjustment of the pipeline system mapping, and input of the necessary information and parameters, that affect solving the applied problems; creates panels of instruments for drawing the elements of location plan and urban development.
- A subsystem of access to databases that organizes the work with different databases applied for

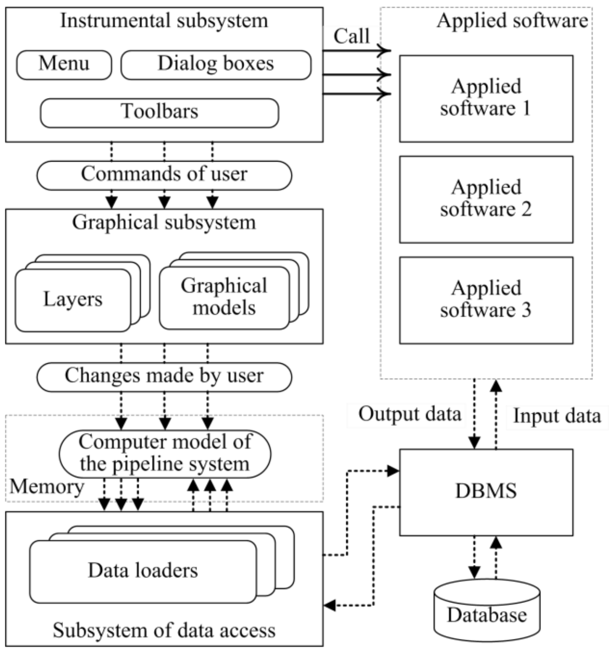

Fig. 2. Architecture of the integrated graphical environment.

storage and repeated use of the computer models of pipeline systems, input data and results of calculations, data on the facilities of urban development.

The proposed architecture is a description of the software system arrangement, which is created in an automated mode and implements the integrated graphical environment for computer modeling of pipeline systems. The implementation of the integrated graphical environment according to the proposed methodological approach.

The basic programming language is Java, because it supports modern technologies for object-oriented, component and functional programming. Moreover, it has embedded support to reflective programming and other types of metaprogramming [16]. The database-management system (DBMS) Firebird is used to work with the database. This system enables a multi-user access and compatibility with various operating systems.

3. Technique of automated construction

In accordance with the suggested technique the automated construction of the graphic environment for computer modeling of energy pipeline systems is performed in the following three stages:

(i) Construction of the instrumental subsystem.
(ii) Construction of the subsystem of access to databases.
(iii) Construction of the graphical subsystem.

The environment is adjusted to the pipeline system type selected by the user at the first and second stages, and to the work with specific features of the modeled pipeline system at the third stage.

Consider contents of the stages in greater detail.

Stage 1. Construction of the instrumental subsystem. The graphical interface elements and the required data structures are created on the basis of description of the pipeline system type corresponding to the user choice.

The instrumental subsystem at the first stage is constructed in accordance with the following algorithm.

(i) Generation of data structures that describe a set of elements of the pipeline system of a particular type and their properties.
(ii) Creation of main menu of the graphical environment that allows the user to solve problems corresponding to the type of the modeled pipeline system.
(iii) Loading of graphical images in the memory to create elements of the tool bars based on description of pipeline system elements in the pipeline system ontology.
(iv) Creation of tool bars containing elements of the graphical interface to work with the computer model of a pipeline system.
(v) Generation of data structures, which describe a list of possible equipment to be installed in the pipeline system and a set of models (graphical and mathematical).
(vi) Generation of data structures, which describe a set of applied problems to be solved and software used for their solution.
(vii) Generation of data structures, which describe a list of input and output parameters for the applied problems to be solved.

Stage 2. Construction of the subsystem of access to databases. The computer models of pipeline systems of different types are stored in the database, whose set of objects corresponds to the set of pipeline system elements, their properties, the set of used equipment and its characteristics. These models are accessed through the specialized software components intended for a concrete type of pipeline systems. Involvement of these components is performed in an automated mode, during which description of the software components corresponding to the modeled system type is read from the software ontology. Further, these components are integrated in the software system by the reflective programming tools based on their description.

Stage 3. Construction of the graphical subsystem. The main idea suggested by us is that the graphical subsystem responsible for display of the active pipeline system model is constructed dynamically (in an automated mode) in response to the user actions on making changes in the computer model of the pipeline system. If the user creates a new model of the pipeline system of a certain type, the integrated graphical environment creates a basic set of models of elements, which correspond to the type of the modeled pipeline system. When the user loads an earlier created model of the pipeline system, the environment includes a set of models of elements, which corresponds to a set of elements of the modeled pipeline system in addition to the basic set of models of pipeline system elements. The model of an element (component-model) is a software component realizing a graphical display and a set of algorithms to contribute to activity of the pipeline system model (for example, processing of user actions by the mouse, test on membership of the point to the element domain, transfer, scaling, etc.). The component-model connected to the software system one day can be used repeatedly to represent pipeline system elements, whose type and state correspond to this model.

4. Implementation

The developed methodological approach is applied to implement a software system, which is a software prototype of the integrated graphical environment for computer modeling of pipeline systems. The software prototype is intended for work with computer models of heat supply systems and enables to construct, make changes and store them in the database for repeated use in solving the applied problems.

The subsystems of the integrated graphical environment responsible for creation of graphical interface tools (instrumental subsystem), interaction with the database (subsystem of the access to databases) and work with the active model of pipeline systems (graphical subsystem) are designed in an automated mode in terms of specific features of a concrete type of the modeled pipeline system. The subsystems are constructed in accordance with the algorithm presented in the paper.

Fig. 3 presents a type of the main window of the integrated graphical environment, which contains interface elements formed by the instrumental and graphical subsystems. The instrumental subsystem supports work of the following elements of the main window of the integrated graphical environment:

- the main window, which allows the user to perform the following operations: to work with database files; control the display of the pipeline system model; control the type of the working window area; solve the applied problems of pipeline system modeling;

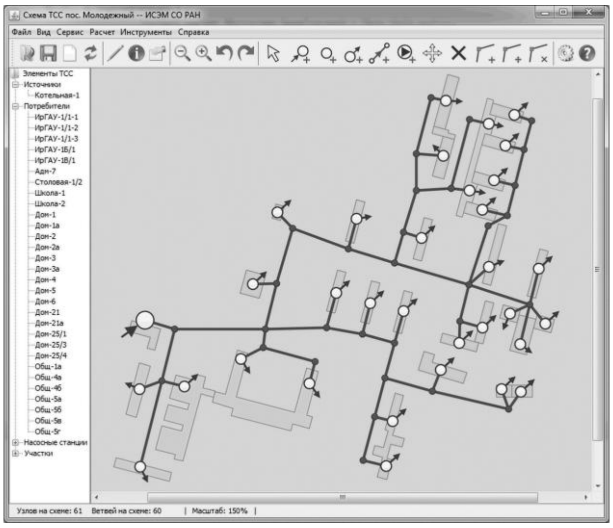

Fig. 3. Type of the graphical interface of integrated environment.

- the tool bars, which allow the user to fulfill the following functions: work with the database file; control of the display of the pipeline system model; control of the type of working window area; work with elements of the active graphical model of the pipeline system; work with the objects of the urban area on the site plan;
- the tree of the modeled pipeline system scheme to work with its elements;
- the line of the state, which displays basic information on the process of pipeline system modeling.

The central part of the main window is the working area responsible for user work with the active graphical model of the pipeline system and objects of the urban area on the site plan. The graphical subsystem, which is dynamically adapted to the next user operations is responsible for implementation of this part of the user interface:

- loading of the data describing the earlier created model of the pipeline system;
- construction of a new pipeline system model when choosing its type by the user;
- changes made by the user in the computer software of the pipeline system.

5. Conclusion

The paper presents an approach to automated construction of the integrated graphical environment for computer modeling of pipeline systems of different types. Distinction of this approach consists in implementation of the following ideas:

- The computer model of the pipeline system of a certain type can be represented as a combination of the graph describing configuration of this system and a set of software component-models describing properties of its elements.

- The automated construction of the software system, which implements the integrated graphical environment, is performed on the basis of the computer model of the pipeline system and the ontologies on the basis of the Model-Driven Engineering conception and the reflective programming.

The software system, which is a software prototype of the integrated graphical environment for computer modeling of energy pipeline systems of different types and purposes, is designed.

The suggested methodological approach is of universal character and can be applied in institutions engaged in software development for computer modeling of energy pipeline systems of different types and purposes.

Acknowledgements

This work is supported by the Russian Foundation for Basic Research (grant No.16-07-00948).

References

1. R. C. Martin, *Agile Software Development: Principles, Patterns and Practices* (Pearson Education, New York, 2002).
2. K. Beck, *Extreme Programming Explained: Embrace Change* (Addison-Wesley, Boston, 1999).
3. G. Booch, *Object-Oriented Analysis and Design with Applications* (Addison-Wesley, Boston, 2007).
4. M. Fowler, D. Rice, M. Foemmel, E. Hieatt, R. Mee and R. Stafford, *Patterns of Enterprise Application Architecture* (Addison-Wesley, Boston, 2002).
5. M. Fowler and R. Parsons, *Domain-Specific Languages* (Addison-Wesley, Boston, 2010).
6. M. Volter, T. Stahl, J. Bettin, A. Haase and S. Helsen, Model-Driven Software Development: Technology, Engineering, Management (Wiley, New York, 2006).
7. M. Brambilla, J. Cabot and M. Wimmer, *Model Driven Software Engineering in Practice. Synthesis Lectures on Software Engineering* (Morgan & Claypool, San Rafael, 2012).
8. V. Štuikys and R. Damaševičius, *Meta-Programming and Model-Driven Meta-Program Development* (Springer-Verlag, London, 2013).
9. G. Booch, I. Jacobson and J. Rumbaugh, The Unified Software Development Process (Prentice Hall, Upper Saddle River, New Jersey, 1999).
10. G. Booch, J. Rumbaugh and I. Jacobson, *The Unified Modeling Language User Guide*, 2nd edn. (Addison-Wesley, Boston, 2005).
11. C. Larman, *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development*, 3rd edn. (Prentice Hall, Upper Saddle River, New Jersey, 2004).
12. N. M. Goldman, Ontology-oriented programming: Static typing for the inconsistent programmer, in *2nd Int. Conf. Semantic Web* (Sanibel Island, Florida, 2003).
13. J. Z. Pan, S. Staab, U. Aßmann, J. Ebert and Y. Zhao, *Ontology-Driven Software Development* (Springer-Verlag, Berlin, 2013).
14. H. Paulheim, *Ontology-based Application Integration* (Springer-Verlag, New York, 2011).
15. K. Hazzard and J. Bock, *Metaprogramming in .NET* (Manning Publications, Greenwich, 2012).
16. I. Forman and N. Forman, *Java Reflection in Action* (Manning Publications, Greenwich, 2005).
17. B. C. Smith, *Procedural Reflection in Programming Languages*, PhD Thesis (MIT, Massachusetts, 1982).
18. V. A. Stennikov, E. A. Barakhtenko, D. V. Sokolov and T. B. Oshchepkova, Problems of modeling and optimization of heat supply systems: new methods and software for optimization of heat supply system parameters, in *Sustaining Power Resources through Energy Optimization and Engineering Premier Reference Source* (IGI Global, Hershey PA, 2016), pp. 76–101.
19. V. A. Stennikov, E. A. Barakhtenko and D. V. Sokolov, Use of Multilevel Modeling for Determining Optimal Parameters of Heat Supply Systems, *Thermal Engineering* **64**(7) (2017) 518–525.
20. V. A. Stennikov, E. A. Barakhtenko and D. V. Sokolov, A Methodology for the Creation of an Integrated

Graphical Environment for Computer Modeling of Energy Pipeline Systems, *Information technologies* **24**(5) (2018) 313–320 (in Russian).