METHODOLOGICAL FOUNDATION FOR IMPROVING THE QUALITY OF INTELLIGENT DECISION-MAKING SYSTEM SOFTWARE

On the basis of a detailed analysis, existing terminological interpretations of the concept of "software quality" have been generalized, conclusions are drawn about the correspondence of the terms used to assess the quality of general software in the process of assessing the quality of software of intelligent decision-making systems (IDMS). It has been proved that the quality of the IDMS software is a complex multi-criteria indicator that takes into account not only the performance of the individual software module as a subsystem, but also the causal relationships of the elements of the software system itself. The main differences in software quality assessment between the functional and formal approaches are shown. The structure of the criterion of guarantor capacity of decision-making systems software has been investigated and conclusions have been drawn on the influence of its main components on the evaluation of IDMS software and on the provision of reliable computing process. On the basis of the analysis of the list of attributes and the quality metric of the IDMS software, it is established that the guarantee is determined by the reliability of the software structure itself and is characterized by the restoration of the functional state after failures or failures. The interrelationship and influence of IDMS software design quality indicators on the characteristics and sub-characteristics of the IDMS software is established, an example of the interrelationship between characteristics (factors) and quality indicators, the method of measuring quality indicators and design processes is given. On the basis of the conducted research, IDMS software denial regimes have been defined and their impact on the decision-making process has been shown. Detailed classes of failures and their influence on compliance of IDMS software with the task of development are shown. It has been shown that the reliability of IDMS is a dynamic concept, manifested in time, and is strongly dependent on the presence / absence of defects in the interaction. A detailed analysis of methods of software quality assurance and control has been carried out, and conclusions have been drawn on the possibility of their application IDMS software. The maturity model of the IDMS software has been improved and validated, and the maturity structure of the software as an indicator of the quality of the IDMS has been introduced.

Keywords: automated control systems, decision support, intelligent system software, knowledge base, intelligent decision-making system.

Introduction

Problem statement. By the quality of software, including intelligent systems, it is understood: the ability of a software product, under given conditions, to satisfy established or anticipated needs [1]; the entire scope of features and characteristics of programs, which refers to their ability to meet the stated or anticipated needs [2], the degree to which a system, component or process meets the needs and expectations of the user [3]. In summarizing the definitions given, the quality of the software of the intelligent systems will be understood to mean the set of properties that characterize the ability of the software to maintain the necessary (defined terms of reference). The level of quality of operation under the set conditions within a given period of time.

It is clear that the quality of the ISPR software as a whole is a complex multi-criteria indicator that takes into account not only the performance of the individual software module as a subsystem, but also the causal relationships of the elements of the system itself. Namely, under considering the knowledge base as a central part of the intellectual decision-making system, it is necessary to consider not only the influence of input data and its quality, but also the influence of the associated subsystems. However, methods to ensure the quality of the knowledge base are still not formalized and there is no single approach to solving this problem. On the other hand, the process of intellectualizing management systems, decision support systems, is accompanied by a significant increase in the knowledge base and requires the establishment of effective quality control methods for this essential component of the intellectual decision-making system.

Analysis of recent research and publications.
An analysis of existing solutions and approaches to assessing software quality has shown that there are quite a number of solutions that in some respects are quite opposite (in terms of the estimation itself). For example, the formal approach regards the quality of a programme product as the extent to which the product meets the
stated requirements and needs of stakeholders, that is, it focuses more on the structural characteristics of the software. A functional approach [4–6], based on the assertion that the functionality and level of satisfaction of the users of a software product is more important than its structural characteristics, which are commonly used in assessing software quality. In considering these approaches, it should be noted that the concept of software quality is rather subjective in nature, as it depends on people who assess quality [7]. The development of quality theory for modern information systems using the intellectual, network and cloud computing technologies needed to solve large-scale management problems is not sufficiently effective. The main reason is that models and methods of classical quality theory cannot describe and provide adequate estimates of objects whose performance is impaired not only by failures related to physical nature, but also by design errors, Information impacts and errors in knowledge bases. For such objects, it is difficult to define the very concept of waiver and to identify the full range of causes that give rise to it. On the other hand, the development of intelligent decision-making systems today is often carried out by specialists of informal engineering philosophy (the so-called new formation with the use of hybrid methodologies), whose view of the issues of software development and analysis and the results obtained is somewhat "artistic" in nature, but not so, containing scientific rationale in which an orientation towards qualitative rather than quantitative assessment tools that do not fit into the mathematical apparatus of harsh formal methods is preserved [8].

**Purpose and objectives of the study.** Based on the analysis and research results of existing solutions, an urgent task arises, which consists in the development of methodological provisions for assessing the quality of software of information decision-making systems in which models and methods of the theory of software quality would provide an adequate assessment of objects (software modules, classes, functions), the performance of which is disrupted due to design errors, information influences and errors in knowledge bases.

For solving this problem it is necessary:
- to investigate the correlation of software quality indicators, validate ISO / IEC 25010: 2011 and indicators of software development process quality;
- to evaluate the decision on whether to apply to the task to examine the structure of the guarantee function of software decision-making systems and draw conclusions.

**Statement of basic materials**

Quite often, the term “dependability” is used to denote the quality of software systems, which in fact can be interpreted as a “guarantor” [8–11]. The existing developments of domestic scientists point to the fundamental underpinnings of this definition and ensure the unquestionable realization of the following characteristics:
- availability – a feature of the availability of software resources for the provision of necessary services;
- reliability – the ability to provide the necessary services continuously;
- high confidence – the property of correctly assessing the correctness of the services provided, that is to determine the degree of trust in the service;
- maintainability – property of adaptation to modifications;
- confidentiality – ability to prevent unauthorized access to service information;
- functionality – ability to perform a function or multiple functions;
- safety – ability to comply with safety policy.

The basic principle of guarantor-capable dependable computing, formulated in the work [12], defines reliable computing (obtained from the results of guarantor-capable software systems) as resistant to faulty hardware and software, that is to say, failures caused by defects made during development and not detected during debugging. Another feature of guarantor-capable software systems is the secure fault tolerance; this concept has been introduced in [13] and is defined as a means of supporting the concomitant properties of the guarantee capacity. This approach to the guarantee is justified by the fact that the means studied can provide (improve) the reliability, readiness, security and durability of the software system. Thus, guarantor-capable software systems formally realize the property of producing results that can be trusted. The comprehensive guarantee criterion consists of three main components: threats, attributes, and means. The structure of the guarantee criterion is shown in Fig. 1.

![Fig. 1. Structure of the guarantee criterion for decision making software systems](Source: developed by the authors.)
ISO / IEC 25010: 2011 proposed to use a multi-level model to describe software quality. The top level defines eight basic quality characteristics. Each characteristic is described by the corresponding attributes, which in turn are defined by sets of metrics. The structure of software quality according to the requirements of ISO / IEC 25010: 2011 is shown in Fig. 2.

![Software quality structure](image)

**Fig. 2. Software quality structure according to ISO / IEC 25010: 2011**

An analysis of the attribute list and the quality metric of the IDMS software showed that the reliability of the program structure determines the guarantor’s reliability. In turn, the reliability of the IDMS characterises the recovery of the operational condition of the system software after failures or failures. Resistance to rejection depends on the level of defects and errors that have not been corrected and the ability of IDMS to respond to them in a manner that does not affect reliability [14].

![Characteristics and sub-characteristics of the IDMS software](image)

**Fig. 3. Characteristics and sub-characteristics of the IDMS software**

In existing regulations, fault tolerance is defined as the property of a system to continue to perform correctly with a limited number of hardware or software defects, as well as the completeness and duration of the restoration of IDMS during the restart process. At the same time, the set of categorical - descriptive, quantitative and qualitative characteristics of the software should correspond to the set value in the task of development.

The quality of the IDMS software can only be assured if the quality indicators of the software development processes are consistent (Fig. 4).

![Indicators of the quality of software development processes](image)

**Fig. 4. Indicators of the quality of software development processes**

Let us briefly characterize indicators of the quality of software design processes and show their impact on the sustainability of software characteristics.

**Efficiency.** A property that shows how effectively a supplier can develop new functionality. Efficiency can be measured over time with a limited development resource.

**Timeliness.** An indicator that determines whether a given functionality can be realized within the required time frame. A design process that does not achieve a timely result even with a substantial increase in resources is considered unacceptable.
Transparency. The transparency of development processes requires that there be clear stages and operations that can be analysed by stakeholders. It is important for the customer to see how the IT system is developed: this will allow to understand current processes and control possible risks.

There is a direct relationship between the values of the characteristics of the IDMS software and the indicators of the quality of the software development processes.

An example showing the relationship between characteristics (reliability and maturity), quality indicators, the way of measuring quality indicators and design processes is given in Fig. 5.

The IDMS failure will be understood to mean that an event occurs when the result of the system work is different from what is supposed to be. Threats are classified as failure, errors, and improper work. The system may malfunction if it is out of specification or the specification does not adequately match the system's functions. Failure is active when it causes error, otherwise it is considered passive.

The methods of failure are called modes. In turn, failure regimes can be classified according to their severity: type of failure, handling of failures, consistency of failures where the system has two or more users, consequences of failures and their influence on the decision-making process of the decision-maker Fig. 6.

Fig. 5. Relationship between characteristics (factors) (Reliability and Maturity), quality indicators, measurement of quality indicators and design processes
Source: developed by the authors.

Fig. 6. Failure modes of IDMS
Source: developed by the authors.
The challenge of achieving the required quality is addressed through the use of modern process management techniques and the use of IDMS life cycle tools [15], which are supported by international and domestic standards.

Consider that a quality system in terms of reliability has in general a set of workable (safe) states $MS$, partially workable (safe) states $MR^{ci}$, $i = 1, n$ ($n$ – the number of allowable deterioration levels) and fully non-functional states $MN$ if $n > 1$, speak of the characteristics of the survivability of the system [3]. The $MN$ set consists of many disabled but safe states $MB$ and dangerous states $MD$, where $MD$ is represented as a set of functionally dangerous (emergency) states, and the sets $MR^{ci}$ and $MS$ combine functionally safe states.

For various types of failures, the system goes from a healthy initial state $S_0$ to a subset of states $S_1, S_2, S_3$, where $S_1 = S_{pc} \cup S_{hk}$ is a set of development or design defects, $S_{pc}$ is a set of software design defects, $S_{hk}$ is a set of design defects in the knowledge base of an intelligent decision-making system; $S_2$ – sets of physical defects of the system, characteristic of the hardware, often arise due to the physical aging of the elements; $S_3 = S_{si} \cup S_{hi}$ is a set of interoperability defects, where $S_{si}$ is a set of software interoperability defects, $S_{hi}$ is a set of hardware interoperability defects.

Communication defects can result from external influences, such as information attacks or other unauthorized interventions, service personnel errors, physical extremes, and are the primary cause of IDMS failures. When a defect is detected and fixed, the system reverts to $S_0$, unless additional (backup) hardware and software resources are used. If in real conditions, the output data falls into the range of values that cause failures that were not taken into account during the development of the system or defined by the specification or other regulatory documents, and at the same time there is a quick recovery of the system, such that the failure is not recorded, then these events do not affect such indicators reliability of IDMS as mean time between failures and availability factor. Therefore, it can be said that the reliability of IDMS is a dynamic concept, manifested in time [3; 16]. The quality IDMS condition model is shown in Fig. 7 for reliability.

The basis of the quality assurance system for IDMS software is quality assurance and quality control methods. Quality assurance methods are technologies, which during the application guarantee the achievement of relevant software quality indicators. Quality control techniques are needed to confirm that certain characteristics of software quality have been achieved. They help to determine whether they have produced the intended result and identify defects, errors and deviations.

The following classification of quality control methods is used in the literature:

- methods, approaches and technologies related to the definition of quality indicators based on software modelling using different models (model testing, prototyping);
- methods, approaches and technologies to detect violations of source code, documentation and design models (program code inspection, code error search methods based on interpretation, documentation checking for consistency and compliance with standards);
- methods, approaches and technologies associated with formalized or conventional analysis of project documents and source code to identify their properties (software architecture analysis methods);
- methods, approaches and technologies for determining the properties of software during its operation, e.g. dynamic control methods. These include testing, profiling and measuring those quantitative quality measures that are determined by software performance, such as performance in reliability, decision time, availability and other resources.

The following types of testing can be distinguished for IDMS as a software system:

- modular testing is used to verify the correct functioning of individual modules or subsystems of the IDMS. In this case, it is checked that if the subsystem receives input data that meets the set correctness criteria, then the results of its work will be correct. The criteria for the correctness of the input and output data are usually set by program documents describing the prerequisites for each operation on which the input data must work, postconditions describing the ratio of the input data for each case under consideration and returned by the results module, criteria for the integrity of the internal data of the modules;
- integration testing is used to verify that the interaction of a given group of system modules or subsystems is correct. At the same time, it is also verified that,
in the course of their joint work, subsystems exchange data and calls from operations, without violating mutual restrictions on such interaction, such as the prerequisites for calling operations. Integration testing is applied at a later stage of development;

- system testing is used to control the correctness of the operation of the decision-making system as a whole, to analyse the extent to which the whole spectrum of tasks assigned by users in different conditions and on different input data is correctly solved. System testing also involves testing the user interface (or with the help of the user interface), including the Web application interface (WebUI) for IDMS Internet applications. While integration and modularization testing is usually conducted using a software application interface (Application Programming Interface, API), most system-level applications use the user interface, although API testing can also be implemented;

- regression testing is used for various levels of testing (modular, integration, system) to verify that errors are corrected.

In many technically complex industries where greater reliability is required, industry standards already incorporate the basic quality requirements of IDMS. Examples include the European Commission’s sectoral standards 50126, 50128, 50129 CENELEC on electrical engineering, supplemented by RAMS (Reliability, Availability, Maintainability, Safety) - requirements for software decision-making systems that include such criteria, as reliability, accessibility, support, safety [16–18]. The standards supplement regulates management processes for the development of decision-making software systems, which enhances the reliability of information systems, but does not ensure that management decisions are error-free [19].

CMMI SE / SW (Capability Maturity Model for Systems and Software Engineering) is a maturity process model Fig. 8, consists of interrelated processes reflecting different levels of maturity, and is shown in Fig. 9.

![Software maturity process model](image)

**Fig. 8. Software maturity process model**

Source: developed by the authors.

It should be noted that by placing the knowledge base in a special class of software, in general, the problem of quality assurance of ISDR can be divided into 3 component parts:

- hardware quality assurance;
- software quality assurance;
- quality assurance of knowledge bases.

The fact that the quality of the knowledge base, on the one hand, directly influences the decision-making of the decision-maker, makes the knowledge base a special
part of the IDMS software a separate group. It follows that software quality assurance methods applied to knowledge bases must have their specificities, models and algorithms.

Methods for ensuring the quality of IDMS hardware are well described in the literature, methods for ensuring the quality of classical software are described at the level of regulatory documents, however, quality indicators are quite generalized, and when detailing them, developers have to introduce their own specifying characteristics, which are often unique for each complex decision-making system, and many quality indicators of IDMS depend on the functional purpose and properties of each system.

Conclusions

The current state of play in the development of methods, models and algorithms for software quality in intelligent decision-making systems can be described as follows:

1) As a result of the study, a contradiction has been revealed that the number of intelligent decision-making systems in critical areas is steadily growing due to the use of expert knowledge in poorly formalized areas, the complexity of systems increases, the volume of program code and the size of knowledge bases increases, and the number of errors made during development increases. While the existing methods of quality assurance of IDMS are developed mainly at the empirical level and cannot ensure the effective development of this class of systems. Intellectual decision-making systems are increasingly being used in critical areas where mistakes in decision-making can lead to loss of life and great material damage.

2) The quality of the IDMS software is a complex criterion, combining quantitative and qualitative metrics for which no formal methods of achieving it are available in the literature.

3) The knowledge base is the centrepiece of the intellectual decision-making system; currently the most common is the production knowledge base and artificial neural networks based on three-layer perceptron.

4) An important step in the IDMS life cycle for quality is the development of knowledge bases and two major groups of methods:
   - static analysis methods, testing the knowledge base at the level of formal quality control, and do not require the system interpreter to be started;
   - testing, which consists of running the IDMS on a given set of test data and comparing the IDMS output results with reference defined by experts.

References

Відомості про авторів:

Павленко Максим Анатолійович
dоктор технічних наук професор
начальник кафедри Харківського національного
університету Повітряних Сил ім. І. Кожедуба,
Харків, Україна
https://orcid.org/0000-0003-3216-1864

Осієвський Сергій Валерійович
cандидат технічних наук
dоцент кафедри Харківського національного
університету Повітряних Сил ім. І. Кожедуба,
Харків, Україна
https://orcid.org/0000-0003-0861-9417

Данюк Юрій Володимирович
cандидат технічних наук
cандидат науковця-науково-виховного центру мової
підготовки Харківського національного
університету Повітряних Сил ім. І. Кожедуба,
Харків, Україна
https://orcid.org/0000-0002-3846-1078

Information about the authors:

Maksym 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

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

Yurii Daniuk
Candidate of Technical Sciences
Head of the Research and Training Linguistic Center
of Ivan Kozhedub Kharkiv National
Air Force University,
Kharkiv, Ukraine
https://orcid.org/0000-0002-3846-1078

METODOLOGІЧНІ ОСНОВИ ПІДВИЩЕННЯ ЯКОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ІНТЕЛЕКТУАЛЬНИХ СИСТЕМ ПРИЙНЯТТЯ РІШЕННЯ

М.А. Павленко, С.В. Осієвський, Ю.В. Данюк

На основі детального аналізу проведено узагальнення існуючих термінологічних трактувань поняття "якість програмного забезпечення", зроблені висновки щодо відповідності термінів, що застосовуються для визначення якості залежного програмного забезпечення до процесу вивчення якості програмного забезпечення інтелектуальних систем прийняття рішень (ІСПР). Доведено, що якість програмного забезпечення ІСПР є комплексним багатокритеріальним показником, що враховує не тільки якість роботи окремого програмного модуля як відсімств, а й причинно-наслідкові зв'язки елементів самої програмної системи. Показано основні розбіжності в оцінюванні якості програмного забезпечення (П) при застосуванні функціонального та формального підходів. Досліджено структуру критерію гарантованості програмних систем прийняття рішення та зосереджено висновки щодо впливу його основних компонентів на оцінку програмного забезпечення ІСПР при прийнятті рішень інтелектуальних систем. На основі аналізу переліку атрибутів і метрик якості ПІ ІСПР встановлено, що гарантованість визначається нейтралітетом самої програмної структури та характеризується відношенням впливу підходу до вибору певних критеріїв для оцінки якості програмного забезпечення ІСПР за характеристиками та субхарacterистиками програмного забезпечення ІСПР, наведено приклад взаємозв'язку між характеристиками (факторами), показниками якості, способом вимірювання показників якості і процесами проектування. На основі проведених досліджень визначено режими впливу в програмному забезпечені ІСПР та показано їх вплив на процес прийняття рішення. Деталізовані класи впливів та показано їх вплив на відповідність програмного забезпечення ІСПР завданню на розробку.
Доведено, що надійність ІСПР є поняттям динамічним, що виснажиться в часі та суттєво залежить від наявності/відсутності дефектів взаємодії. Проведено детальний аналіз методів забезпечення та контролю якості ПЗ, зроблено висновки щодо можливості їх застосування до програмного забезпечення ІСПР. Вдосконалено та обґрунтовано модель зрілості програмного забезпечення ІСПР та наведено структуру забезпечення рівня зрілості ПЗ як показника якості ІСПР.

**Ключеві слова:** автоматизовані системи управління, підтримка прийняття рішень, програмне забезпечення інтелектуальних систем, баґа знань, інтелектуальна система прийняття рішень.

**МЕТОДОЛОГИЧЕСКИЕ ОСНОВЫ ПОВЫШЕНИЯ КАЧЕСТВА ПРОГРАМНОГО ОБЕСПЕЧЕНИЯ ИНТЕЛЕКТУАЛЬНЫХ СИСТЕМ ПРИНЯТИЯ РЕШЕНИЯ**

М.А. Павленко, С.В. Осиповский, Ю.В. Даник

На основе детального анализа проведено обобщение существующих терминологических трактовок понятия "качество программного обеспечения", сделаны выводы о соответствии терминов, применяемых для оценки качества общего программного обеспечения в процесс оценки качества программного обеспечения интеллектуальных систем принятия решений (ИСПР). Доказано, что качество программного обеспечения ИСПР является комплексным многокритериальным показателем, учитывающим не только качество работы отдельного программного модуля как подсистемы, но и причинно-следственные связи элементов самой программной системы. Показаны основные разногласия в оценке качества программного обеспечения (ПО) при применении функционального и формального подходов. Исследована структура критериев гарантированности программных систем принятия решения и сформированы выводы о влиянии его основных компонентов на оценку программного обеспечения ИСПР и обеспечения процесса надежных вычислений. На основе анализа перечня атрибутов и метрик качества ПО ИСПР установлено, что гарантированность определяется надежностью самой программной структуры и характеризуется восстановимостью работоспособного состояния после отказов или сбоев. Установлена взаимосвязь и определены факторы влияния показателей качества проектирования программного обеспечения ИСПР на характеристики и субхарактеристики программного обеспечения ИСПР, приведен пример взаимосвязи между характеристиками (факторами), показателями качества, способом измерения показателей качества и процессами проектирования. На основе проведенных исследований определены режимы отказов в программном обеспечении ИСПР и показано их влияние на процесс принятия решения. Детализированы классы отказов и показано их влияние на соответствие программного обеспечения ИСПР заданию на разработку. Доказано, что надежность ИСПР является понятием динамическим, проявляющимся во времени и существенно завися от наличия/отсутствия дефектов взаимодействия. Проведен детальный анализ методов обеспечения и контроля качества ПО, сделаны выводы о возможности их применения программного обеспечения ИСПР. Усовершенствована и обоснована модель зрелости программного обеспечения ИСПР и приведена структура обеспечения уровня зрелости ПО как показателя качества ИСПР.

**Ключевые слова:** автоматизированные системы управления, поддержка принятия решений, программное обеспечение интеллектуальных систем, база знаний, интеллектуальная система принятия решений.