Розвиток радіотехнічного забезпечення, АСУ та зв'язку Повітряних Сил

UDC 004.052

DOI: 10.30748/nitps.2020.41.15

O. Turinskiy, M. Pavlenko, H. Pievtsov, S. Osiievskyi

Ivan Kozhedub Kharkiv National Air Force University, Kharkiv

THE METHOD OF INCREASING THE EFFICIENCY OF THE MAN-MACHINE SYSTEM FUNCTIONING BY IMPROVING THE QUALITY OF DECISION SUPPORT SYSTEM SOFTWARE

The article addresses improving the performance of man-machine systems (MMS) by improving the quality of decision support system software (DSSS). The proposed solutions are based on existing models and methods, which have been analysed in detail in a scientific study, in particular the application of methodical static application debugging in certain solutions obtained. It is proposed to use the methods for solving the problem of MMS control according to the vector criterion as the basis for solving this problem. A mechanism for synthesizing a comprehensive plan of operations to improve the quality of the DSS software is proposed, and the possibility of analysing the step-by-step implementation of a plan to obtain an analytical assessment is shown. For this purpose, a comprehensive plan is presented in the form of a relevance tree, built in accordance with the provisions of graph theory and the rank approach. The time complexity of the algorithms implementing the relevance tree tasks was evaluated. Based on the formalized solutions obtained, it was found that the maximum number of edges for a graph that implements the relevance tree does not exceed n^2 , the number of views of edges does not exceed n , that is, the total number of operations that implements the scanning of edges and adding them to the weighted minimum cover (WMC) does not exceeds n^3 . It is established that during the implementation of the operation of adding ribs in the WMC, at each step, after the fixation of the previous result, the number of ribs to be examined decreases, so the total complexity of the algorithms will not exceed n^3 . The solutions obtained meet the requirements of DSTU ISO/IEC 9126, ISO DSTU / IEC 14598, and take into account the requirements of the Software Quality Requirements and Evaluation series of standards as the value of the vertices of the event tree graph. In the process of solving the problem, the specificity of the functioning of the MMS was taken into account, the possibility of formalizing various aspects of knowledge (aesthetic, disizional, causal, diontic) and ensuring a given level of efficiency in finding solutions.

Keywords: software, decision support system, graph, relevance tree.

Introduction

Problem statement. Comprehensive expansion of decision-making automation capabilities in poorly formalised areas. through the use of expert knowledge. leads to an increase in the number of errors in decision support system software (DSSS). And as a consequence, there is an increase in possible sources of failures (both software and hardware). An analysis of DSSS errors on the operation of software complexes in general showed that such errors caused up to 70% of their failures [1-4]. On the other hand, the intensification of the development of knowledge processing methods in DSS actually provokes a sharp increase in their intellectualization, that is, the volume of knowledge bases increases, and requires the parallel development of effective quality control methods for special DSSS. From the point of view of the fact that the European Electrotechnical Commission has made adjustments and additions to industry standards RAMS-requirements for software

decision-making software (which contain recommendations for the application of a comprehensive software quality criterion), the relevance of quality control issues for special DSS software should be discussed.

Analysis of recent studies and publications. The problem of developing methods for assessing the quality of DSSS and debugging the knowledge bases has been described in some detail in the writings of scientists such as V. Lipaev, S. Zykova, A. Narinani, Scott, Suva, Nguyen, Perkins, Tepandi, Kragun, Stendel [5–11] and others. It is clear from the analysis of these works and publications that the greatest success has been achieved in the development of static debugging methods, those methods that do not require the launching of an expert system for execution. However, there is currently no uniform approach to formalizing the so-called structural errors detected by static debugging methods: redundancy, incompleteness, contradiction. This is why the statically correct knowledge bases do not guarantee the quality of decision-making at the expense of errors in

the knowledge, often related to the complexity of the individual subject matter, which, for example, allows for duplication of reasoning logic. The most difficult for detecting errors of type "forgetting to delete" and "critical combination of events", which leads to errors in decision-making. Testing is the way to detect knowledge errors. Today there are only isolated attempts to automate the construction of tests. Despite the development of hybrid DSS combining different types of knowledge presentation, most modern systems are based on the product approach. Another developed way to create DSSS is to use trilayer personatron as a variant of artificial neural networks for which there is no single approach to debugging at all. The ever-increasing complexity of the functionalities implemented by the software in intelligent systems increases their volume and labour-intensive development. In line with the changing complexity of the software, the number of detected and undetected defects and errors is increasing, which affects the operation of the software complex as a whole. Complex millions of lines of code DSS with tens of thousands of rules in the knowledge base cannot fundamentally be unmistakable. In [12] it is shown that "the problem of detecting and fixing errors intensifies as the complexity of the tasks to be solved by the programs, and threatens to catastrophe in systems performing critical functions of controlling large, expensive and especially important objects or processes".

This is why the task of assessing the quality of the DSSS with a view to improving the performance of man-machine systems (MMS) is relevant.

Purpose of the article. Based on the aforementioned existing solutions and theoretical provisions of the theory of Data Base (DB) and Knowledge Base (KB), the solution to the main task is seen in the development of a practical mechanism, it consists in the synthesis of a comprehensive plan of operations to improve the quality of software and analysis of the possibility of implementing this plan under various control influences and states of the external environment, followed by construction of models and algorithms for the synthesis of this plan in the form of an event tree (based on elements of Boolean algebra and the rank approach), which will allow finding 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 DSS software.

Statement of basic materials

The solution to this problem is proposed to be based on methods for solving the problem of MMS control according to a vector criterion. This is primarily related to the ability to analyse and manage the performance characteristics of a dedicated special DSSS such as performance, design and operation costs, flexibility, upgrades and others. In addition, the mathemati-

cal formulation of this problem allows a more reasonable approach to the development of a methodology for achieving the goal, which is to increase the efficiency of the functioning of complex man-machine systems, from the standpoint of system analysis and to show the influence of local tasks implemented in the special DSS software modules on solving the problem of improving the functioning of man-machine systems and due to this to develop more effective models and methods that can significantly improve the quality of special DSSS.

Formally, the problem of MMS control according to a vector criterion is reduced to the development of an algorithm for the synthesis of a vector of control actions $\vec{u}(t) \in \{\vec{U}(t)\}$, the implementation of which under any admissible conditions of the environment $\vec{x}(t) \in \{\vec{X}(t)\}$ and allows maximizing the criterion $\vec{K} = (K_1, K_2, \dots, K_m)$, which characterizes the efficiency of the functioning of a man-machine system at a given time interval $[t_s, t_f]$:

$$\begin{aligned} K_1 &= \int_{t_s}^{t_f} K'_1(\vec{x}(t), \vec{u}(t)) dt; \\ K_2 &= \int_{t_s}^{t_f} K'_2(\vec{x}(t), \vec{u}(t)) dt; \\ &\dots \\ K_m &= \int_{t_s}^{t_f} K'_m(\vec{x}(t), \vec{u}(t)) dt. \end{aligned} \quad (1)$$

Under functional restrictions

$$\begin{aligned} F'_i(\vec{x}(t), \vec{u}(t)) &\geq 0, i = \overline{1, n_1}; \\ F'_i(\vec{x}(t), \vec{u}(t)) &< 0, i = \overline{1, n_2}. \end{aligned} \quad (2)$$

Boundary conditions:

$$\begin{aligned} F'_i(t_s)(\vec{x}(t), \vec{u}(t)) &\geq 0, i = \overline{n_2 + 1, n_3}; \\ F'_i(t_f)(\vec{x}(t), \vec{u}(t)) &= 0, i = \overline{n_3 + 1, n_4}, \end{aligned} \quad (3)$$

related to the specific functioning of the man-machine system. In expressions (1–3): $n_1 - n_4$ – known constants determined by the specifics of the functioning of a man-machine system; $\{\vec{X}(t)\}$ and $\{\vec{U}(t)\}$ – areas of admissible values of vectors $\vec{u}(t)$ and $\vec{x}(t)$, respectively; $K'_1(\vec{x}(t), \vec{u}(t))$ – subintegral function characterizing the quality of the software of the special purpose intelligent systems of the man-machine system; $K'_2(\vec{x}(t), \vec{u}(t))$ – subintegral function characterizing the security of data systems). Consideration of the specifics of the MMS

functioning is mandatory and justified in [13]. In particular, the specific features of the man-machine system are as follows:

- taking into account the possibility of formalizing different aspects of knowledge (aletic, dissident, causal, diontic);
- consistency property;
- ensuring a given level of completeness in the description of the subject area;
- ensuring a predetermined level of speed in the search for solutions;
- providing the ability to obtain many alternative solutions.

Therefore, taking (1–3) into account, in order to solve the given problem it is necessary first to develop mathematical models, numerical methods, algorithms and software complexes allowing to optimize one of the basic scalar components of $\vec{K} = (K_1, K_2, \dots, K_m)$ – target function $K_1'(\vec{x}(t), \vec{u}(t))$ describing the quality of the DSS software.

In view of the above, the objective of improving the quality of the DSS software is outlined.

The production of the proposed solutions is oriented towards meeting the requirements of ISO DSTU / IEC 9126 Software Engineering. Product quality (Parts 1-4) [14] and DSTU ISO / IEC 14598 Information technology for software product evaluation (Parts 1-6) [15] which complete the quality model and place the evaluation process in a separate group of ISO / IEC standards. In addition, the requirements of the Square Standard Series (Software Quality Requirements and Evaluation) (Fig. 1) are taken into account, which defines the software quality performance model as consisting of the following quality criteria:

- internal quality criteria, that is, requirements for the quality of the code and for the internal architecture;
- external quality criteria, i.e., capability requirements;
- quality in use criteria, that is, those that are established not only for software but for the entire information system.

Section on quality requirements 2503n	Section of quality model 2501n	Section of quality assessment 2504n
	Section of quality management 2500n	
	Section of quality measurement 2502n	
Section of further expansion 25050 – 25099		

Fig. 1. Organization of a series of SQuaRE standards

This is due to the fact that the SQuaRE provides a specification of software quality requirements and reduces uncertainty when organizations work together to develop, implement and maintain software, for example, between developers, developers and independent appraisers [16].

According to these documents, the quality of the software, including the DSSS of the systems, is the set of properties that characterize the ability to maintain its level of performance under the specified conditions for a given period of time.

Based on this definition of quality, the subintegral expression of the performance criterion of a problem is function $P_Q(\vec{x}(t), \vec{u}(t))$, characteristic of the probability

of maintaining at a given time interval $[t_s, t_f]$ the required level of software quality under unfavourable circumstances. The adversity refers to the occurrence in the operation of an intellectual system of such a sequence of events, each of which individually has no significant impact on the quality of the software, the combined effect is a significant reduction in quality.

In view of the above, the objective of improving the performance of software in the design and operation of the special DSSS is to develop an algorithm for determining such control effects as $\vec{u}(t)^* \in \{\vec{U}(t)\}$, the implementation of which, under any permissible environmental conditions $\vec{x}(t)^* \in \{\vec{X}(t)\}$, would maximize

the $K_1 = \int_{t_s}^{t_f} P_Q(\vec{x}(t), \vec{u}(t)) dt$ criterion, characteristic of

the probability of maintaining the required performance of the DSS software at the $[t_s, t_f]$ time interval.

In a formalized form, the problem of improving the quality of DSSS has the following formulation:

$$K_1 = \int_{t_s}^{t_f} P_Q(\vec{x}(t), \vec{u}(t)) dt \rightarrow \max, \quad (4)$$

under restrictions

$$\begin{aligned} F_i(\vec{x}(t), \vec{u}(t)) &\geq 0, i = 1, n_1; \\ F_i(\vec{x}(t), \vec{u}(t)) &< 0, i = 1, n_2. \end{aligned} \quad (5)$$

boundary conditions

$$\begin{aligned} F_i^{(t_s)}(\vec{x}(t), \vec{u}(t)) &\geq 0, i = n_2 + 1, n_3; \\ F_i^{(t_f)}(\vec{x}(t), \vec{u}(t)) &= 0, i = n_3 + 1, n_4, \end{aligned} \quad (6)$$

arising from the specificity of the control object specified in [13], where $n_1 - n_4$ are known constants; $\{\vec{X}(t)\}$ and $\{\vec{U}(t)\}$ – the domain of the valid values of

the vectors $\vec{u}(t)$ and $\vec{x}(t)$ respectively.

A general approach to problem (4) can be formed based on the assumption that (4) is a variable calculus class for conditional extremum. Its solution using the classical mathematical apparatus of the theory of variational calculus, based on the use of Euler's equations, involves considerable difficulties, the main ones being:

- high task dimension;
- the uncertainty of a number of scalar constituents of environmental state vectors $\vec{x}(t) \in \{\vec{X}(t)\}$ at $[t_s, t_f]$ time intervals of considerable length (due to the specificity of the control object);
- the presence of both quantitative and qualitative variables in the $\vec{x}(t) \in \{\vec{X}(t)\}$ and $\vec{u}(t) \in \{\vec{U}(t)\}$ vectors, which are also unclear;
- the need to solve in real time a complex system of high-order nonlinear differential equations for synthesis of $\vec{u}(t)$, etc.

Due to this circumstance, the developed algorithm of the vector synthesis of control effects $\vec{u}(t)$ is based on the statement, according to which, in order to achieve the goal of maximizing the $K_1 = \int_{t_s}^{t_f} P_Q(\vec{x}(t), \vec{u}(t)) dt$ criterion, it is sufficient to develop and implement a detailed comprehensive plan of operations $PL(t)$ to improve the performance of the DSS software [17].

In implementing this approach, the main challenge is to develop a formal procedure for testing the feasibility of a $PL(t)$ plan at different $\vec{x}(t) \in \{\vec{X}(t)\}$ values.

This would reduce the task (4) procedure to a periodic test of feasibility $PL(t)$, including under unfavourable circumstances.

According to the terminology used in the development of operation research models and methods, an operation is understood as a system of action that shares a common vision and aims to achieve its objective [18]. If the plan is executed at a given point in time, the quality of the DSSS under consideration has been maximized, that is to say, the task has been completed. Otherwise, it is necessary to identify the obstacles to $PL(t)$ implementation and also to develop and perform a set of measures to address them.

To implement this heuristic approach to solving problem (4), it is necessary to develop models and algorithms for checking the feasibility of the plan of operations based on the use of:

- apparatus of Boolean functions;
- productive models;

– models and methods of system dynamics.

In the event that all of the above-mentioned stages of verification at the established sites confirm the feasibility of the plan of operations, $PL(t)$ shall be considered feasible and, for each operation of the plan, the values of the optimal set of control effects $\vec{u}(t)^*$ shall be determined, which need to be realized for its implementation.

If, at one stage or another of the verification, the plan of operations $PL(t)$ turns out to be impossible, then the reasons for the impossibility of its execution are determined and the control actions are specified, the implementation of which will allow $PL(t)$ to be executed.

If under the circumstances it is not possible to ensure the feasibility of a $PL(t)$ plan, task (4) has no solution, and optimizing its efficiency criterion requires the development of another solution.

The implementation of individual elements $PL(t)$ is proposed to be represented by an event tree G_{et} , a connected acyclic graph $G(u(t), e(t))$ of whose vertices $u(t) \in \{U(t)\}$ correspond to the individual operations of plan $G(u(t), e(t))$, and an edge of graph $e(t) \in \{E(t)\}$ characterizes the hierarchical cause effect relationship given on the set of its vertices ($\{U(t)\}, \{E(t)\}$) are the set of vertices and edges of $G(u(t), e(t))$ respectively.

As events, individual operations of the $PL(t)$ plan are considered, which $\forall t \in const, t \in [t_s, t_f]$ may or may not be executed (partial execution is not taken into account). In essence, graph $G(u(t), e(t))$ is a formalized description of the plan of operations $PL(t)$.

Constructing an operation plan to improve the quality of software functioning is a rather complex and time-consuming task, the solution of which must be carried out for each DSSS, taking into account the peculiarities of its functioning, operating experience, and expert opinion. In this case, it is advisable to use known models and methods of management of programme projects to increase their efficiency and reduce the cost of implementation [19].

On the basis of the above, a modified heuristic procedure for constructing the $PL(t)$ plan based on the requirements of ISO / IEC 9126 and ISO / IEC 14598 standards is proposed. It consists of the following main steps.

- 1) A graph of the hierarchical cause effect relationships G_{cl} between the individual elements of the DSSS is constructed, for which the assigned task, the structure of which is given in (Fig. 2) [20] is performed. The main software modules of the given system

$N_i, i = \overline{1, m}$, which influence the quality of its functioning are defined.

2) For each programme module $N_i, i = \overline{1, m}$, according to [14–15], the characteristics, sub-characteristics and properties (indicators) of the quality of its performance shall be selected. As a result, a graph of hierarchical cause effect relationships between the characteristics and performance indicators of program modules $N_i, i = \overline{1, m}$ is formed. To construct this graph, it is proposed to use positions in graph theory using a rank approach where the characteristics, sub-characteristics, and properties have corresponding rank. The basic principles for the use of rank approach in graph construction are described in [21].

3) Graph G_{cl} characterizes the complex of cause effect relationships that exist between the quality indicators of $N_i, i = \overline{1, m}$ software modules and the most common types of DSSS errors that affect the quality of its operation. The most common types of software errors as well as their impact on the quality indicators of the software modules are discussed in detail in [22].

errors inherent in the DSSS software.

The event tree graph is defined by merging the graphs G_1 and $G_m: G_{cl} = G_1 \cup G_2 \cup \dots \cup G_m$, the fragment of the graph is given in Fig. 3.

Fig. 3. Fragment of an event tree graph G_{cl}

The edges of this graph are conjunctive edges.

A formalized description of the vertices of the fragment of the graph shown in Fig. 3 (events) is given in Tabl. 1.

Table 1

Formalised description of the vertices of an event tree graph

Number of vertices	Process
u_1	Compatibility option
u_7	Adequacy
u_8	Verity
u_9	Capacity for interaction
u_{10}	Security
u_{11}	System consistency as a whole
u_{20}	Consistency of modules
u_{34}	Failure to perform functions specified in the requirement specification
u_{35}	Inconsistency of programme documentation to specifications
u_{36}	deficiencies in documentation
u_{37}	Lack of software delivery package files
u_{46}	Errors in interaction with telecommunication information exchange software

Fig. 2. List of tasks to be carried out in the development of the information support system for decision-making (Set of information models)

From the point of view of the specificity of the tasks to be addressed by DSSS, it is necessary to supplement the general list of errors with specific types of

End of Tabl. 1

u_{47}	Errors in the use of data from other software tools
u_{48}	Incompatible data and file formats
u_{49}	Deficiencies in access management
u_{50}	Deficiencies in information management tools and records
u_{51}	Deficiencies in the protection of information against distortions
u_{52}	Deficiencies in the protection of information against distortions
u_{53}	Non-compliance with the requirements of the used cryptographic tools
u_{57}	Reliability errors

The constructed event tree G_{cl} is the basis for the development of models and algorithms to quantify the ability to execute a plan of $PL(t)$ operations while searching for a solution to the problem (4).

Conclusions

On the basis of an analysis of existing approaches to the task of improving the performance of the man

machine system, it has been determined that the most effective solution would be to consider it as a vector optimization problem with limitations in the form of equality and inequality. It has been established that it is necessary to develop mathematical models, numerical methods and algorithms that allow to optimize the target function characterizing the quality of the DSS software at different time intervals. On the basis of the formal objective of improving the quality of the DSS software of the systems under unfavourable circumstances, it has been established that it belongs to the class of variable calculus for conditional extremes. A heuristic approach to solving this problem is proposed and justified. It has been shown that the task of improving the quality of DSSS can be reduced to the task of synthesizing a complex plan of operations for improving the quality of software and analysing its feasibility under different control influences and environmental conditions. The proposed approach to the synthesis of a complex plan of operations for improving the quality of software in the form of an event tree is based on the positions of the apparatus of graph theory, in particular the rank approach.

References

1. Besnard D. Structure for Dependability: Computer Based Systems from Interdisciplinary Perspective / D. Besnard, C.Gacek, B. Jones Cliff // Springer Verlag London Limited, 2006. – 304 p.
2. Computer Safety, Reliability and Security. ed. by R. Winther, Biorn A. Gran, G.Dahl // Proc. 24th Intern. Conf. SAFECOMP 2005, Friedrikstadt, Norway, September 28-30, 2005. – 409 p.
3. Бахмач Е.С. Отказобезопасные информационно-управляющие системы на программируемой логике / Е.С. Бахмач, А.Д. Герасименко, В.А. Головир и др.; под ред. В.С. Харченко, В.В. Скляра. – Национальный аэрокосмический университет «ХАИ», Научно-производственное предприятие «Радий», 2008. – 380 с.
4. Laprie J.C. Dependability Handbook / J.C. Laprie // LAAS Report no 98-346. – Toulouse: Laboratory for Dependability Engineering, 1998. – 365 p.
5. Липаев В.В. Надежность и функциональная безопасность комплексов программ реального времени / В.В. Липаев. – М., 2013. – 207 с.
6. Нариньяни А. Продукционные системы. Представление знаний в человеко-машинных и робототехнических системах / А. Нариньяни, Т. Яхно. – М.: ВЦ АН СССР, ВИНТИ, 1984. – С. 136-177.
7. Zadeh L.A. Fuzzy Sets / L.A. Zadeh // Information and Control. – 1965. – Vol. 8. – P. 338-353.
8. Zadeh L.A. The Concept Of A Linguistic Variable And Its Application To Approximate Reasoning / L.A. Zadeh // Information Sciences. – 1975. – Vol. 8. – P. 199-249, 301-357; – Vol. 9. – P. 43-80.
9. Tsukamoto Y. An Approach To Fuzzy Reasoning Method / Y. Tsukamoto // Advances In Fuzzy Set Theory And Applications, 1979. – P. 137-149.
10. Miller S.P. Software model checking takes off / S.P. Miller, M.W. Whalen, D.D. Cofer // Commun. ACM. – 2010. – № 53(2). – P. 58-64.
11. Checking Expert System Knowledge Bases for consistency and completeness / T. Nguen, W. Perkins, T. Laffey, W. Pecora // Proc. of the 9th Int. Joint Conf.on AI, Los.Ang, 1985. – P. 375-378.
12. Липаев В.В. Проблемы обеспечения качества сложных систем [Электронный ресурс] / В.В. Липаев. – Режим доступа: <http://quality.eur.ru/MATERIALY4/poksps.htm> (дата обращения: 19.09.2020).
13. Метод формирования признаков информационной модели конфликтных ситуаций для подсистем поддержки принятия решений в перспективных системах управления специального назначения / М.А. Павленко, Г.С. Степанов, М.В. Касьяненко, В.Н. Руденко // Збірник наукових праць Харківського національного університету Повітряних Сил. – 2016. – № 3(48). – С. 101-103.
14. ISO/IEC 9126 Software Engineering. Product quality. – Kyiv, 2014. – 20 p.
15. ISO / IEC 14598 Information Technology. Software product evaluation – Kyiv, 1999. – 18 p.
16. Використання стандартів SQuaRE при проведенні випробувань програмного забезпечення у збройних силах України / О.В. Рижков, М.О. Геращенко, Р.Д. Барвінок, Ф.В. Єрмоленко // Сучасні інформаційні технології у сфері безпеки та оборони. – 2018. – № 1(31). – С. 89-94.

17. Долинина О.Н. Алгоритмы и методы разработки и отладки экспертных систем / О.Н. Долинина. – Саратов: Саратов. гос. техн. ун-т, 2015. – 226 с.
18. Татха Х. Введение в исследование операций: пер. с англ. / Х. Татха. – 6-е изд. – М.: Вильямс, 2001. – 912 с.
19. Фатрелл Р.Т. Управление программными проектами: достижение оптимального качества при минимуме затрат: пер. с англ. / Р.Т. Фатрелл, Д.Ф. Шафер, Л.И. Шафер. – М.: Вильямс, 2003. – 1136 с.
20. Когнітивний підхід до розробки інформаційних моделей в системах підтримки прийняття рішень / М.А. Павленко, В.К. Медведєв, П.Г. Бердник, С.В. Міхасьов // Наука і техніка Повітряних Сил Збройних Сил України. – 2016. – № 2(23). – С. 138-141.
21. Ранговый подход к решению задач линейного и нелинейного булевого программирования для планирования и управления в распределенных вычислительных системах / С.В. Листровой, Е.С. Листровая, М.С. Курцев // Электронное моделирование. – 2017. – Т. 39, № 1. – С. 19-38.
22. Майерс Г. Качество программного обеспечения / Г. Майерс. – М.: Мир, 1980. – 360 с.

Received by Editorial Board 09.09.2020

Signed for Printing 13.10.2020

Відомості про авторів:

Турінський Олександр Васильович

кандидат технічних наук
начальник Харківського національного
університету Повітряних Сил ім. І. Кожедуба,
Харків, Україна
<https://orcid.org/0000-0001-6888-6045>

Павленко Максим Анатолійович

доктор технічних наук професор
начальник кафедри Харківського національного
університету Повітряних Сил ім. І. Кожедуба,
Харків, Україна
<https://orcid.org/0000-0003-3216-1864>

Пєвцов Геннадій Володимирович

доктор технічних наук професор
заступник начальника Харківського національного
університету Повітряних Сил ім. І. Кожедуба
з наукової роботи,
Харків, Україна
<https://orcid.org/0000-0002-0426-6768>

Осієвський Сергій Валерійович

кандидат технічних наук
доцент кафедри Харківського національного
університету Повітряних Сил ім. І. Кожедуба,
Харків, Україна,
<https://orcid.org/0000-0003-0861-9417>

Information about the authors:

Oleksandr Turinskyi

Candidate of Technical Sciences
Chief of Ivan Kozhedub Kharkiv National
Air Force University,
Kharkiv, Ukraine
<https://orcid.org/0000-0001-6888-6045>

Maxim Pavlenko

Doctor of Technical Sciences Professor
Head of Department of Ivan Kozhedub Kharkiv National
Air Force University,
Kharkiv, Ukraine
<https://orcid.org/0000-0003-3216-1864>

Hennadii Pievtsov

Doctor of Technical Sciences Professor
Deputy Chief of Ivan Kozhedub Kharkiv National
Air Force University
in Science,
Kharkiv, Ukraine
<https://orcid.org/0000-0002-0426-6768>

Serhiy Osievskiy

Candidate of Technical Sciences
Senior Lecturer of Ivan Kozhedub Kharkiv National
Air Force University,
Kharkiv, Ukraine
<https://orcid.org/0000-0003-0861-9417>

МЕТОД ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ ФУНКЦІОНУВАННЯ ЛЮДИНО-МАШИНОЇ СИСТЕМИ ЗА РАХУНОК ПІДВИЩЕННЯ ЯКОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ

О.В. Турінський, М.А. Павленко, Г.В. Пєвцов, С.В. Осієвський

В статті розглянуто питання підвищення ефективності функціонування людино – машинних систем за рахунок підвищення якості програмного забезпечення систем підтримки прийняття рішень (СППР). Запропоновані рішення базуються на існуючих моделях та методах, детальний аналіз яких проведено в ході наукового дослідження, зокрема в певних отриманих рішеннях знайшла своє застосування методологія статичного відлагодження програмних додатків. В основу рішення зазначеного завдання запропоновано покласти методи рішення задачі управління ЛМС по векторному критерію. Запропоновано механізм синтезу комплексного плану операцій щодо підвищення якості програмного забезпечення СППР, показано можливість аналізу поетапного виконання плану з отриманням аналітичної оцінки. З цією метою комплексний план представлено у вигляді дерева цілей, побудованого у відповідності до положень теорії графів

та рангового підходу. Проведена оцінка часової складності алгоритмів, що реалізують рішення завдання побудови дерева цілей. На основі отриманих формалізованих рішень виявлено, що максимальна кількість ребер для графа, що реалізує дерево цілей, не перевищує n^2 , кількість переглядів ребер не перевищує n , тобто загальна кількість операцій, що реалізує перегляд ребер та додавання їх в зважене мінімальне покриття (ЗМП), не перевищує n^3 . Встановлено, що при реалізації операції додавання ребер в ЗМП, на кожному кроці, після фіксації попереднього результату кількість переглядаємих ребер зменшується, тому сумарна складність алгоритмів не перевищить n^3 . Отримані рішення відповідають вимогам ДСТУ ISO/IEC 9126, ДСТУ ISO/IEC 14598 та враховують вимоги серії стандартів Software Quality Requirements and Evaluation як значення вершин графа дерева подій. В процесі рішення завдання врахована специфіка функціонування ЛМС, зокрема можливість формалізації різних аспектів знань (алетичних, дисизіональних, каузальних, діонтичних) та забезпечення заданого рівня оперативності пошуку рішень.

Ключові слова: програмне забезпечення, система підтримки прийняття рішень, граф, дерево цілей.

МЕТОД ПОВЫШЕНИЯ ЭФФЕКТИВНОСТИ ФУНКЦИОНИРОВАНИЯ СИСТЕМЫ ЧЕЛОВЕК–МАШИНА ЗА СЧЕТ ПОВЫШЕНИЯ КАЧЕСТВА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ СИСТЕМ ПОДДЕРЖКИ ПРИНЯТИЯ РЕШЕНИЙ

А.В. Туринский, М.А. Павленко, Г. В. Певцов, С.В. Осиевский

В статье рассмотрены вопросы повышения эффективности функционирования систем человек-машина за счет повышения качества программного обеспечения систем поддержки принятия решений (СППР). Предлагаемые решения базируются на существующих моделях и методах, детальный анализ которых проведен в ходе научного исследования. В частности, в полученных решениях нашла свое применение методология статической отладки приложений. В основу решения указанной задачи предложено использовать методы решения задачи управления ЛМС с векторным критерием. Предложен механизм синтеза комплексного плана операций по повышению качества программного обеспечения СППР, показана возможность анализа поэтапного выполнения плана с получением аналитической оценки. С этой целью комплексный план представлен в виде дерева целей, построенного в соответствии с положениями теории графов и рангового подхода. Проведена оценка временной сложности алгоритмов, реализующих решение задач построения дерева целей. На основе полученных формализованных решений обнаружено, что максимальное количество ребер для графа, которое реализует дерево целей, не превышает n^2 , количество просмотров ребер не превышает n . Таким образом общее количество операций, которое реализует просмотр ребер и добавления их в взвешенное минимальное покрытие (ВМП), не превышает n^3 . Установлено, что при реализации операции сложения ребер в ВМП, на каждом шагу после фиксации предыдущего результата количество просматриваемые ребер уменьшается, поэтому суммарная сложность алгоритмов не превысит n^3 . Полученные решения отвечают требованиям ДСТУ ISO / IEC 9126, ДСТУ ISO / IEC 14598 и учитывают требования серии стандартов Software Quality Requirements and Evaluation как значение вершин графа дерева событий. В процессе решения задачи учтена специфика функционирования систем человек-машина, возможность формализации различных аспектов знаний (алетичных, дисизіональних, каузальних, діонтичних) и обеспечения заданного уровня оперативности поиска решений.

Ключевые слова: программное обеспечение, система поддержки принятия решений, граф, дерево целей.