Rule creation based on decision tables in knowledge-based systems development platform

O.A. Nikolaychuk [[0000-0002-5186-0073], A.I. Pavlov [0000-0002-7753-7514] and A.B. Stolbov [0000-0001-6513-7030]

Matrosov Institute for System Dynamics and Control Theory of Siberian Branch of the Russian Academy of Sciences, 134 Lermotov st., Irkutsk, Russia
{nikoly,asd,stolboff}@icc.ru

Abstract. New results related to rule-based reasoning component of knowledge-based systems development platform are considered in the paper. Namely, the technique for the alternative way of rule creation based on the decision table approach is proposed. The data model of the decision table in the context of the platform, the transformation scheme from the decision table into rules are suggested. The implementation issues based on previously developed components are also discussed. The graphical user interface and database diagram are shown. As an illustrative example, the problem of identification of degradation processes of mechanical systems is chosen. The proposed technique is well suited for situations when there are numerous combinations of the facts of quite large set of templates in condition and action parts of rules. The end-user benefits are to represent knowledge in a tabular form that is convenient for analysis and evaluation, as well as to reduce the number of actions when creating a rule by the description of repeated actions in the transformation scheme.

Keywords: Knowledge-based system, Decision table, Rule-based reasoning.

1 Introduction

Knowledge-based systems (KBSs) are commonly accepted valuable tools for solving poorly formalized problems by means of structuring and integrating information for its intelligent processing with the help of a set of designed models and methods. During recent years the authors are creating a knowledge-based systems development platform (KBSDP) with the following key features: web-oriented component-based open client-server architecture [1]; a possibility of external problem-oriented tools integration into applied KBS; central role of the conceptual model in the development process; workflow approach utilization for integrating and assembling components into applied KBS [2].

Despite the initial research motivation for KBSDP creation, up to date, more than 10 system, core, applied and auxiliary components are under development. The system components provide common software functionality adopted for knowledge-

based systems: the data control component provides operations for unified data manipulation and it is currently implemented for PostgreSQL database; the communication component supporting bidirectional client-server interaction; the component for building graphical user web interface containing an extensible library of control elements with predefined templates of client-server interactions; the workflow component can be used either for creation assembling scheme of platform components or for representing some imperative behavior used in rules. Utilizing system components features the core components provide knowledge processing functionality. The subject domain model design component is about all the stuff (data storing, editing, visual representation, etc.) concerning operations with conceptual models (concept, relation, attribute, instance). The rule-based reasoning component carries out creating knowledge bases with the selected conceptual model as an initial source of information for designing fact templates. This component provides original visual tool for rule design, code generation and executing reasoning based on Drools and Clips engines. Auxiliary components are developed to facilitate the processing of special types of information: geodata and multidimensional data sets. The applied components cover such subject area as unique mechanical system [3], agent-based simulation modeling [4], infrastructure logistics [1, 2], scenario analysis of ecological problems [5].

In the current paper, the important problem of knowledge acquiring in relation to the rule creation task is considered. As stated in the authors' publications [1] the rule is visually constructed by the user based on the previously obtained conceptual model. In such a way, rules with different types of conditions, including a comparison of the slot values with other facts or their slot values can be designed. Moreover, in action part of the rule, a call to methods of some registered in the context of the platform problem-oriented component can be also performed. So, the currently available functionality of rule-based reasoning component is rather general and allows one to design complex rules integrating external data and knowledge processing procedures.

However, there are a lot of situations when all capabilities of the component are not necessary to solve the user problem: both on the side of the expressive power of the rules and on the side of their visual construction. To address this problem, the platform is currently being refined for supporting well-known knowledge acquiring methods. As for the current paper, rule creation technique based on the particular knowledge acquiring method – decision table approach – is proposed.

Decision table has been used since the 1960s in the process design automation, software testing [6], control problems of complex technical objects [7], and others. The decision table is an effective representation of knowledge if the rules differ slightly from each other in condition or action parts. Currently, decision tables are used to create fuzzy production knowledge bases [7, 8].

The rest of the paper is organized as follows. The decision table data model, transformation scheme, and implementation issues are consistently considered in the next sections. Then as an illustrative example, a fragment of the knowledge base for identifying the mechanisms of degradation processes of mechanical systems is considered.

2 Decision table and its data model

2.1 Decision table approach overview

A decision table (DT) is a method for compact representation of a model with complex logic. The formal definitions of DT can be found, for example in [9]. Similar to rules DT represent a relationship between conditions and actions. In general DT is divided into four quadrants as shown in table 1. Condition and action options are alternatives from appropriate list. For example, in simple case option can be a Boolean value (0/1 or yes/no). Also note, that action options can be elementary or refer to other decision tables.

Table 1. Typical decision table structure.

Conditions	Condition options
Actions	Action options

The advantage of DT from the user viewpoint is the ability to construct the logic from opposite start points: either determining an action set for considered conditions (data-driven approach) or defining under which conditions an action occurs (goal directed approach). Note also that unlike procedural programming languages, DT can establish the relationship between a set of independent conditions and actions.

2.2 Decision table data model

To represent the DT, it is suggested a data model in which each table M^{DT} is associated with a set of values for table elements ($Value^{TableElement}$) and rules (*Rule*). The value of a table element is a text string with a description. Each rule consists of a set of same elements (*TableElement*) that describe both a condition and an action. A table element is described by the following set of properties: element type (*Kind*), value ($Value^{TableElement}$), and status (*Status*). Summarizing the previous ideas, the decision table data model can be expressed as follows:

$$M^{DT} = <\{Value^{TableElement}\}, \{Rule\}>, \quad (1)$$

$$Value^{TableElement} = <Constant^{Literal}, Description, Translation>, \quad (2)$$

$$Rule = <NameOfRule, \{TableElement\}>, \quad (3)$$

$$TableElement = <Kind, Value^{TableElement}, Status>, \quad (4)$$

$$Kind = <Condition \mid Action>, \quad (5)$$

$$Status = <Yes \mid No \mid Undefined>. \quad (6)$$

To ensure model transformation, the following information must be described:

$$Translation = <Kind, NameOfTemplate, NameOfSlot, \{ValToCopy\}>, \quad (7)$$

ValToCopy=<NameOfTargetSlot, NameOfSourceTemplate, NameOfSourceSlot>. (8)

The proposed model allows one to create DT of any dimension and provides an opportunity to construct rules by inputing only the values that are significant for this rule, skipping the rest.

3 Rule creation based on decision table

Based on the information from the obtained decision table, it is proposed a technique to automatically generate a set of rules. The technique applies the model of the rule from KBSDP [1] as the target model:

$$<Rule> = <\{Condition\}, \{Action\}>,$$

$$<Condition>=<FactExistence, \{RestrictionOnValueOfSlot\}>,$$

$$<RestrictionOnValueOfSlot> = <NameOfSlot, CompareOperator, ValOfCondition>,$$

$$<CompareOperator> = <Equal \mid Not\ equal \mid Less, Less\ or\ equal \mid Greater \mid Greater\ or\ equal>,$$

$$<ValOfCondition> = <Constant^{Literal} \mid FactReference \mid SlotReference>,$$

$$<SlotReference> = <FactReference, NameOfSlot>,$$

$$<Action> = <CreationOfNewFact \mid ModificationOfExistedFact \mid RemovalOfFact \mid ExternalMethodInvocation>.$$

The structure of the obtained rules depends on the number and kind of table elements they own. So, a condition may consist of one or more restrictions on the value of the fact slot, and the action may contain one or more new fact creation instruction as shown in Table 2.

Table 2. Transformation of decision table elements into rule elements.

Decision table element		Obtained rule
TableElement with *Kind = Condition* + *Translation*	- - >	Condition for the presence fact of the template = *NameOfTemplate*, where $Value^{TableElement}$ used as a restriction on the value the slot = *NameOfSlot.*
TableElement with *Kind = Action* + *Translation*	- - >	Action of creation a new fact of the template = *NameOfTemplate*, where $Value^{TableElement}$ used as the value of slot = *NameOfSlot.* Additionally, for each item of the *ValToCopy* array the slot of new fact = *NameOfTargetSlot* copy value

		from slot = *NameOfSourceSlot* of the existing in rule condition fact of template = *NameOfSource*Template.

To organize the process of converting a decision table element into a parts of a rule, each value of the table element must have a description of the conversion scheme (eq. 7), containing the name of the knowledge base template and the name of the slot that should be used when elements with this value are processing.

Also note, that the conditions and actions of the conversion scheme are set independently. And for elements with the "action" view, the conversion scheme can be supplemented with information about the slots of the created fact, the values of which must be copied from the facts that exist in the rule condition (eq. 8). The proposed technique of forming rules allows one to perform a single entry of repetitive information and, thus, automate the process of creating the same type of rules.

4 Implementation issues

The proposed decision table data model was implemented using the PostgreSQL DBMS. The structure of the "Decision tables" schema is shown on figure 1. External table "Rules" from the schema "Production knowledge bases" is highlighted by white color.

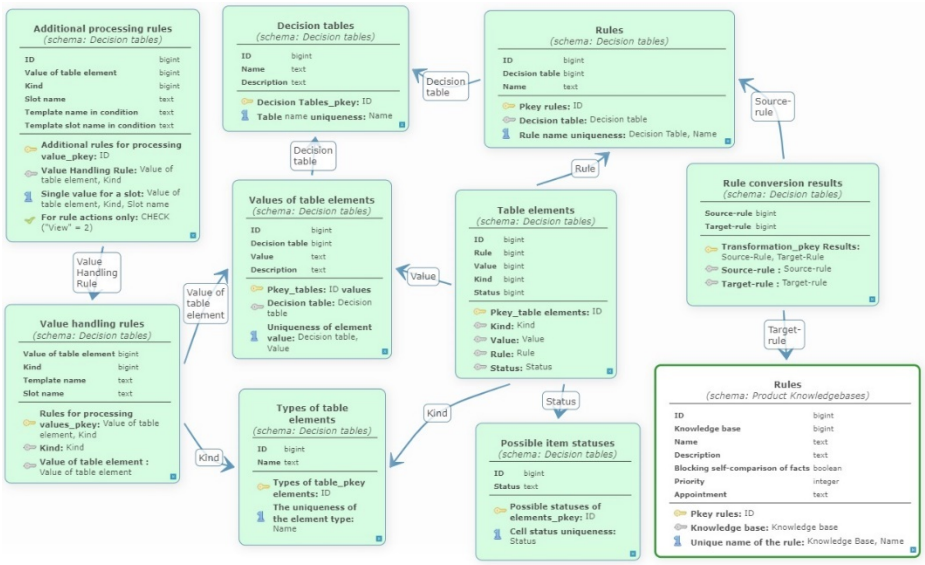


Fig. 1. The data model for storing decision tables.

Note, that the considered figure is obtained automatically by the data control component of KBSDP.

The visual interface supporting all rule creation process based on proposed technique is done on the top existing KBSDP functionality. Namely, the data control component [1] and the data representation and editing component [1] are utilized.

The task of creating and completing DT is fully supported by existing functionality. In particular, it is used a function for inputting data by a range of values, which in some cases allows one to input data for all conditions or all actions during single request towards a column (rule) in the decision table. This situation is illustrated on the figure 2.

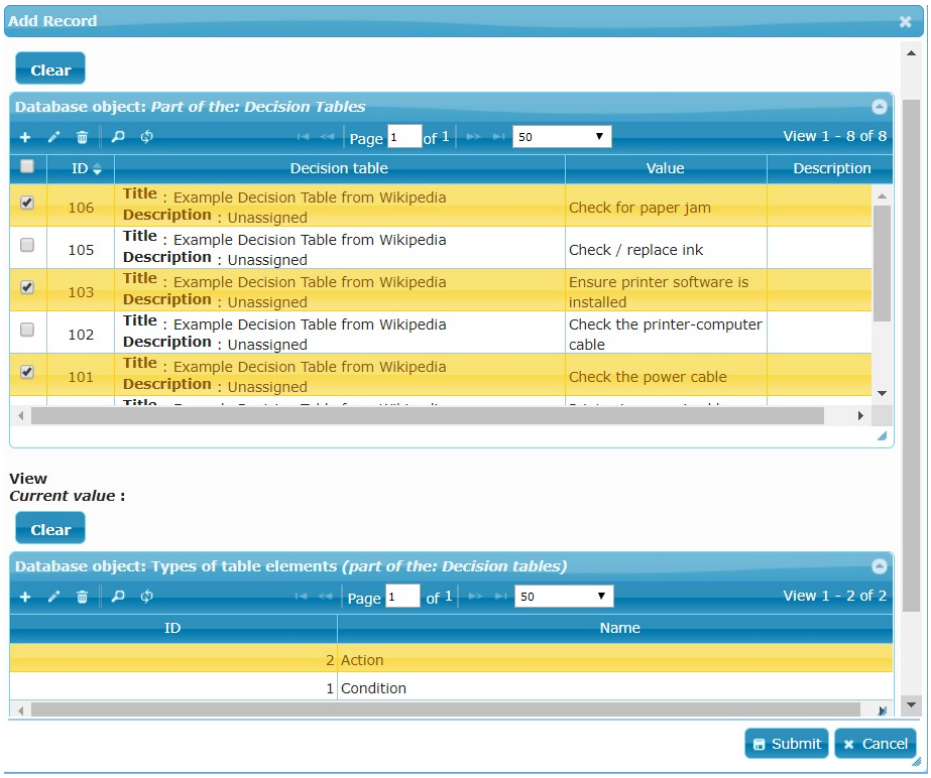


Fig. 2. Example of ranged input of values.

To view the ready-made DT, a specialized form (fig. 3) of the user interface is used. It is based on the standard "dialog" control of the KBSDP.

Inputs	Item type	Rule-1	Rule-2	Rule-3	Rule-4	Rule-5	Rule-6	Rule-7	Rule-8
Printer prints	Condition	Not	Not	Not	Not	Yes	Yes	Yes	Yes
A red light is flashing	Condition	Yes	Yes	Not	Not	Yes	Yes	Not	Not
Printer is recognized by computer	Condition	Not	Yes	Not	Yes	Not	Yes	Not	Yes
Check the power cable	Action	-	-	Yes	-	-	-	-	-
Check the printer-computer cable	Action	Yes	-	Yes	-	-	-	-	-
Ensure printer software is installed	Action	Yes	-	Yes	-	Yes	-	Yes	-
Check / replace ink	Action	Yes	Yes	-	-	-	Yes	-	-
Check for paper jam	Action	-	Yes	-	Yes	-	-	-	-

Fig. 3. Example of decision table

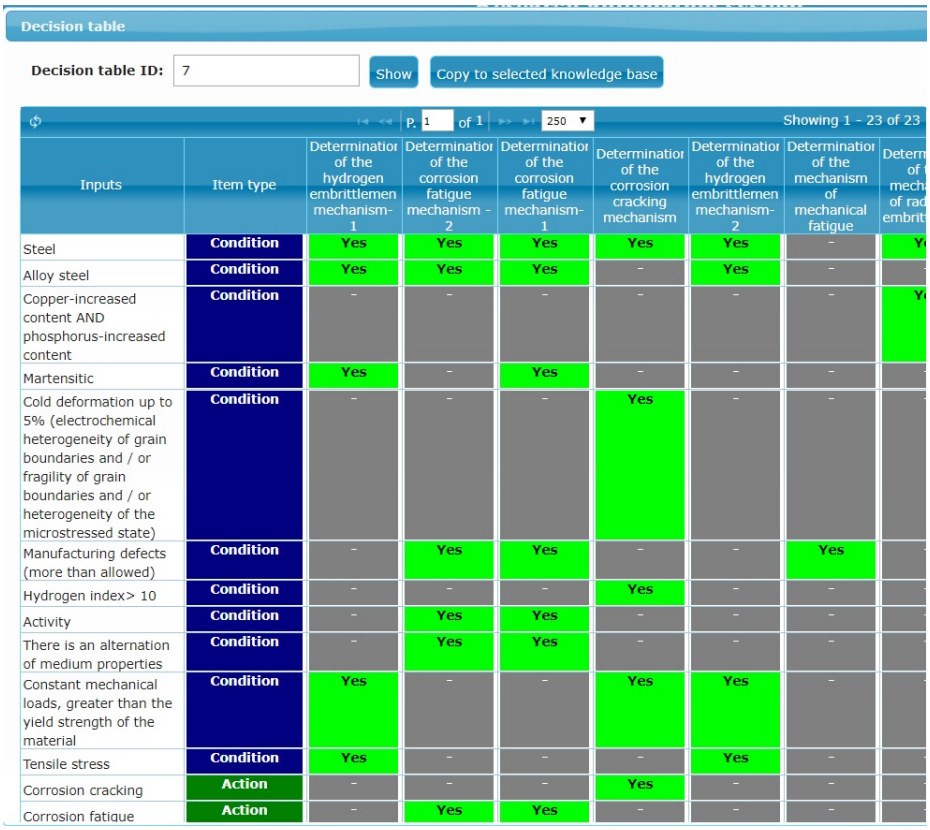
Thus, the algorithm of technique, i.e. the algorithm for rule creation based on a decision table can be divided into three main stages:

1. Forming a decision table. At this stage, one inputs a name and description, creates a list of possible values for table elements, and describes the rules.
2. Forming rules for processing values. At this stage, relationships between the values of the decision table and the fact slots of the selected knowledge base templates are established.
3. Generating rules in the format of a rule-based reasoning component. At this stage, rules are automatically created.

5 An illustrative example

To demonstrate the proposed forming rules technique capabilities, a fragment of the knowledge base for identifying the mechanisms of degradation processes of mechanical systems is implemented [10]. The principle algorithm of technique application includes the following stages.

At first, DT must be created (see Fig. 4).



Inputs	Item type	Determinatior of the hydrogen embrittlemen mechanism-1	Determinatior of the corrosion fatigue mechanism - 2	Determinatior of the corrosion fatigue mechanism-1	Determinatior of the corrosion cracking mechanism	Determinatior of the hydrogen embrittlemen mechanism-2	Determinatior of the mechanism of mechanical fatigue	Determ of mecha of rad embrit
Steel	Condition	Yes	Yes	Yes	Yes	Yes	-	Y
Alloy steel	Condition	Yes	Yes	Yes	-	Yes	-	
Copper-increased content AND phosphorus-increased content	Condition	-	-	-	-	-	-	Y
Martensitic	Condition	Yes	-	Yes	-	-	-	
Cold deformation up to 5% (electrochemical heterogeneity of grain boundaries and / or fragility of grain boundaries and / or heterogeneity of the microstressed state)	Condition	-	-	-	Yes	-	-	
Manufacturing defects (more than allowed)	Condition	-	Yes	Yes	-	-	Yes	
Hydrogen index> 10	Condition	-	-	-	Yes	-	-	
Activity	Condition	-	Yes	Yes	-	-	-	
There is an alternation of medium properties	Condition	-	Yes	Yes	-	-	-	
Constant mechanical loads, greater than the yield strength of the material	Condition	Yes	-	-	Yes	Yes	-	
Tensile stress	Condition	Yes	-	-	-	Yes	-	
Corrosion cracking	Action	-	-	-	Yes	-	-	
Corrosion fatigue	Action	-	Yes	Yes	-	-	-	

Fig. 4. Example of decision table of the mechanisms of degradation processes.

Then, a set of related rules is generated based on the defined conversion scheme. The rule example corresponding one of the mechanisms of degradation processes is shown on Fig. 5.

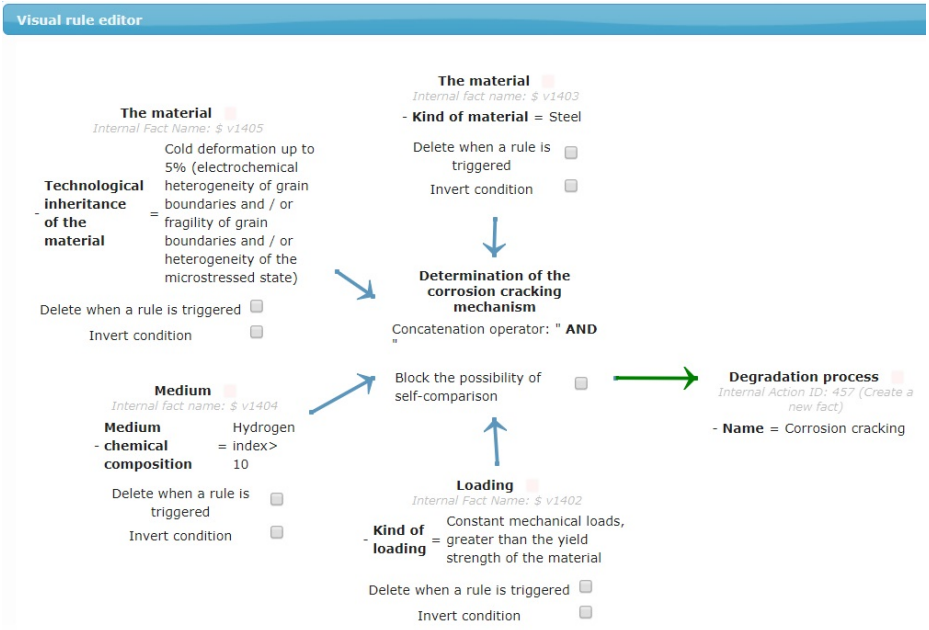


Fig. 5. Example of one column of decision table represented as a rule in the rule-based reasoning component.

Finally, the knowledge base code generation is performed. Further the Drools example is presented:

```
rule "Determination of the corrosion cracking mechanism"
 @IDOfRule(R448)
 dialect "mvel"
 when
  $v1405: T2116(s945 == "Cold deformation up to 5% (electrochemical  heterogeneity of grain boundaries and / or fragility of grain boundaries  and / or heterogeneity of the microstressed state)") /* The material */
  $v1404: T2117(s947 == "Hydrogen index > 10") /* Medium */
  $v1403: T2116(s941 == "Steel") /* The material */
  $v1402: T2118(s939 == "Constant mechanical loads, greater than the yield  strength of the material") /* Loading */
 then
  $nf_v457 = new T2119();
    $nf_v457.s937 = "Corrosion cracking";
  insert($nf_v457);
end
```

6 Conclusions

The technique for automatic generation a rule set based on the decision table approach is proposed in the paper. The technique is implemented as an extension of the rule-based reasoning component of knowledge-based systems development platform. An example of applying this technique to the creation of the knowledge base about degradation processes is shown.

This technique is necessary to improve the efficiency knowledge bases creation process by non-programming domain specialists. Application of the technique is promising, on the one hand, when it is necessary to describe the domain knowledge, where a sufficiently large set of facts is repeated in various combinations in a variety of rules. On the other hand, the implementation of this technique expands the users' capabilities, providing them with the choice of the most convenient tool depending on their skills and experience.

7 Acknowledgments

The research was supported by the Program of the Fundamental Research of the Siberian Branch of the Russian Academy of Sciences, project no. IV.38.1.2 (reg. no. AAAA-A17-117032210079-1). Results are achieved using the Centre of collective usage «Integrated information network of Irkutsk scientific educational complex».

References

1. Nikolaychuk, O.A., Pavlov, A.I., Stolbov, A.B.: The software platform architecture for the component-oriented development of knowledge-based systems. In: Proceedings of the 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), pp. 1234–1239. IEEE, Opatija Croatia (2018). https://doi.org/10.23919/MIPRO.2018.8400194
2. Pavlov, A.I., Stolbov, A.B., Dorofeev, A.S.: The workflow component of the knowledge-based systems development platform. In: Bychkov, I.V., Karastoanov D. (eds.) Proceedings for 2nd Scientific-practical Workshop Information Technologies: Algorithms, Models, Systems (ITAMS), CEUR Workshop Proceedings, vol. 2463, pp. 47–58. CEUR-WS.org, Irkutsk Russia (2019).
3. Nikolaychuk, O.A., Pavlov, A.I., Stolbov, A.B.: The Agent-Based Modeling Method For The Study Of Unique Mechanical Systems. Advances in Intelligent Systems Research, 166, 201–206 (2019). https://doi.org/10.2991/itids-19.2019.36
4. Pavlov, A.I., Stolbov, A.B., Dorofeev, A.S.: The architecture of the software tool for the agent-based simulation models specification development. In: Bychkov, I.V., Karastoanov D. (eds.) Proceedings for 2nd Scientific-practical Workshop Information Technologies: Algorithms, Models, Systems (ITAMS), CEUR Workshop Proceedings, vol. 2463, pp. 112–121. CEUR-WS.org, Irkutsk Russia (2019).
5. Pavlov, A.I., Stolbov, A.B.: The Application of the Knowledge-Based Systems Development Platform for Creating Scenario Analysis Support Tools. Advances in Intelligent Systems Research, 169, 43–48 (2019). https://doi.org/10.2991/itids-19.2019.36

6. Copeland, L.: A Practitioner's Guide to Software Test Design. Artech House Publishers (2003).
7. Grevtsov, N.M., Perchitz, S.N., Fedunov, B.E., Yunevich, N.D.: Intellectual support of commander of escort jet fighter group in solving the task for the returning subgroup, which repulsed an attack of enemy jet fighters. Journal of Computer and Systems Sciences International 57 (4), 608-619 (2018).
8. Eremeev, A.P. Vinogradov, O.V.: Software tools for decision support based on fuzzy tabular models of knowledge representation. Software & systems 4, 22 (2008).
9. Wets, G.: Decision tables in knowledge-based systems : adding knowledge discovery and fuzzy concepts to the decision table formalism. Technische Universiteit Eindhoven (1998). DOI: https://doi.org/10.6100/IR510201
10. Berman, A. F.: Degradation of mechanical systems. Publisher “Nauka”, Novosibirsk (1998).