Models and algorithms for the formation of an event tree used in analysing the quality of system software for knowledge-based intelligent decision support systems

Kolomiitsev Oleksii¹, Osilevskyi Serhiï², Tretiak Viacheslav³, Voronin Viktor⁴, Komarov Volodymyr⁵, Pustovarov Volodymyr⁶, Heiko Hennadii⁷, Brechko Veronika⁸, Kovalchuk Ihor⁹, Prysiazniuk Viktoriia¹⁰, Rybalchenko Alina¹¹, Rudakov Ihor¹²

¹ Honored Inventor of Ukraine, Doctor of Technical Sciences, Professor at the Department of National Technical University; Kharkiv Polytechnic Institute; Ukraine
² Candidate of Technical Sciences, Associate Professor; Department Head of Ivan Kozhedub Kharkiv National Air Force University; Ukraine
³ Candidate of Technical Sciences, Associate Professor, Lead Researcher; Ivan Kozhedub Kharkiv National Air Force University; Ukraine
⁴ PhD in Engineering Associate Professor, Senior Researcher; Ivan Kozhedub Kharkiv National Air Force University; Ukraine
⁵ Honored Inventor of Ukraine, Candidate of Technical Sciences, Leading Researcher; Military Institute of Telecommunication and Information technologies named after the Heroes of Kruty; Ukraine
Abstract.
A formal statement of the problem of improving the quality of software for intelligent systems under unfavourable circumstances is made. Its belonging to the class of problems of the calculus of variations on a conditional extremum is established. A ranked approach based on heuristic algorithms to solve this scientific problem is proposed and substantiated.

Keywords:
software quality
intelligent decision support system
In the current environment of information technology development, there is an increase in requirements for software developed for various sectors of human activity. Intelligent decision support systems (IDSS) are used in complex production systems, management of various processes, traffic control of large transport systems, etc. The purpose of their use is to help decision makers in difficult conditions to make a complete and objective analysis of the subject area. The quality of software for such systems is characterised by the ability of the software product to meet certain needs under specified conditions [1].

At present, the basic provisions of quality theory in relation to intelligent information systems, systems using cloud computing technologies, as well as information systems designed to solve large-scale management tasks, are not sufficiently effective. The main reason is that the models and methods of classical quality theory cannot describe and provide adequate assessments of objects whose performance is deteriorated not only by related physical failures, but also by design errors, information influence and errors in knowledge bases (KB). For such objects, it is difficult to define the very concept of failure and identify the full range of causes that generate these failures.

On the other hand, the development of IDSS today is often carried out by specialists of informal engineering philosophy (the so-called new formation using hybrid methodologies), whose views on software development, analysis and results are somewhat "artistic" in nature, but not containing scientific substantiation [2].

That is, there is a contradiction between the comprehensive growth of decision-making automation capabilities in poorly formalised areas through the use of knowledge-based information systems (KBIS) and the actual compliance with the quality requirements for intelligent information systems.

The analysis of the influence of KBIS software errors on the functioning of software systems has shown that such errors cause up to 70% of their failures [4-7]. On the other hand, the intensification of the development of knowledge processing methods in KBIS actually provokes a sharp increase
in their intellectualisation, i.e. the volume of KB increases, which requires the parallel development of effective methods of quality control of special KBIS software.

Taking into account the fact that the European Commission for Electrical Engineering has adjusted and supplemented the RAMS Industry Standards with requirements for software of decision-making software systems (which contain recommendations for the application of a comprehensive software quality criterion), it is worth talking about the relevance of the issues of quality control of the SS of the KBIS.

Certain solutions to the problem of developing methods for assessing the quality of IDSS software and debugging the software are described in detail in the works of such scientists as: Lipaev V.V., Zyкова S.A., Narignani A., Scott, Suvi, Nguyen, Perkins, Tepandi, Kragun, Stendel [8-14], etc. From the analysis of these works and publications, it is clear that the greatest success has been achieved in the development of methods of static debugging of the KB, i.e., those methods that do not require launching the KBIS for execution.

However, to date, there is no single approach to formalising the so-called structural errors detected by static debugging methods: redundancy, incompleteness and contradiction. That is why statistically correct KBs do not guarantee the quality of decisions made due to errors in the knowledge itself, often related to the complexity of a particular subject area, which, for example, allows for duplication of reasoning logic. The most difficult to detect are errors such as "forgetting about exceptions" and "critical combination of events", which, in turn, leads to errors in decision-making.

Testing is the way to detect errors in knowledge. However, to date, there have been only a few attempts to automate the construction of tests to detect these classes of errors. Despite the development of hybrid KBIS that combine different types of knowledge representation, the basis of most modern systems is the productive approach.

Another advanced way to create an KBIS is to use a three-layer perceptron as a type of artificial neural network, for which there is no single approach to debugging.
Continuous increase in the complexity of the functions implemented by software in the KBIS leads to an increase in their volume and labour intensity of development. In accordance with the change in software complexity, the number of detected and undetected defects and errors that affect the functioning of the software system as a whole increases. Complex KBIS with millions of lines of programme code and tens of thousands of rules in the KBs cannot be error-free in principle.

In [15], it is shown that "the problem of detecting and eliminating errors becomes more acute as the complexity of the tasks performed by software applications increases and threatens to cause disasters in systems that perform critical control functions for large, expensive and particularly important objects or processes".

Thus, the solution of the problem of assessing the quality of KBIS software in order to improve the efficiency of human-machine systems (HMS) is relevant.

Based on the above-mentioned existing solutions and theoretical provisions of the theory of databases (DB) and KBs, the main task is to develop a practical mechanism. Such a mechanism is based on the synthesis of a comprehensive plan of operations to improve the quality of software and the analysis of the possibility of implementing this plan under various control influences, as well as environmental conditions, followed by the construction of models and algorithms for synthesising this plan in the form of an event tree (based on elements of Boolean algebra and the rank approach). All this will allow to find ways to reach the top of the first rank (root) of the event tree and determine the minimum number of combinations of events that can cause a defect in the software IDSS.

The solution of this problem is proposed to be based on the methods of solving the problem of managing the KBIS by the vector criterion. This is primarily due to the possibility of analysing and managing such characteristics of the KBIS SS as the quality of functioning, design and operation costs, degree of flexibility, possibility of modernisation, etc. [3].

Taking into account the above, let us formulate a partial task of improving the quality of KBIS software as a parameter
The proposed solutions are focused on meeting the requirements of NSoU ISO/IEC 9126 Software Engineering. Product quality (parts 1-4) [16] and NSoU ISO/IEC 14598 Information technologies for software product evaluation (parts 1-6) [17], which supplement the quality model and put the evaluation process in a separate group of NSoU ISO/IEC standards. In addition, the paper takes into account the requirements for the SQuaRE (Software Quality Requirements and Evaluation) series of standards, which defines a model of software quality characteristics and consists of such quality criteria as:

- internal quality criteria, i.e. requirements for code quality and internal architecture;
- external quality criteria, i.e. requirements for functionality;
- quality criteria in use, i.e. those criteria that are set not only for the software, but also for the whole of the IDSS.

In accordance with these documents, the quality of software, including IDSS software, is understood as a set of properties that characterise the ability to maintain its level of quality of functioning under the established conditions for a given period of time.

Based on this definition of quality, the integral expression of the efficiency criterion of the task to be solved is the function

\[ P_0(\bar{x}(t), \bar{u}(t)) \]

which characterises the probability of maintaining the required level of software quality at a given time interval \([t_i, t_f]\) under an unfavourable set of circumstances.

An unfavourable set of circumstances means the occurrence of such a sequence of events in the course of the functioning of the KBIS, each of which individually does not have a significant impact on the quality of the software, but in aggregate leads to a significant decrease in its quality.

Taking into account the above, the task of improving the quality of software functioning in the development and operation of KBIS is to develop an algorithm for determining...
such control influences \( \overrightarrow{u}(t)^* \in \overrightarrow{U}(t) \), the implementation of which, under any permissible environmental conditions \( \overrightarrow{x}(t)^* \in \overrightarrow{X}(t) \) will maximise the criterion:

\[
K_1 = \int_{t_i}^{t_f} P_0(\overrightarrow{x}(t), \overrightarrow{u}(t)) dt,
\]

which characterises the probability of maintaining the required quality of the KBIS software functioning over a time interval \([t_i, t_f] \). 

In a formalised form, the task of improving the quality of IDSS software has the following formulation:

\[
K_1 = \int_{t_i}^{t_f} P_0(\overrightarrow{x}(t), \overrightarrow{u}(t)) dt \rightarrow \max,
\]

under restrictions:

\[
F_i(\overrightarrow{x}(t), \overrightarrow{u}(t)) \geq 0, i = 1, n_1;
\]
\[
F_i(\overrightarrow{x}(t), \overrightarrow{u}(t)) < 0, i = 1, n_2.
\]

boundary conditions:

\[
F_i^{(t)}(\overrightarrow{x}(t), \overrightarrow{u}(t)) \geq 0, i = n_2 + 1, n_3;
\]
\[
F_i^{(i)}(\overrightarrow{x}(t), \overrightarrow{u}(t)) = 0, i = n_3 + 1, n_4,
\]

due to the specifics of the object of management specified in [18], where \( n_1 - n_4 \) - known constants;

\( \overrightarrow{X}(0) \) and \( \overrightarrow{U}(0) \) - are the regions of permissible vector values
\( \tilde{u}(t) \) and \( \tilde{x}(t) \) respectively.

The developed algorithm for synthesising the vector of control influences \( \tilde{u}(t) \), is based on the statement that to solve the problem of maximising criterion (1), it is enough to develop and implement a detailed comprehensive plan of operation \( PL(t) \) to assess the quality of functioning of the KBIS SS [19].

The software implementation of the functions of individual elements (software modules) of solving a problem in IDSS is proposed to be represented by means of an event tree, which is a connected acyclic graph whose vertices correspond to individual operations of the problem-solving algorithm (sequences of implementation of functions and procedures), and the edges of the graph characterise hierarchical cause-and-effect relationships set on the set of its vertices - the set of vertices and edges of the graph, respectively.

Events are individual operations of the plan that can be executed or not executed (partial execution is not taken into account). In essence, the graph is a formalised description of the plan of operations.

Building an operation plan to ensure the appropriate quality of software functioning is a rather complex and time-consuming task, the solution of which must be carried out for each subject area of the KBIS, taking into account the peculiarities of its functioning, operating experience and expert opinion. At the same time, it is advisable to use well-known models and methods of software project management, which allow to increase their efficiency and reduce implementation costs [20].

Based on the above, we propose a certain modified heuristic procedure for building an operations plan based on the requirements of the NSoU ISO/IEC group of standards. It consists of the following main stages:

- a graph of hierarchical cause-and-effect relationships between individual elements of the KBIS software for which the task is being solved is built [21]. The main software
modules of this system that affect the quality of its functioning are determined;

- for each software module, in accordance with [16, 17], characteristics, subcharacteristics and properties (indicators) of the quality of SS functioning are selected. As a result, a graph of hierarchical causal relationships between the characteristics and indicators of the quality of functioning of software modules is formed. To build this graph, it is proposed to use the provisions of graph theory with the use of the rank approach, where characteristics, subcharacteristics and properties have the appropriate rank. The basic principles of using the rank approach to construct graphs are disclosed in [22];

- graph describes a set of cause-and-effect relationships that exist between the quality indicators of software modules and the most common types of KBIS software errors that affect the quality of its functioning. The most common types of software errors, as well as their impact on the quality indicators of software modules, are discussed in detail in [23].

The graph of an event tree is determined from the union of graphs. A fragment of the graph is shown in Figure 1.

The edges of this graph are conjunctive edges. A formal description of the vertices of the graph fragment shown in Figure 1 (events) is given in Table 1.

The constructed event tree is the basis for the development of models and algorithms for quantifying the possibility of executing the operations plan when searching for a solution to the problem (2).

This task can be solved in the following ways:

- finding an approximate solution based on the indicator function constructed using Boolean logic. Such a solution allows to establish the fact that a certain quality indicator does not correspond to the specified values and can be applied to short backlogs of hybrid software application development technology;

- finding an accurate solution with the determination of the factor influencing the obtained value of the quality indicator and the formation of forecast decisions on its impact on the quality of software both at the subsequent
stages of software development and on the final product. The obtained predictive values will allow adjusting the software development process without returning to the previous stages, which will ultimately significantly reduce the time for software development while ensuring compliance with quality requirements.

![Figure 1](image)

**Figure 1**

Fragment of the event tree graph $G_{cl}

It is the second way that is of particular interest and reveals multifaceted approaches to its implementation. In this paper, we propose to use the Kolmogorov-Champman differential equations solution apparatus to solve the problem.
Table 1

<table>
<thead>
<tr>
<th>Node number</th>
<th>Process</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha_1$</td>
<td>Functional features</td>
</tr>
<tr>
<td>$\alpha_2$</td>
<td>Eligibility</td>
</tr>
<tr>
<td>$\alpha_3$</td>
<td>Properness</td>
</tr>
<tr>
<td>$\alpha_4$</td>
<td>Ability to interact</td>
</tr>
<tr>
<td>$\alpha_5$</td>
<td>Consistency of the system in general</td>
</tr>
<tr>
<td>$\alpha_{34}$</td>
<td>Module consistency</td>
</tr>
<tr>
<td>$\alpha_{35}$</td>
<td>Failure to perform the functions specified in the requirements specification</td>
</tr>
<tr>
<td>$\alpha_{36}$</td>
<td>Non-compliance of software documentation with the specification</td>
</tr>
<tr>
<td>$\alpha_{47}$</td>
<td>Documentation shortcomings</td>
</tr>
<tr>
<td>$\alpha_{48}$</td>
<td>Errors when using data from other software tools</td>
</tr>
<tr>
<td>$\alpha_{49}$</td>
<td>Data and file format incompatibility</td>
</tr>
<tr>
<td>$\alpha_{50}$</td>
<td>Disadvantages of access controls</td>
</tr>
<tr>
<td>$\alpha_{51}$</td>
<td>Disadvantages of accounting and information registration tools</td>
</tr>
<tr>
<td>$\alpha_{52}$</td>
<td>Disadvantages of protecting information from distortion</td>
</tr>
<tr>
<td>$\alpha_{53}$</td>
<td>Disadvantages of protecting information from distortion</td>
</tr>
<tr>
<td>$\alpha_{54}$</td>
<td>Non-compliance with the requirements of the cryptographic tools used</td>
</tr>
<tr>
<td>$\alpha_{55}$</td>
<td>Reliability errors</td>
</tr>
</tbody>
</table>

Thus, a formal formulation of the problem of improving the efficiency of HMS functioning as a vector optimisation problem with constraints in the form of equations and inequalities has been developed.

It is established that in order to solve it, it is necessary to develop mathematical models, numerical methods, algorithms and software packages that will allow optimising the objective function that characterises the quality of software of intelligent systems at different time intervals.

A formal statement of the problem of improving the quality of the IDSS software under unfavourable circumstances is made. Its belonging to the class of problems of the calculus of
variations on a conditional extremum is established.

A rank-based approach based on heuristic algorithms to solve this problem is proposed and substantiated.

It is established that the task of improving the quality of software can be reduced to the task of synthesising a comprehensive plan of operations to improve the quality of software and analysing its feasibility under different control influences and environmental conditions.

References:


This work is distributed under the terms of the Creative Commons Attribution-ShareAlike 4.0 International License (https://creativecommons.org/licenses/by-sa/4.0/).