Fishbone Diagrams for the Development of Knowledge Bases

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

Abstract **- The paper describes an approach for the automated development of rule-based knowledge bases by transforming fishbone diagrams. The approach is based on the identification and extraction of structural cause-effect elements of fishbone diagrams and their transformation into the elements of a target knowledge representation language, in particular, C Language Integrated Production System (CLIPS). The source metamodel of fishbone diagrams, the target metamodel for a unified representation of rules (a model for representation logical rules), transformation operators and a transformation technique are presented. An illustrative example describes the development of a rule-based knowledge base for diagnosing and forecasting the states of complex technical systems based on the approach proposed.**

Keywords – knowledge acquisition; knowledge base; rules; fishbone diagrams; model transformation, code generation; CLIPS

I. Introduction

Currently, the development of new approaches and tools for the automated creation of domain-specific intelligent systems remains a relevant topic of scientific research [1-5]. It should be noted that such domain-specific systems use heterogeneous structural information that requires the application of adequate processing methods, in particular, expert systems designed to process a poorly- and semi-structured knowledge.

The main element of expert systems is a knowledge base (KB). The KB development process includes the stages of acquisition, structuring, formalization and codification of expert knowledge, which traditionally are considered as a "bottleneck" of the intelligent systems engineering [6, 7]. In recent years there is a trend to automate these stages by the means of visual (cognitive) modeling. The main results of this modeling are visual conceptual models (e.g., concept maps or mind maps, etc.), which have a high cognitive power and ability to describe the regularities of a subject domain. This trend is confirmed by examples [8-10].

Today, there are many different languages and standards for representing the conceptual models. For example, UML, IDEF, DFD and others are widely used for information systems designing. There are also domain-specific means addressing the features of knowledge representation models (e.g., RVML [11], XTT2 [12], etc.). However, domain experts prefer to use models that take into account the features of the subject domain problems and have a system-wide focus. It should be noted that most of the tools for designing such models considering these models only as graphic artifacts (primitives) ie. there is no possibility to generate program codes, specifications or software components.

In this paper we propose the approach for the automated development of rule-based knowledge bases by transforming fishbone diagrams [13]. These models describe the dynamics of degradation processes (DP) and processes of violation of technogenic safety (fire, explosion and other dangerous processes) in petrochemistry. In this case, the transformation is the process of an automatic generation of a target KB from a source model according to a set of transformation rules. Transformation rules describe how one or more constructs of the source language can be transformed into one or more constructs in a target knowledge representation language (KRL).

The C Language Integrated Production System (CLIPS) [14] is selected as the target KRL. CLIPS is one of the most widely used means for rule-based expert systems developing.

II. The problem statement

The approach proposed provides to automate the stages of knowledge formalization and codification by the generation the rule-based KB codes in the CLIPS format on the basis of the fishbone diagram analysis. In turn, the fishbone diagrams analyzed are the results of the stages of acquisition and structuring (conceptualization).

Let's formalize the problem statement as follows:

$$T : CM^{FD} \rightarrow KB, \qquad (1)$$

where CM^{FD} is a source conceptual model in the form of a fishbone diagram; KB is a target KB in the form of CLIPS codes, $KB = \left\langle Code^{CLIPS} \right\rangle$.

Thus, it is necessary to define a set of transformation rules (T) in the context of designing rule-based KBs and implement them in the form of a special tool (transformation module).

The previously developed technology [15] that was implemented in the form of a web-based program system – Knowledge Base Development System (KBDS) [16] was used. This technology supports the creation of software components (transformation modules) designed for converting the source conceptual models (represented in XML-like formats) into the KB codes of the target KRL (in particular, CLIPS).

III. Fishbone diagrams

The fishbone diagrams (also called Ishikawa diagrams or "fish skeletons") are cause-and-effect diagrams that represent the graphical ordering of factors effecting the object. These diagrams are useful for extracting and visualizing knowledge about the mechanisms of different processes. Figure 1 shows the meta-model of the fishbone diagrams.

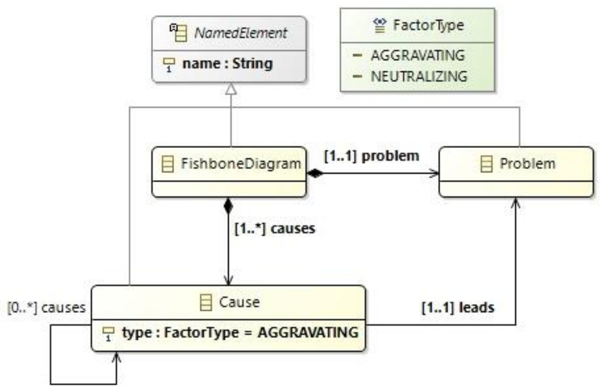

Figure 1. The meta-model of fishbone diagrams

The "fishbone head" of the diagram is the problem under investigation. The "fishbone spine" is depicted as a direct horizontal arrow. The "bones" represent causes and factors directly and indirectly effecting the considered problem and they are depicted by inclined arrows. The arrows of the effecting factors of the second on other levels can be added to the main arrows of cause categories. At the same time, different factors can both aggravate and neutralize the problem. Traditionally the "brainstorming" method is used to identify and form the possible main cause categories and their details (offshoots of "bones"). In fact, this method provides the generation of creative ideas by a group (team) of experts.

In the case of the problem of the definition of mechanisms of degradation or dangerous process we propose to specialize the fishbone diagram and develop a specific diagram template that includes the main cause categories (the first level of "bones"). This fishbone diagram template is shown in Figure 2. Only three levels of "bones" are sufficient to determine the mechanism of different DP types. So, the subject domain experts need to specify this diagram template. After that, it is possible to generate automatically a rule draft that describes the mechanism. If necessary this draft can be completed with the corresponding logical relationships (e.g., "AND" or "OR"):

IF (Technical_object_property$_1$ **AND** … **AND** Technical_object_property$_n$) $\circ$ (Operational_factor$_1$ **AND** … **AND** Operational_factor$_n$)

THEN Mechanism **AND** (Event$_1$ $\circ$ Even$_n$), where $\circ$ is a logical operation $\circ \in \{\wedge, \vee, \oplus\}$.

Also it is suggested to associate a certainty factor (CF) with each possible specific cause (the second level of "bones"). The CF value reflects the subjective degree of the expert's confidence about the effect of this specific cause on the problem occurrence. The calculated CF value depends on the corresponding aggravating and/or neutralizing factors (the third level of "bones"). At the same time, aggravating and neutralizing factors can be expressed by a concrete value (a "name-value" bundle).

The abstract syntax of the fishbone diagram for describing the DP mechanism is shown in Figure 4. The metamodel proposed corresponds to the meta-metamodel Ecore and it is further used as the initial metamodel for the development of transformation rules (transformation models), which are describe the correspondence between the elements of the initial metamodel and the target metamodel of the rule-based KB.

IV. Knowledge Base Development System

The KBDS implements the approach to the development of software components (modules) for the program code generation for KBs based on conceptual models presented in the XML format (the most common format for the exchange and storage of the different conceptual models).

The main purposes of the developed software are:

- the support of the development of software components for the program code generation for KBs based on different conceptual models;
- the support of the development of KBs (with the use of the developed software components).

Further we present the main functions and the architecture of KBDS in detail.

The main functions of KBDS in the context of the development of the software components are the following:

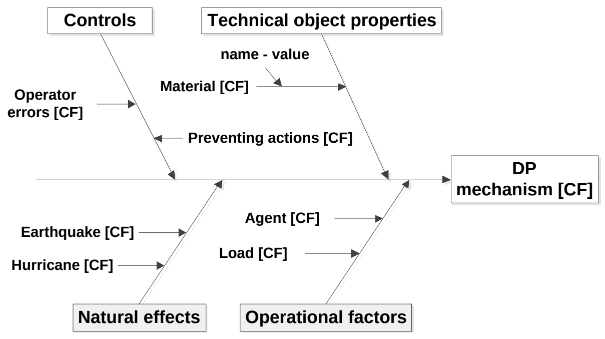

Figure 2. The fishbone diagram template with main cause categories of degradation processes

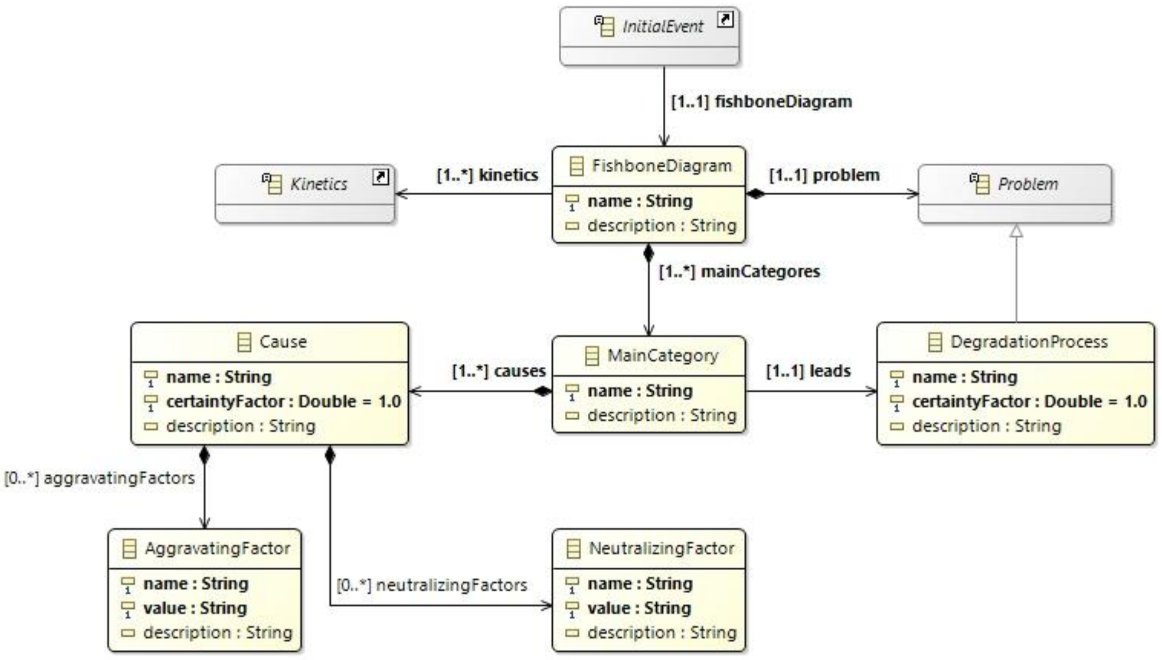

Figure 3. A metamodel of the fishbone diagram for determining the mechanism of the DP

- the creation (import) of a meta-model of the source conceptual model on the basis on the XML schema (XSD) of this model, or by analyzing the model (Reverse Engineering);
- the visual representation and modification of the obtained meta-models;
- the visual design of a transformation model (in the form of a set of rules for transforming the elements of the source model to the elements of the target model) and the automatic TMRL (Transformation Model Representation Language) [15] code generation;
- the generation (assembling) of the software component based on the developed transformation model and the analyzer and generator units selected (depends on the type of the software component).

The KBDS provides the following main functions for the design of the KBs:

- the code generation for KBs on the targeted knowledge base programming language (CLIPS or OWL) using the software components developed;
- the automated synthesis of an ontological and a rule-based model (the internal representation of knowledge in the KBDS) based on the analysis of conceptual models;
- the storage and representation of obtained knowledge with the use of these models;
- the use of the special graphic notation - Rule Visual Modeling Language (RVML) [11] for the representation and modeling the logical rules;
- the visual representation and modeling knowledge in the form of ontological model.

The KBDS has the client-server architecture that implements the main factions.

The client part of the KBDS includes the following main modules:

- the RVML editor that provides a visual representation and editing the logical rules with the aid of RVML;
- the ontology editor that provides a visual representation and editing knowledge in the form of a graph (ontological model);
- the meta-model editor that provides a visual representation and editing the meta-model elements;
- the transformation model editor that provides a visual representation and editing the transformation rules.

The server part of the KBDS includes the following main modules:

- the administration module that provides an user interaction with the KBDS (the limitation of user rights, the collection and analysis of various statistical information, etc.);
- the knowledge bases management module that provides a creation and managing KB projects;
- the meta-level management module that provides an internal representation of knowledge in the KBDS in the form of the rule-based and ontological models. These models allow us to abstract form the features in elements descriptions of various knowledge representation languages which are used in the implementation of KBs (for example, CLIPS, Jess, Drools, RuleML, OWL,

SWRL, etc.) and to store knowledge in own independent format;

- the software components development module that provides the creation and managing software component projects, as well as code generation of the software components based on the developed transformation model and the analyzer and generator units selected;
- software components that provide a synthesis of the KB model (ontological or rule-based models) based on the analysis of conceptual models and a program code generation for a KB (CLIPS or OWL) based on the analysis of the KB model or the source conceptual model.

The prototype is implemented using: PHP, Yii2 Framework and JQuery, jsPlumb libraries. The PostgreSQL was used as the data base management system.

V. Transformation Model Representation Language (TMRL)

The TMRL [15] grammar belongs to the class of context-free grammars (CF-grammars, for example - LL (1)). The TMRL constructs allow one to describe the elements of the transformation model in a declarative form, in particular, the rules for the correspondence of metamodel elements, as well as the mechanism of interaction with previously developed (external) software components for the transformations. TMRL specifications meet the requirements of accuracy, clarity and completeness, ie. the specifications for TMRL contain all the necessary information (for the considered transformations) to solve the task, all objects of the model are well formalized, while the specifications are compact enough and at the same time understandable (readable). TMRL contains 15 special elements (lexemes), which are used to describe the source and target meta-models.

The main difference between TMRL and existing models transformation languages is its ease of use, achieved through a limited set of elements. TMRL is not an extension of other languages and does not use the constructions of other languages, as other transformation languages very often do, in particular, ATL uses the OCL [17]. In addition, TMRL has human-readable syntax for the purpose of making the necessary modifications to the model of the transformation manually, if necessary. An additional feature of TMRL is the ability to describe interaction with previously developed software components of the transformation in support of the import of different formats of conceptual models.

The TMRL programs are created in the special editor for visual programming (it is a subsystem of the KBDS) (Fig. 4).

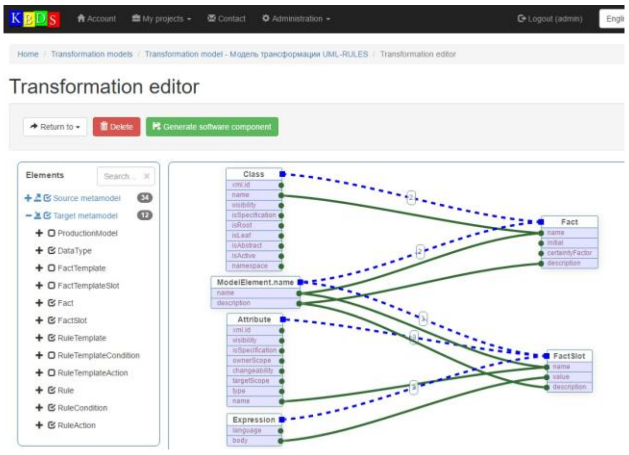

Figure 4. The editor for TMRL visual programming

VI. The Transformation Model

Let's define the transformation operator (T) from (1):

$$T = \langle T_{CM-RM}, T_{RM-KB} \rangle, \qquad (2)$$

$$T_{CM-RM} : M^{FD}_{XML} \rightarrow M^{PR},$$
$$T_{RM-KB} : M^{PR} \rightarrow Code^{CLIPS},$$

where T_{CM-RM} is a set of rules for transformation of the source fishbone diagram into the model of logical rules; T_{RM-KB} is a set of rules fro transformation of the model of logical rules into the KB code in CLIPS; M^{FD}_{XML} is a model of fishbone diagrams in the XML-like format (in our case the Fishbone Diagram Mapping Extensible Language (FDXL) is used); M^{PR} is a model for the representation of gained knowledge in the form of logical rules. This model doesn't depend on the used KRL (e.g., CLIPS, Jess, Drools, RuleML, etc.) and contains only general constructions inherent in all rule-based KRL; $Code^{CLIPS}$ is a target KB code in CLIPS.

The transformations from (2) are carried out at the abstract level of metamodels. For this purpose it is necessary to define:

- the transformation rules for T_{CM-RM} ie. between the elements of the metamodels of the fishbone diagram and the model of logical rules;

- the transformation rules for T_{RM-KB} ie. between the

TABLE I. The Elements Correspondences

Fishbone diagrams (FDXL)	Logical rules	CLIPS
FishboneDiagram	FactRule (шаблон правила)	defrule
Effect	FactTemplate / Action	deftemplate
Effect (certaintyFactor)	Fact / Action(certaintyFactor)	-
Cause (certaintyFactor)	Rule(certaintyFactor)	-
AggravatingFactor	FactTemplate / Condition	deftemplate / defrule
AggravatingFactor (name)	Slot	Slot
NeutralizingFactor	FactTemplate / Condition	deftemplate / defrule

elements of the metamodel of logical rules and CLIPS constructions.

The transformation rules for T_{CM-RM} and T_{RM-KB} form a transformation model (scenario) in terms of the domain-specific declarative language – Transformation Model Representation Language (TMRL) [15].

Examples of compliance between the main elements of the fishbone diagrams, a logical rules model and CLIPS are presented in Table 1.

VII. TRANSFORMATIONS OF FISHBONE DIAGRAMS

The technique for transformations of fishbone diagrams into the KB code with the use of the KBDS consists of the following main steps:

Step 1: specifying the fishbone diagram templates by means of various software (domain-specific editors or general editors, for example, Microsoft Visio and etc.). At this step the subject domain expert defines the cause-and-effect information about mechanisms of degradation or dangerous process.

Step 2: storing the designed diagrams in the XML-like format, for example, FDXL.

Step 3: analysis of the obtained XML documents of fishbone diagrams with the KBDS [16]. The results of this step are the model elements and their relationships corresponding to the transformation model.

Step 4: generating the model for the logical rules representation based on the extracted diagram elements. This model is stored in the KBDS.

Step 5: visualization, modification and verification of rules obtained with a special graphical editor that is a part of the KBDS. The original author's notation – Rule Visual Modeling Language (RVML) is used to improve the visibility of represented rules. RVML provides:

- to use special graphic primitives to represent the elements of logical rules;
- to assign the subjective probabilities in the form of the CFs for facts and rules;
- to show the type of actions for a logical rule (assert, modify, retract).
- to show the logical operators in the rule conditions (e.g., "AND", "OR" and "NOT").

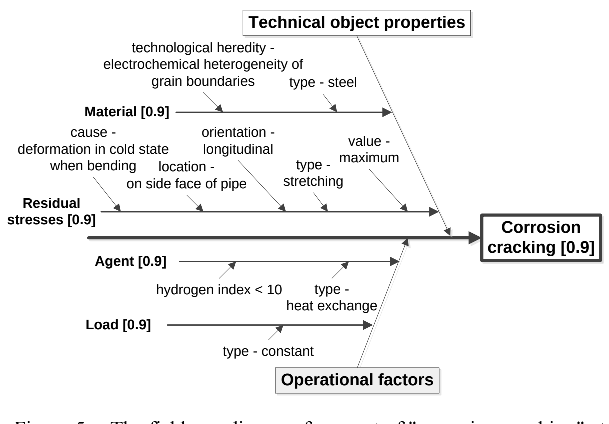
Technical object properties
technological heredity - electrochemical heterogeneity of grain boundaries
type - steel
Material [0.9]
cause - deformation in cold state when bending
orientation - longitudinal
value - maximum
location - on side face of pipe
type - stretching
Residual stresses [0.9]
Corrosion cracking [0.9]
Agent [0.9]
hydrogen index < 10
type - heat exchange
Load [0.9]
type - constant
Operational factors

Figure 5. The fishbone diagram fragment of "corrosion cracking" at the damage stage

Step 6: automatic generation of KB codes in the CLIPS format based on the model modified.

VIII. AN ILLUSTRATIVE EXAMPLE

Let's consider an illustrative example of application of the approach proposed for the automated development of a rule-based KB by transforming the fishbone diagrams. These diagrams describe the "corrosion cracking" DP mechanism at the damage stage (Fig. 5).

The following FDXL fragment corresponds to the fishbone diagram (Fig. 5):

```
<FishboneDiagram name="Mechanism of a degradation process" description="Fishbone diagram for description of a mechanism of a degradation process ">
<MainCategory id="MC-1" name="Operational factors" description="Main category of causes">
<Cause id="C-1" name="Load" certainty-factor="0,9" description="Root factor">
<AggravatingFactor id="AF-1" name="type" value="constant" description=""/>
</Cause>
<Cause id="C-2" name="Agent" certainty-factor="0,9" description="Root factor">
<AggravatingFactor id="AF-2" name="type" value="heat exchange" description=""/>
<AggravatingFactor id="AF-3" name="hydrogen index" value="< 10" description=""/>
</Cause>
</MainCategory>
<MainCategory id="MC-2" name="Technical object properties" description="Main category of causes">
<Cause id="C-3" name="Material" certainty-factor="0,9" description="Root factor">
<AggravatingFactor id="AF-4" name="type" value="steel" description=""/>
<AggravatingFactor id="AF-5" name="technological heredity" value="electrochemical heterogeneity of grain boundaries" description=""/>
</Cause>
<Cause id="C-4" name="Residual stresses" certainty-factor="0,9" description="Root factor">
<AggravatingFactor id="AF-6" name="value" value="maximum" description=""/>
<AggravatingFactor id="AF-7" name="type" value="stretching" description=""/>
<AggravatingFactor id="AF-8" name="orientation" value="longitudinal" description=""/>
<AggravatingFactor id="AF-9" name="location" value="on side face of pipe" description=""/>
<AggravatingFactor id="AF-10" name="cause" value="deformation in cold state when bending" description=""/>
</Cause>
</MainCategory>
<Effect id="DP-1" name="Corrosion cracking" certainty-factor="0,9" description="Degradation process is a corrosion cracking (a damage stage)"/>
</FishboneDiagram>
```

The extracted fishbone diagram elements are used to generate rules templates, which can be represented in the RVML notation (Fig. 6). Further, the templates are modified and used to create specific rules.

The following CLIPS code corresponds to the analyzed FDXL document:

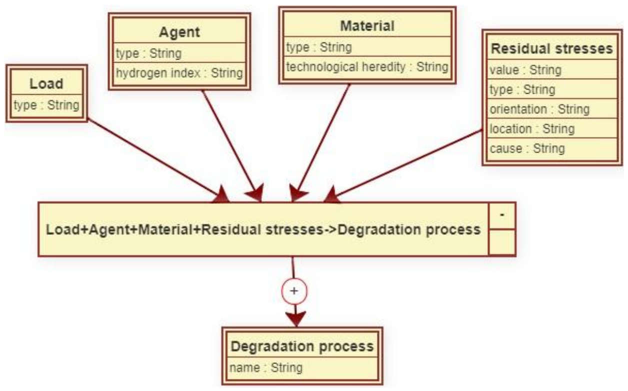

Figure 6. An example of a rule template in RVML

```
    (defrule Material+Load+Residual_stresses+Agent -
>Degradation_process
        (declare (salience 1))
        (Material (type "steel")
                  (technological_heredity
   "electrochemical heterogeneity of grain
   boundaries") )
        (Load (type "constant") )
        (Residual_stresses (value "maximum")
             (type "stretching")
             (orientation "longitudinal")
             (location "on side face of pipe")
             (cause "deformation in cold state when
bending")  )
        (Agent (type "heat exchange")
               (hydrogen_index "<10") )
        =>
        (assert (Degradation_process
                  (name "Corrosion cracking") )
        ) )
```

IX. Conlusion

The paper describes the approach for the automated development of rule-based KBs for diagnosing and forecasting the states of complex technical systems, in particular, for definition of the mechanisms of degradation and dangerous processes.

The approach is based on the analysis of structural cause-effect elements of fishbone diagrams describing the mechanism of DPs and their transformations into the target KRL elements (CLIPS). This approach helps to reduce the time for the KB development and to avoid programming errors at the stage of knowledge formalization due to the automatic KB code generation. The approach also provides the ability for the subject domain experts to be involved in the development of domain-specific intelligent systems at the codification stage by means of the generation of program codes on the basis of subject domain models.

The proposed approach implements the technology [15] and the KBDS platform [16], and is used to prototype the KBs for the industrial safety review support [18].

Acknowledgment

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

References

[1] W. Wang, M. Yang and P. H. Seong, "Development of a rule-based diagnostic platform on an object-oriented expert system shell", Annals of Nuclear Energy, 2016, vol. 88, pp. 252–264.

[2] Y. Tanaka and K. Tsuda, "Developing Design Support System Based on Semantic of Design Model", In Proceedings of the 20th International Conference on Knowledge Based and Intelligent Information and Engineering Systems, KES2016. Procedia Computer Science, 2016, vol. 96, pp. 1231–1239.

[3] C. Urrea, G. Henríquez and M. Jamett, "Development of an expert system to select materials for the main structure of a transfer crane designed for disabled people", Expert Systems with Applications, 2015, vol. 42, no. 1, pp. 691–697.

[4] D. Zhang, X. Chen and H. Yao, "Development of a Prototype Web-Based Decision Support System for Watershed Management", Water, 2015, vol. 7, pp. 780–793.

[5] 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.

[6] J. C. Giarratano and G. Riley, Expert Systems: Principles and Programming, 4th ed. Thomson Course Technology, 2005.

[7] P. Jackson, Introduction to Expert Systems, 3rd ed. Addison-Wesley, Harlow, 1999.

[8] T. A. Gavrilova and I. A. Leshcheva, "Ontology design and individual cognitive peculiarities: A pilot study", Expert Systems with Applications, 2015, vol. 42, pp. 3883–3892.

[9] M. Herrero-Zazo, I. Segura-Bedmar and P. Martínez, "Conceptual models of drug-drug interactions: A summary of recent efforts", Knowledge-Based Systems, 2016, vol. 114, pp. 99–107.

[10] R. R. Starr and J. M. Parente de Oliveira, "Concept maps as the first step in an ontology construction method", Information Systems, 2013, vol. 38, pp. 771–783.

[11] A. Y. Yurin, "A notation for the design of rule-based knowledge bases of expert systems", Object systems, 2016, no. 12, pp. 48–54. (in Russ.).

[12] G. J. Nalepa, A. Ligeza and K. Kaczor, "Formalization and modeling of rules using the XTT2 method", International Journal on Artificial Intelligence Tools, 2011, vol. 20, no. 6, pp. 1107–1125.

[13] K. Ishikawa, Guide to Quality Control. Asian Productivity Organization, 1991.

[14] CLIPS: A Tool for Building Expert Systems. (2016). [Online]. Available: http://www.clipsrules.sourceforge.net.

[15] I. V. Bychkov, N. O. Dorodnykh and A. Y. Yurin, "Approach to the development of software components for generation of knowledge bases based on conceptual models", Computational Technologies, 2016, vol. 21, no. 4. pp. 16–36. (in Russian).

[16] N.O. Dorodnykh, "Web-based software for automating development of knowledge bases on the basis of transformation of conceptual models", Open Semantic Technologies for Intelligent Systems, 2017, pp. 145–150.

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

[18] 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.