The Domain-Specific Editor for Rule-Based Knowledge Bases

A.Yu. Yurin[*,**], A.F. Berman[*], O.A. Nikolaychuk[*], N.O. Dorodnykh[*] and M.A. Grishenko[***]
[*] Matrosov Institute for System Dynamics and Control Theory, Siberian Branch of the Russian Academy of Sciences (IDSTU SB RAS), Irkutsk, Russia
[**]Irkutsk National Research Technical University (IrNRTU), Irkutsk, Russia
[***]CentraSib LLC., Irkutsk, Russia
tualatin32@mail.ru

***Abstract* – The aim of the paper is to describe a domain-specific editor for the design of rule-based knowledge bases in the field of the prognosis of technical conditions and remaining operation time of petrochemical equipments. The architecture, main functions and a structure of files for configuration of the editor are presented. The feature of the editor is a semantic layer in the form of a platform-independent model. This layer provides to configure the editor with the account of features of a subjects domain. The semantic layer is implemented as a set of domain specific templates describing facts and rules (cause-and-effect relationships). These templates help to abstract from the syntax of certain knowledge representation languages (programming languages for knowledge bases, in particular, CLIPS - C Language Integrated Production System) and generate the graphic user interface elements.**

Keywords - knowledge base, domain-specific editor, platform-independent model, CLIPS, code generation, assessment of technical conditions and operation life

I. INTRODUCTION

The design of knowledge bases (KB) is the main stage of intelligent systems engineering. The results of the stage are represented in the form of formalized and codified knowledge that contains information about the processes, phenomena and objects of the subject domain under consideration.

Several main trends to improve the efficiency of the knowledge bases design are existing, in particular:

- Using the software for ontological and cognitive modeling, CASE-tools (Protégé, OntoStudio, IHMC CmapTools, XMind, FreeMind, TheBrain, IBM Rational Rose, StarUML and etc.), which create graphic models that correspond to the key software abstractions. However, most of these systems do not cover the KBs creation stage and do not provide the completeness of the development process: from the subject domain model to the program codes. In some cases, they can only help obtain graphic images of KB structures. Perhaps, only Protégé is capable of generating a limited set of KB elements, in particular, for CLIPS/COOL.

- Using KBs editors and expert systems shells (Expert System Designer, Expert System Creator, ARITY Expert Development Package, CxPERT, Exsys Developer, DDTRES and etc.), which are programmer-oriented and allow implementation of a formalized description of the domain concepts and KB structures in a certain programming language (PL), but have a low integration capacity with visual modeling systems and knowledge interpretation modules, in most cases supporting one specific PL.

- Using integrated frameworks and unified approaches that provide the coverage of all phases of the life cycle of knowledge-based systems and the integration of the first two trends. It should be noted that this area offers such solutions as AT-TECHNOLOGY [1] and such special methodologies as HeKatE [2] and CommonKADS [3], however, there is a general tendency to target non-programmers [4] and employ conceptual models, including ontologies and semantic nets, when creating KBs.

So, we propose to improve the effectiveness of this stage is the use of specialized software to design knowledge bases directly by the subject domain experts, [5-6] in particular, in the form of domain-specific editors.

This paper describes the domain-specific editor (DSE) for the creation of rule-based knowledge bases in the field of the assessment of technical conditions and remaining operation time of technical objects in petrochemical industry. The DSE is developed with the use of the Personal Knowledge Base Designer (PKBD) platform [7].

The PKBD platform provides the ability to create rule-based expert systems and knowledge bases in accordance with the main principles of a model-driven approach (Model Driven Development) [8].

II. THE MODEL DRIVEN DEVELOPMENT APPROACH

The Model Driven Development (MDD) is an area in the software engineering that considers information models as the central artifacts in the software development providing the automatic synthesis of software codes.

The main MDD concepts are the following:

1. A model is an abstract description of a system (a process) in a formal language. As a rule, the models are

visualized with the aid of certain graphic notations and serialized (represented) in XML.

2. A metamodel is a model of the formal language used to create models (a model of models).

3. A meta-metamodel is a language that describes metamodels. The most common languages for metamodeling are: MOF (Meta-Object Facility), Ecore, KM3 (Kernel Meta Meta Model), etc.

4. A four-layer metamodeling architecture is the concept that defines the different layers of abstraction (M0-M3), where the objects of reality are represented at the lowest level (M0), then the level of models (M1), the level of metamodels (M2) and the level of the meta-metamodel (M3).

5. A model transformation is the automatic generation of a target model from a source model with the accordance of a set of transformation rules [9]. In this case, each transformation rule describes the correspondence between the elements of a source and a target metamodels.

A more detailed description of these concepts can be found in [10].

Currently, there are some ways to implement the model transformation:

- using graph grammars (graph rewriting) (e.g., VIsual Automated model TRAnsformations (VIATRA2) [11], Graph REwriting And Transformation (GReAT) [12], etc.);
- using visual design of transformation rules and category theory (e.g., Henshin [13]);
- using transformation standards (e.g., Query/View/Transformation [14]);
- using hybrid (declarative-imperative) approach for specifying and constructing transformation rules (e.g., ATLAS Transformation Language [15]);
- using declarative and procedural programming languages [16];
- using languages for transforming XML documents (e.g., eXtensible Stylesheet Language Transformations [17], etc.).

However, programmers who use the special model transformation languages should:

- know the specific syntax of the model transformation language;
- be able to describe transformation rules with the aid of the model transformation language;
- know the additional languages, such as an Object Constraint Language [18] that can be used to construct the transformation rules;
- be able to describe metamodels for the source and target languages (to support the transformation process).

It should be noted that all model transformation languages are supported by specific software tools, which, in turn don't provide an opportunity to visualize the development process, i.e. the transformation rules are defined in special text editors focused on programmers. The combination of these factors complicates the use of these languages and tools in a practical way by end-users (e.g., domain experts, knowledge engineers, analysts, etc.), in particular, when developing KBs and ESs on the basis of conceptual models. So, in practice developers prefer 'ad-hoc' solutions for particular tasks with the use of declarative and procedural programming languages.

The following MDD initiatives are best known today:

- The Eclipse Modeling Framework (EMF) is an Eclipse-based modeling framework and a code generation facility for building tools and other applications based on a structured data model. EMF provides the foundation for interoperability with other EMF-based tools and applications. The heart of EMF is Ecore. Ecore is a special language for description of meta-models (implementation of an OMG's Essential Meta-Object Facility, EMOF). The basic tools to work with meta-models and a skeletal code generation of software (programming skeletons) are EMF.Core, EMF.Edit, EMF.Codegen.
- The 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 some models in the center of the entire system life-cycle, including specification, design, development, verification, integration, and maintenance. MIC provides three core elements: the technology for the specification and use of the domain-specific modeling languages (DSML); a 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).
- The Model-Driven Architecture (MDA) is a registered trademark of the Object Management Group (OMG). The main idea of this 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 platform specific. The viewpoint is an abstraction technique for focusing on a particular set of concerns within a system while dismissing all irrelevant

details. The viewpoint can be represented via one or more models.

In the context of the development of rule-based KBs, the MDA as the most standardized version (initiative) of the MDD was selected in the form of the primary approach.

III. The Personal Knowledge Base Designer

The PKBD is a research platform based on the universal rule-based knowledge base editor designed for the end-users.

This platform provides the creation of domain-specific editors for knowledge bases by means of an implementation of the main principles of the Model Driven-Architecture (MDA) [19] (a separate direction within the MDD). The MDA assumes a clear definition of three levels (viewpoints) of the software representation:

- a computation-independent level that describes the basic concepts and relationships of the subject domain expressed in the form of ontologies. Models of this level can be form automatically on the basis of the analysis of XML-like formats [20], in particular: XMI (XML Metadata Interchange) for UML class diagrams, CXL (Concept Mapping Extensible Language) for concept maps, ETXL (Event Tree Mapping Extensible Language) for event trees;

- a platform-independent level that represents the subject domain model in the context of the knowledge representation formalism used, in particular, a logical rules formalism. The PKBD provides a mean for visual modeling – the Rule Visual Modeling Language (RVML) based on the UML [21];

- a platform-dependent level that represents a formalized description of knowledge bases taking into account the features of a certain software platform. The PKBD currently supports CLIPS.

The architecture of the PKBD includes the main modules for: knowledge base management, for knowledge representation languages support, dictionaries management, integration with CASE-tools, interpretation of models and the graphical user interface (GUI) generation.

The interface is generated both in the form of pop-up windows and wizards.

Wizards represent sequences of GUI forms, which segment and order the processes of entering and editing elements of the knowledge base. In particular, the user is consistently asked to specify: the name of the template (used for display in the editor), the short name (used in the process of inference), the description and the properties (slots) of the template when entering it; the property name, its short name, description, a type (string, symbol or number), a possible default value and a constrain for this value (more, less, equal, unequal) when entering a slot.

The similar wizards are used when entering and editing facts and rules.

Let's consider the process of configuring the PKBD for designing rule-based knowledge bases in the field of a prognosis of technical conditions and an operation life for technical objects in petrochemical industry.

IV. The Domain-Specific Editor

The main difference between the DSE and the universal one (based on the PKBD platform) is the use of a semantic layer in the form of a structure of a platform-independent model (a set of templates for facts and rules). In this case, each fact template describes a certain concept of the subject domain and is considered as an analog of the concepts of the "prototype frame" or "sample frame" from the knowledge representation frame model [5].

The rule template describes the cause-effect relationship between the concepts in the following form:

```
IF template_1 AND template_2 AND … AND template_N THEN template_M.
```

The specialized configuration files are used to represent and store templates (Tables 1 and 2). The special PKBD module interprets these files and generates the elements of the knowledge acquisition system and the graphical user interface (GUI), which provide the interaction of experts with the DSE in terms of a subject domain. The configuration files are created by a knowledge engineer (an administrator) once before the DSE is transferred to the end-user who can add new fact and rules (concrete instances of concepts and relationships), but can't change templates (the concepts).

So, the DSE has the role-based mechanism to control the access to functions. There are two main user roles:

- A knowledge engineer (an administrator) configures the editor and changes templates. This role requires in-deep knowledge of the problem area.
- An expert (a subject domain specialist) inputs of rules and facts without the possibility of making changes to the templates.

The DSE provides following main functions:

- managing knowledge bases, including:
  - open, load and save the knowledge bases;
  - create, delete and edit the elements of knowledge bases (facts and rules);
  - import and export information from conceptual models (XMI, CXL and ETXL formats);
  - preview of the knowledge base elements using the RVML [21];
  - generate knowledge bases codes for the target programming language, in particular, CLIPS, including description of slots, templates, facts and rules.
- managing dictionaries (add and remove the concepts to directories with the purpose to improve the efficiency of entering facts and rules);

TABLE I. THE STRUCTURE OF A CONFIGURATION FILE FOR FACT TEMPLATES

File element	*Description*	*Example*
[**Metadata**]	The file title.	**[Metadata]**
;description	The template description.	;Exist damage
tempale_name=<template name>	The template name, this field is used for the generation of GUI and peogramm codes.	tempale_name=Damage
edited_by_user=<No|Yes>	The instruction for the GUI generator. This field determines the ability to edit the generated forms by the end-user.	edited_by_user=No
[**Fields**]	The section title for the template slots description.	**[Fields]**
<slot_name>=<datatype>	Each line is a slot description, where <slot_name> is the internal name used to generate the GUI forms and CLIPS codes; <datatype> = string | integer | float | <name_of_the_variable_from_the_values_list>, where < name_of_the_variable_from_the_values_list > = <value>; ...; <value> - is defined in the "**Values**" section.	dam-type=string dam-orientation=val3:1
[**Captions**]	The section title for the template slots captions. These captions are displayed on the GUI forms and used for the multilanguage support.	**[Captions]**
<slot_name> = <name>	Each line is a slot description.	form=Exist damage dam-type=Type of the damage
[**Values**]	The section title for the passible values of slots.	[**Values**]
<value_name> = <value_1>,…,<value_N>	Each line is a description of the passible values of slots.	Val3:1=LONGITUDINAL, CROSS-SECTION

TABLE II. THE STRUCTURE OF A CONFIGURATION FILE FOR RULE TEMPLATES

File element	*Description*	*Example*
[Generalized rules]	The file title.	**[Generalized rules]**
#<Name of the process>	The name of the degradation or the dangerouse process that is described by the the rules. This name will be displayed on the screen.	#Corrosion cracking
##<Name of the process modification>	The name of the process modification (the sub-process). This name will be displayed on the screen.	##1
<rule_name> = <title> : <template names from the rule conditions> : <template names from the rule actions>	<rule_name> - the name of the rule for GUI and CLIPS codes generators, <title> - the name of the rule that will be displaed on the GUI forms, <template names from the rule conditions> = <template_name>, …, <template_name>.	fail-mechanism-fail-ky=Rule_for defenition_of_the_failure_mechanism: exist-meh-des,exist-des:exist-meh-fail

- testing the knowledge bases:
  - select the inference machines in the form of dynamic link libraries;
  - logical inference with the aid of libraries selected;
  - description of the decision making process in the form of a chain of activated rules;
- building reports for:
  - description of the knowledge bases;
  - description of the logical inference results.

The process of DSE configuring includes the following steps:

1. Creation of the semantic layer in the form of the platform-independent model, including:

- conceptualization and formalization of the main concepts and relationships;
- designing templates for describing logical rules.

2. Testing and verification of the knowledge base designed and the DSE obtained.

The DES supports the knowledge base engineering for the end-users in accordance with the features of the problem of the prognosis of technical conditions and an remaining operation time. Thus, we analyze failure cases of the equipment in petrochemistry [22] and industry expertise [23]. The main results of this step are the concepts (for fact templates) and relationships between concepts (for rule templates).

In particular, it is defined that the basic concept of the subject domain is the "undesirable process" concept [24] (fig. 1).

The undesirable process [24] is a set of physical-chemical processes resulted from various technological processes, imperfections and infringements of a constructive, industrial and operational origin.

The following fact templates are defined on the basis of the structure of the "undesirable process" concept (fig. 1): the "material" template includes the material name and it's technological (e.g., residual stresses, constructive heredity, etc.) and metallurgical (e.g., manufacturing defects, etc.) heredity; the "mechanical loadings" template includes description of the static, variable, dynamic and temperature loadings, etc. Next, the template configuration files are generated on the basis of templates obtained.

The dynamics of the undesirable process can be represented in the form of a cause-effect chain of classes of technical conditions: initial defectiveness, damage, destruction, failure. The cause-effect chain of classes defines of the sequence of the steps of degradation process description, which, in turn, stored in configuration files.

So, the platform-independent model consists on such and similar templates and describes the sequence of the

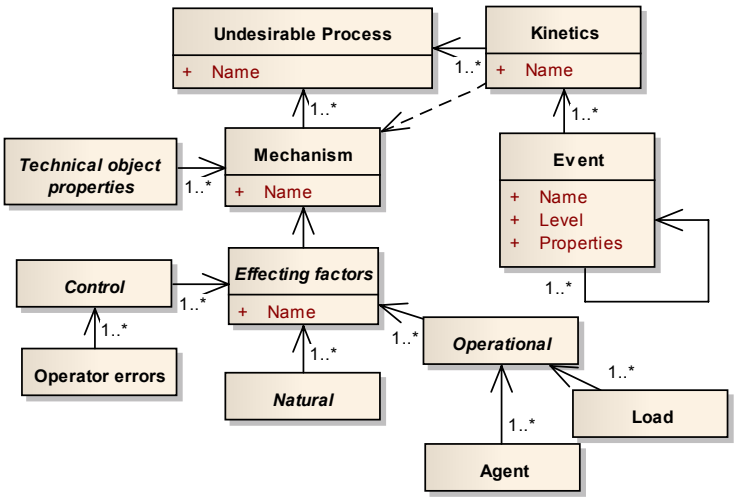

Figure 2. The "undesirable process" concept

technical condition classes or steps of the certain process, where each class/step are defined by a mechanism and kinetics. The mechanism is a set of the effecting factors and properties of the technical object. The technical object is an owner of the process. Kinetics is a set of sequences of events of the process.

Let's present some examples of the sequence of the steps for the description of the "corrosion cracking" degradation process (fig. 2), a rule template configuration file and GUI forms of the DSE (fig. 3) [20]:

```
[Generalized rules]
#Corrosion cracking
##1
dam-mechanism-dam-ky= Rule_for defenition_
of_the_damage_ mechanism:material,structural-
heredity,making-defects,mechanical-stress-
const,thermal-stress,technological-
environment,heat-exchange-technological-
environment,flow-technological-
environment,surface-damage-from-corrosive-
environment:exist-meh-dam
dam-damage-
ky=Rule_for_defenition_of_damages:exist-meh-
dam:exist-dam
des-mechanism-des-
ky=Rule_for_defenition_of_the_desctruction_mecha
nism:exist-meh-dam,exist-dam:exist-meh-des
des-destruction-ky=
Rule_for_defenition_of_the_desctructions:exist-
meh-des:exist-des
…
```

The user can preview the knowledge-base elements in the form of RVML schemes (fig. 4) [20], which are corresponded to CLIPS codes. The CLIPS codes obtained may be used further in external systems.

V. CONLUSION

The effective knowledge base engineering requires the development and the use of the specialized software, in particular, in the form of domain-specific editors (DSE).

In this paper we describe the configuration and application of the DSE on the basis of the PKBD platform. The DSE developed uses the main principles of the MDD, in particular, it provides creating and interpreting the platform-independent models for the certain subject

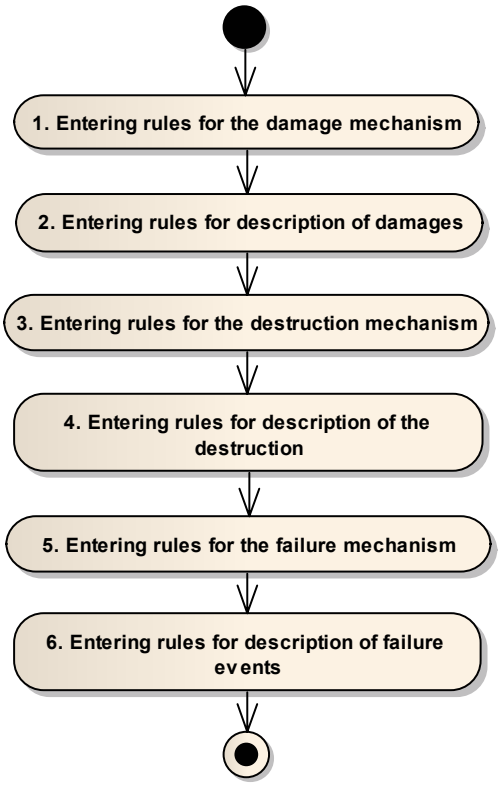

Figure 1. The sequence of the steps for the description of the "corrosion cracking" degradation process

domain. The problem of prognosis of technical conditions and the remaining operation time of petrochemical equipments is used as a case study.

The DSE was used for the development of the knowledge base for defining the causes of damage and destruction of construction materials [23], and they were also used in the educational process at the Irkutsk National Research Technical University (IrNRTU).

ACKNOWLEDGMENT

The reported study was partially supported by RFBR (research project No. 18-07-01164).

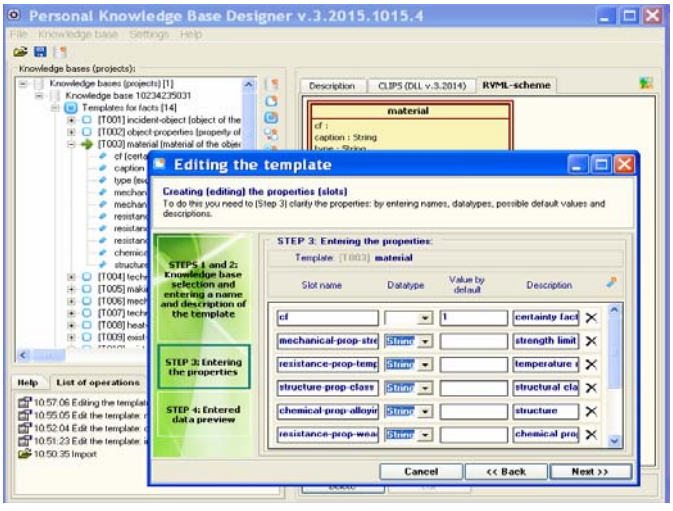

Figure 3. The example of DSE GUI forms

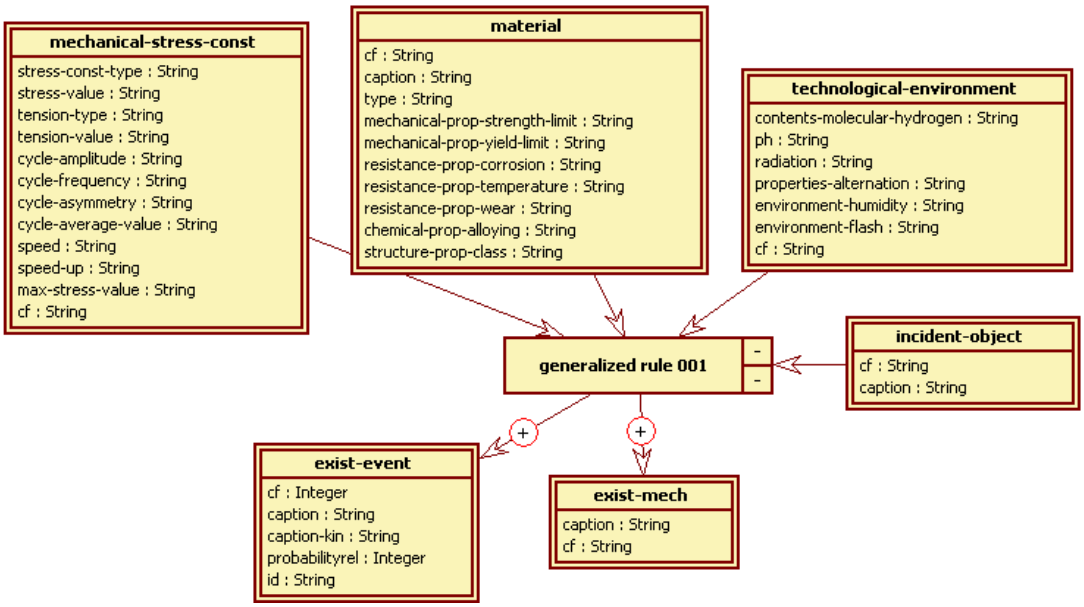

Figure 4. An example of the RVML scheme for a rule

REFERENCES

[1] G.V. Rybina, V.M. Rybin, Y.M. Blokhin and S.S. Parondzhanov, "Intelligent Programm Support for Dynamic Integrated Expert Systems Construction", Procedia Computer Science, 2016, vol.88, pp.205-210.

[2] G.J. Nalepa and A. Ligęza, "HeKatE methodology, hybrid engineering of intelligent systems", International Journal of Applied Mathematics and Computer Science, 2010, vol.20 (1), pp. 35-53.

[3] G. Schreiber, H. Akkermans, A. Anjewierden, R. de Hoog, N. Shadbolt, W.V. de Velde and B.Wielinga. Knowledge Engineering and Management: The CommonKADS Methodology. The MIT Press, Cambridge, 2000.

[4] M. Nofal and K.M. Fouad, "Developing Web-Based Semantic Expert Systems", International Journal of Computer Science Issues, 2014, vol. 11 (1), pp. 103-110.

[5] J.C. Giarratano and G. Riley, Expert Systems: Principles and Programming. 4th Ed. Course Technology, 2004.

[6] G.F. Luger. Artificial Intelligence: Structures and Strategies for Complex Problem Solving. 6th Ed. Addison-Wesley, 2008.

[7] M.A.Grishenko, N.O. Dorodnykh and A.Yu.Yurin, "Software for rule knowledge bases design: Personal Knowledge Base Designer", Open Semantic Technologies for Intelligent Systems, 2016, vol.6, pp.209-212.

[8] B.Sami, M. Book and V. Gruhn. Model-Driven Software Development. Springer-Verlag Berlin Heidelberg, 2005.

[9] A. Kleppe, J. Warmer and W. Bast, MDA Explained: The Model Driven Architecture: Practice and Promise. 1st Ed. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, 2003.

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

[11] D. Varro and A. Balogh, "The model transformation language of the VIATRA2 framework", Science of Computer Programming, 2007, vol.63(3), pp.214-234.

[12] D.Balasubramanian, A.Narayanan, C.Buskirk and G.Karsai, "The graph rewriting and transformation language: GreAT", Electronic Communications of the EASST, 2006, vol.1, pp.1-8.

[13] T.Arendt, E. Biermann, S.Jurack, C. Krause and G. Taentzer, "Henshin: advanced concepts and tools for in-place EMF model transformations", Processing of the 4th International Conference on Model-Driven Engineering and Software Development (MODELSWARD 2010). Lecture Notes in Computer Science, 2010, vol.6394, pp.121-135.

[14] Query/View/Transformation Specification Version 1.3 (2016). [Online]. Available: http://www.omg.org/spec/QVT/1.3/.

[15] F.Jouault, F.Allilaire, J.Bezivin and I. Kurtev, "ATL: A model transformation tool", Science of Computer Programming, 2008, vol.72(1), pp.31-39,.

[16] A.F.Berman, O.A. Nikolaychuk and A.Y. Yurin, "Intelligent planner for control of failures analysis of unique mechanical systems", Expert Systems with Applications, 2010, vol.37, pp.7101-7107.

[17] XSL Transformations (XSLT) Version 2.0 (2007). [Online]. Available: http://www.w3.org/TR/xslt20/.

[18] Object Constraint Language Specification Version 2.4 (2014). [Online]. Available: http://www.omg.org/spec/OCL/2.4/.

[19] D. Frankel. Model Driven Architecture: Applying MDA to Enterprise Computing. 1st Ed. New York: Wiley, 2003.

[20] N.O. Dorodnykh and A.Yu.Yurin, "Using UML class diagrams for design of rule-bases", Software Engineering, 2015, vol. 4, pp. 3-9 (in Russ.).

[21] A.Y.Yurin, "A notation for the design of rule-based knowledge bases of expert systems", Object systems, 2016, vol.12, pp.48-54 (in Russ.).

[22] A.F. Berman and Khramova V., "Automated data base for failures in pipelines and tubular high-pressure apparatus", Chemical and Petroleum Engineering, 1993, vol. 29, pp. 63-66.

[23] A.F. Berman, O.A. Nikolaichuk, A.Y.Yurin and K.A. Kuznetsov, "Support of Decision-Making Based on a Production Approach in the Performance of an Industrial Safety Review", Chemical and Petroleum Engineering, 2015, vol. 50 (11-12), pp. 730-738.

[24] A.F. Berman and O.A. Nikolaichuk, "Technical state space of unique mechanical systems", Journal of Machinery Manufacture and Reliability, 2007, vol.36 (1), pp. 10-16.