Power Planet Control and Monitoring using state space diagram and Multi-Agent Environment

MAHMOUD MOHAMMAD AL-SUOD[1], A. USHKARENKO[2], O. DOROGAN[3].
[1]Department of Electrical Power Engineering and Mechatronics, [2,3]Department of Electrical and Electronic Engineering
[1]TafilaTechnical University, [2,3]Admiral Makarov National University of Shipbuilding
Tafila -11660, Nikolaev
Jordan, Ukraine
m.alsoud@ttu.edu.jo, maestrotees@gmail.com, dorogan.olga.n@gmail.com

Abstract: - The characteristics of the quality of the specialized software for a ship power system control and monitoring are discussed. The methodology of the evaluation of functionality, usability and reliability of specialized software for monitoring and managing ship's power plants are presented. The availability of such a methodology makes it possible to obtain a composition of software quality attributes and quantitative indicators, indicate the direction of priority changes and assess the results of their implementation, efficiency and reliability of specialized software.

Key-Words: - functional testing, integration testing, software quality, ship power system.

1 Introduction

Modern control systems of marine electric power systems are distributed and conditionally divided into three hierarchical levels. Hardware and software of the upper level coordinates the operation of the entire system and contains the data collection server and the operator's automated workstation. Hardware and software of the lower level is a set of territorially distributed automation hardware. Communication infrastructure is an intermediate layer element.

Requirements to the software (SW) for ship power system (SPS) control and monitoring, which is considered in this study, can be divided into three groups according to the agents involved in separate phases of the program and conventionally named as the System Configuration Operator (performs the creation of a mnemonic diagram, the adjustment and correction of the properties of its components), the Power Plant Operator (performs real-time power system monitoring and control) and the Analyst (performs the analysis of the SW and SPS operating modes for their optimization, after which the entire process of working with the software – configuration of the properties of components, real-time operation and its analysis – is repeated anew).

The preparatory stage before real-time monitoring and control of the power plant is the creation of a mnemonic diagram of the power plant and adjustment of the properties of its components. To perform the above operations, the System Configuration Operator must be able to interact with the component library.

During the operation of the software in the SPS monitoring and control mode, in addition to performing the basic operations, the information on the program operation (data exchange channel load, the contents of data bursts) and the power units (the condition and power given to the total load) should be collected over a certain period of time. Further, the stored information is processed by the Analyst with the purpose of the following:

- verification of the previously calculated response time of the system and the decision on the possibility of using the protections of diesel generator units (DGU) and the control of discrete signals at the program level; for this purpose, the information about the load of the data channel is used;
- troubleshooting during data exchange with automation tools (information about the contents of data bursts is used);
- calculation of individual power quality indicators (average, minimum and maximum values and frequency of power dips and surges in the

power system), based on information on the DGU operating modes;

- identification of such a combination of DGU nominal power, at which the amount of fuel consumed by them will be minimal, based on an analysis of the operating modes of generator units; these data can be used at rearrangement of the power plant.

Also, the Analyst's tasks include SW functional testing, so the availability of means to automate this process is desirable.

Since the use of the software is a three-step iterative process (alteration or creation of a mnemonic diagram of a power plant and configuration of the properties of its components, the use of software in its major mode, system performance analysis) and the first and third stages are visually identical from the user's point of view, the SW use can be divided into two modes: the diagram developer mode and the power plant monitoring and control mode. In order to ensure reliable operation of the whole SPS, it is necessary for the software used to control and monitor the SPS parameters to meet the necessary quality standards. This is achieved by testing the software at the design step and its verification for compliance with quality standards.

The quality is one of the key issues in the top level software design for the automated control system of the marine electric generating plant. The software quality model SQuaRE is recognized to be generally accepted, it is based on series of ISO/IEC standards (25000-25099) [1]. According to this model the quality evaluation is three-level and consists in determination of required features for each software type, attributes for each feature and metrics for each attribute. Standards also specify main features for all software types: functionality, usability, efficiency, reliability, maintainability and mobility.

Several software functionality attributes can be distinguished such as suitability and precision of functions performance, interoperability with third party software and hardware automation [2]. The paper [3] provides software functionality evaluation methods and justifies testing utilization for this purpose.

The requirements to be met by the software are the basis for the execution of its functionality testing. Since software is an integral part of the hardware-software system the availability of models that simulate behavior of hardware and automation object as an entity is required for its testing. The functionality testing is usually divided into several levels: unit, integration and system [4]. The test procedure consists of the execution of a sequence of actions [5]: the definition of testing completeness criterion; the development of a complete set of test cases; the execution of a report with the information on testing results. The results of the testing execution should be metric values of functionality attributes – suitability, precision and interoperability.

As shown in [6, 7], the transition graphs of digital automata are successfully used to describe ship automation systems. In this paper, transition diagrams are used to describe the behavior of software' components. Thus, the completeness of automata transition coverage is determined to be the criteria of unit testing completeness for all components; the complete set of test cases is being written using automata transition conditions. Since this test phase refers to debugging, the transition to functionality adding and performance of the next test phase only occurs if 100% of test cases are covered.

Evaluation software is practicality in terms of values of its three attributes: ergonomics, clarity and efficiency of development. For each of the attributes which are allocated sub characteristics – indicators t are measured numerically. In [8, 9], a methods of evaluating usability metrics was among the most widely used identified, peer review and survey tests. The increasing role of metrics in software quality assessment was noted in [10, 11] and their measurement shall be performed by competent testers. The software for ship electric power systems has its own specifics, so the development of a SW testing methodology for SPS and its verification for the compliance with quality standards is a topical task.

2 Problem Formulation

At present, the problem of creating a methodology for testing and evaluating the main quality indicators of specialized software for monitoring and managing ship's power plants in accordance with international standards is topical. The availability of such a methodology will make it possible to obtain a composition of software quality attributes and quantitative indicators, indicate the direction of priority changes and assess the results of their implementation, efficiency and reliability of specialized software.

3 Problem Solution

In the power plant monitoring and control mode, the

software shall perform several tasks (the implementation of each of them is further considered separately):
- information exchange with automation means - transmission of operator's commands; reading, displaying and processing of hardware-measured analog and discrete values;
- accumulation of statistical data on data channel load – the number of transmitted and received bytes per time unit;
- accumulation of information on the operation modes of diesel-generator units: of their average values for total load for a certain period of time.

A transaction consisting of a request and a response - recording and reading of data bursts from the COM port - is the unit of data exchange. Data exchange with automation tools shall be cyclic occurring at certain intervals. Therefore, when entering the power plant control mode, a timer is activated, the triggering of which initiates the data exchange cycle as an atomic block of operations. Input actions are indicated as $e1$ - transition of the software into the power plant control mode; $e2$ - transition of the software into the diagram designer mode; $e3$ - data burst is received; $e4$ - moving the cursor in the component area; $e5$ - changing the power plant structure; $e6$ - timer expiration; $e7$ - the load of at least one DGU exceeds the limits; $e8$ - opening the synchronization window; $e9$ – set points are obtained; $e10$ - data on the state of the generator switch are obtained; $e11$ - the values of stresses, frequencies and shear angle between stress vectors are obtained; $e12$ - changing the set points; $e13$ - changing the parameters of control actions; $e14$ - data burst is received; $e15$ - COM-port opening error; $e16$ - closing the synchronization window; $e17$ - synchronization process startup; $e18$ - data are received by the component; $e19$ - clicking the left mouse button.
Input variables are denoted as $x1$ - data burst error; $x2$ - the error is eliminated; $x3$ - the error is incorrigible without replacing hardware; x4 - there is a connection: $x4.1$ - with indication components (at least one), $x4.2$ - with DGU Protection component; $x5$ - data are developed; $x6$ - the request queue is not empty; $x7$ - data have been sent to the component; $x8$ - timer is operating; $x9$ - the data received by the Load Distribution System are used for DGU protection; $x10$ - the independently measured data are used for DGU protection; $x11$ - the direct condition is satisfied; $x12$ - the reverse condition is satisfied; $x13$ - DGU is running in parallel with other DGUs; $x14$ - the generator switch is closed; $x15$ - the queue is empty; $x16$ – the record to the database is allowed; $x17$ – the display of the data burst contents is allowed; $x18$ - a communication error with the database; $x19$ - a communication error with the database has been eliminated; $x20$ - the system of correction of the engine speed and/or excitation of the generator is activated.

Output actions occurring at certain combinations of input actions and variables are designated as $z1$ - add queuing bursts whose contents are associated with values polling or with the command: $z1.1$ - the Generator parameters (active values of voltage and current; frequency and power factor values); $z1.2$ - parameters required to provide DGU protection functions; $z1.3$ - command to close the binary output; $z1.4$ - command to open the binary output; $z1.5$ - synchronization settings and control action parameters; $z1.6$ - data on the generator circuit breaker status; $z1.7$ - current values of frequencies, voltages and shear angle between the stress vectors of the main switchboard bus (MSB) and the started DGU; $z1.8$ – a command to change the synchronization settings; $z1.9$ – a command to change the control action parameters; $z1.10$ - data on the status of the generator circuit breakers of the section; $z1.11$ - data for calculating the online capacities given for DGU parallel operation; $z1.12$ – a command to activate corrective actions for frequency and voltage regulators; $z1.13$ – a command to deactivate corrective actions for frequency and voltage regulators; $z1.14$ – a command to close the circuit breaker or relay; $z1.15$ – a command to open the circuit breaker or relay; $z2$ - delete the request from the queue; $z3$ - transmit data to the component; $z4$ - timer startup; $z5$ - data display; $z6$ - timer stop; $z7$ - create a data exchange queue; $z8$ - change the Bus color; $z9$ - transmit the request packet to the hardware automation tools; $z10$ - transmit data to the state that initiated the data exchange; $z11$ - remove the request from the top of the stack; $z12$ - generate a permanent queue of requests; $z13$ - open the COM port; $z14$ - close the COM port; $z15$ - delete a request; $z16$ - to store the data about DGU load and load conditions; $z17$ - display the contents of data bursts,
then the described behavior of the Operational Field class in the power plant control mode can be described by the automaton $A_8 = \{Q, X, E, Z, f, g\}$, where
– $Q = \{q0 - q8\}$ is the set of automaton states: $q0$ is the initial state; $q1$ is waiting for the start of the data exchange cycle; $q2$ is waiting for response to the request; $q3$ is waiting for the record in the database; $q4$ is recording to the database; $q5$ is a communication error with the database; $q6$ is

waiting for the DGU and the MSB buses synchronization; $q7$ is synchronization of the DGU and the MSB buses; $q8$ is waiting for deactivation of the control actions for frequency and/or voltage regulators;

– X, E, Z are sets of accepted input actions, input variables and output actions, decoded above;

– f and g are functions of states and output symbols, represented as a graph in Fig. 1.

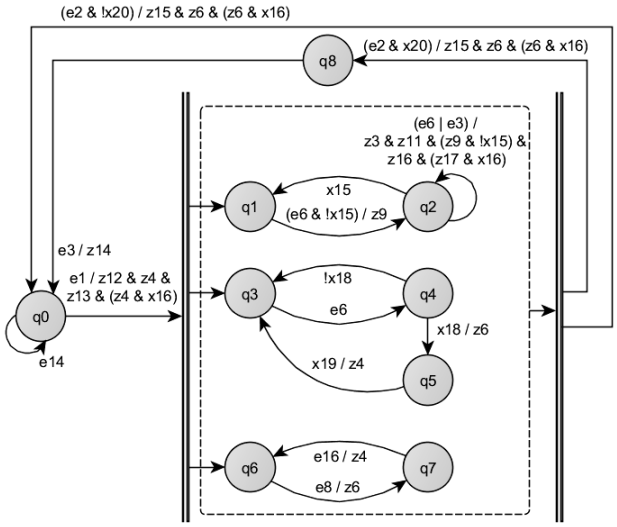
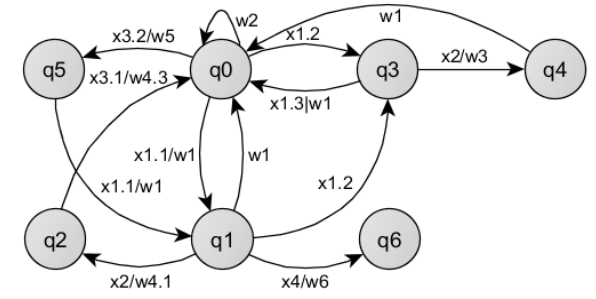


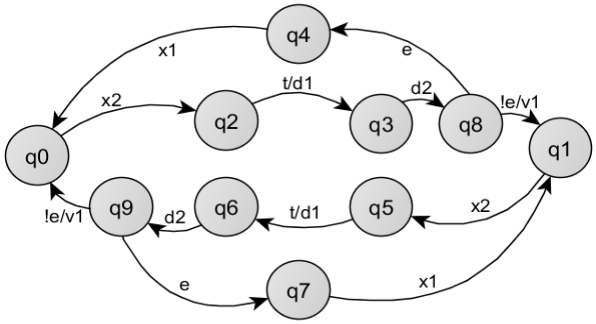
Fig.1. State chart of the "Operating Field" class in the power plant monitoring and control mode.

The task of the unit testing is metrics definition of functionality attributes of separate software components. The behavior in the design mode differs significantly from the behavior in the electric plant operation mode for most software components and is implemented (designed and coded) as two individual finite-state automata. The use of two automata enables to separate calculation of suitability metrics (completeness of test coverage for automaton that is simulating behavior in the diagram design mode) and interoperability metrics (for behavior in the electric plant operation mode).

As an example the automatic circuit breaker component testing is examined. Fig.2 shows automata that describe component behavior in the diagram design mode (Fig.2, a) and in the electric plant operation mode (Fig.2, b).



a



b

Fig.2. Behavior of the automatic circuit breaker component.

Set of states of the automata shown in Fig.2, a, is

$$Q = \{q0, q1, q2, q3, q4, q5, q6\},$$

$q0$ – object is created (the initial state); $q1$ – object is in focus (highlighted by markers); $q2$ – the subject is moving (condition is complex); $q3$ – communication with bus available; $q4$ – start communication with bus; $q5$ – end of the communication with bus; $q6$ – the object is removed (final state).

Set of states of the automata shown in Fig.2, b, is $Q = \{q0 - q9\}$, where $q0$ – the circuit breaker in the initial state; $q1$ – the circuit breaker in a state opposite to the initial, $q2$ – waiting for the expiry of the circuit breaker trip time; $q3$ – waiting for a response from the working area; $q4$ – waiting for the user's actions on the elimination of errors; $q5$ – waiting for the expiry of the return trip the circuit breaker time; $q6$ – waiting for a response from the working area; $q7$ – waiting for the user's actions on the elimination of errors; q8 – verification of the data packet with the odd change of state of the circuit breaker; $q9$ – verification of the data packet for even change the status of circuit breakers.

The same automata are used both for testing and for describing and coding of component behavior; the state transition verification is done by built-in compiler facilities for programs debugging. The full coverage of the automaton shown in Fig.2, a, comes with the following message sequence generation

$x1.1 \to x2 \to x3.1 \to x1.2 \to x2 \to w1 \to w2 \to x3.2$
$\to x1.1 \to \to x1.2 \to x1.3 \to x1.1 \to w1 \to x1.2 \to x2$
$\to w1 \to x1.1 \to x4;$

For automata shown in Fig.1, b,

$x2 \to t \to d2 \to e \to x1 \to x2 \to t \to d2 \to !e \to$
$\to x2 \to t \to d2 \to e \to x1 \to x2 \to t \to d2 \to !e.$

It should be noted that generation of individual messages (data acquisition *d*2, error *e* or no error !*e*) is possible only when using the automation controller or its model.

The next level of testing is integration testing, the purpose of which is to verify the interactions between components. The use of the apparatus of pattern networks in which automata that implement the behavior of components are represented in the form of generators, and each of the types of information exchange between components - in the form of a relationship, allows us to formalize the description of the relationships between components.

The final testing level is the system one. The power plant operation is a specific sequence of diesel generator units' and actuating units' operation according to the technological process. The aim of the system testing is a software health check when controlling electric power plant, that's why the software requirements coverage can be distinguished as a criteria of testing completeness.

Prior to compiling a set of test situations, the classes of nonequivalent situations shall be determined. In this paper, the following set of nonequivalent situations is proposed to be used: the use of a component of each type in the diagram designer mode; start-up and shutdown of the diesel-generator unit (DGU); load connection and disconnection from the buses of the main switchboard (MSB); display of DGU parameters; actuating each of the DGU protection functions; synchronization of the DGU to the MSB buses; load sharing between DGU running in parallel. In order to perform full software testing, a set of situations shall include all listed nonequivalent situations.

The software for the monitoring and control of the electric power plant is a discrete-event system, so finite-state automata could be used as means of the test case's development [3]. Although, the software is also a component of hardware-software system, so the R-diagram test sequence should be specified to enable the definition of functions that are subject to examination, as well as software models and tools used for this process [8].

To automate and enhance the testing process, the R-scheme in the form of a control script is recommended, which supports the principle of special moments as a simulation scheme [8]. For software, the script can be presented as an additional dialog box; running the script will initiate the execution of the actions recorded in it. To interpret the test results, all information notifications and control actions are recorded in the log file. After the end of testing, the protocol data are analyzed and serve as the basis for the calculation of suitability (the proportion of correctly performed functions from all the tested ones), interoperability (the proportion of correctly sent and processed queries) and the accuracy for each component (the effective value of the temporary mismatch of protection operation (in relative units) for the protection of the diesel generator unit; the mismatch of the percentage of load distribution unevenness for the Load Balancing System).

System testing tasks include also functions of the checking of tools required for performing actions by the Analyst – analysis of content of the data exchange packages, data channel load statistics, work schedule by power units and electric load. This checking is possible only after certain time of software operation in the electric plant operation mode, and the only functionality attribute resulting from such checking is suitability.

Weighting factors are defined for the calculation of numeric values of suitability, interoperability and precision for each component and for each tool required for the Analyst work. Then attribute values are calculated via additive reduction and normalization:

$$M = \frac{\sum_{i=1}^{15} k_{m.n.i} \cdot k_{p.m.i} \cdot \alpha_i}{\sum_{i=1}^{15} \alpha_i}, \qquad (1)$$

where M – attribute value; $k_{m.n.i}$ – test coverage value; $k_{p.m.i}$ – testing results; α_i – metric's weighting factor.

As the basis for assessing practicality of the software used for the ship power plant monitoring and control, a survey was selected, with an example of a questionnaire and the results of the processed data given in Table 1.

Table 1. Data Processing Results

Attributes and indicators	Mean normalized estimate
1. Ergonomics	–
1.1. Self-description of the interface	0.82
1.2. Simplicity of frequent operations	0.84
1.3. Simplicity of complex operations	0.74
1.4. Acceptability of delays	0.92
1.5. Drawing of graphic components	0.84
1.6. Information completeness of secondary dialog boxes	0.88
2. Understandability	–
2.1. Intuitive clarity	0.82
2.2. Correspondence of software behavior to the expected	0.94
2.3. Usability of the component library	0.95
3. Adoption efficiency	–
3.1. Simplicity of mnemonic diagram recreation	0.88
3.2. Simplicity of restoration of software operation skills	0.84
3.3. Ease of incorrect settings detection	0.80

The following attribute values were calculated for the monitoring and control software of the marine electric generating plant basing on the results of testing: suitability – 0.91; interoperability – 0.86; precision – 0.81. In analyzing of all test phases' results it can be concluded that the values of «suitability» and «interoperability» attributes can be improved by system testing strengthening, and the improvement of precision is related to changes in software functionality.

The functionality value is calculated by similar algorithm: weighting factors are given for attributes and the numeric value is calculated via reduction and normalization. When setting factors for suitability and interoperability as 1, and for precision – as 0.8, the functionality value will be – 0.86.

There were m users involved in order to determine the parameters of the study. Based on the data obtained from the survey, the normalized average and the resulted statistical will be used as indicator.

$$S_j = \frac{1}{k} \cdot \frac{\sum_{i=1}^{m} \alpha_{ij}}{m}, \quad (2)$$

where k – normalization factor (maximum score – 10); α_{ij} – assessment of the j-index i-th user; m-number of users surveyed.

Further, introducing weighting factors has kind of importance to the additive convolution calculates attribute values

$$A_j = \frac{1}{\sum_{i=1}^{m_j} p_{ij}} \cdot \sum_{i=1}^{m_j} \left(S_{ij} \cdot p_{ij}^{(n)} \right), \quad (3)$$

where $p_{ij}^{(n)}$ weight of i-j-index of the attribute; S_{ij} – normalized average statistical value of i-index j-th attribute; m_j-number of indicators of the j-th attribute.

To visualize the results of the calculation are encouraged to use the metric radial diagram, with the help of which it is convenient to display the "covering" performance requirements (Fig.3).

Presentation of usability evaluation using a single number does not identify possible problem areas, but it allows you at different stages of development and implementation of software to determine the need for further changes to the results of the implementation.

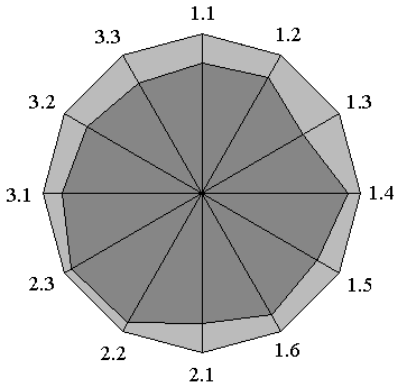


Fig.3. Coverage to the practical performance requirements.

The most important weights of the attributes, chosen among – the lowest calculated values

$$\min_{A_j} \max_{p_j} \left\langle A_j, p_j^{(a)} \right\rangle, \quad (4)$$

and then select the smallest value from the largest weight indexes of attributes that have been selected in the previous step

$$\min_{S_{ij}} \max_{p_{ij}} \left\langle S_{ij}, p_{ij}^{(n)} \right\rangle. \quad (5)$$

The result of this procedure will be having a variety of indicators, in the direction where the improvement is most important. To facilitate the calculation of minimax criteria, you can calculate the values of attributes and metrics based on the "reverse" weight, calculated as the cofactor 2 "true" weight.

$$A_j^* = \left(2 - p_j\right) \cdot A_j, \quad (6)$$

$$S^*_{ij} = \left(2 - p_{ij}\right) \cdot S_{ij}. \quad (7)$$

Then determine the most critical indicators in order to improve software usability which will meet determination procedure lowest values supplemented by attributes, and then – among the lowest values of indicators supplemented by

$$\min_{p_{ij}} \min_{A_j} \left\langle A^*_j, \left\{ S^*_{ij} \right\} \right\rangle \quad (8)$$

After calculation chart "covering" the requirements and the numerical value of practicality were obtained – 0.86. For example, the value and coating were found unsatisfactory, therefore it decided to modify the software. For that of the weights greatest attributes is chosen – 1.0, which stands for "for this software ergonomics is the most important". Further parameters of the weighting factors, which are components of ergonomics, the largest has been chosen. Two indicators correspond to the selected weight - "Simplicity of frequent operations" and "Simplicity of complex operations". Out of these indicators, the one with the smallest mean statistical value ("Simplicity of complex operations") is selected. It is in this direction that the changes are most important. Most often, however, the satisfactory usability requires the change of several interdependent indicators. Therefore, it is possible to exclude the received indicator and repeat the above procedure. The result for the situation in question is the indicator "Ease of detection of incorrect settings for data exchange." Changes in this direction will also significantly affect the improvement of usability.

This sequence of actions can be simplified by calculating the complemented values of attributes and indicators (the values are "complemented" to minimize the impact of attributes and indicators with small weights on the result). After calculating them, the least complemented value of the attribute is selected - 0.83 (attribute "Adoption efficiency") and for the corresponding attribute - the smallest complemented value of the indicator - 0.8 (indicator "Ease of detection of incorrect settings for data exchange"). To determine the second direction requiring changes, a similar procedure is carried out, but with the exclusion of "Adoption efficiency" attribute. The result of the latter procedure will be the indicator "Simplicity of complex operations."

The simplicity of the analysis, variability, stability and verifiability are the attributes of maintainability. Using the automaton approach and the device of pattern networks in software design and implementation simplifies the changes in the behavior of system components (by adding new states, input and output actions to the automaton) and the addition of new components (by creating new generators with the corresponding relationships of connection to the pattern network). The use of these methods also leads to a high level of analysis simplicity and verifiability of new software functionality.

The attributes of mobility include adaptability, flexibility of installation and replaceability. The developed software is compatible with 32-bit and 64-bit operating systems of the Windows family, it does not require the installation wizard and access to the registry and allows replacing certain modules and libraries without replacing the rest of the program.

The software reliability consists of three attributes: failure-free, error-proof and recoverability.

Among the indicators of software failure-free operation are the mathematical expectation and the mean square deviation of the mean time between failures, the conditional survivor function (the probability that the random mean time till the next failure will be more than specified). Calculation of these indicators is based on the intervals gained as a result of functional testing between successive failures, obtained experimentally and presented in Table 2.

Table 2. Statistics on failures

failure number	interval between failures, hours	failure number	interval between failures, hours
1	0.08	6	10
2	1	7	12
3	4	8	12
4	7	9	14
5	9	10	20

Mathematical expectation $m^*_{\Delta t}$, mean square deviation $\left[\sigma^2_{\Delta t}\right]$ and the conditional survivor function are calculated on the basis of statistical estimates of the numerical characteristics of the random time between failures:

$$m^*_{\Delta t} = \frac{1}{n_н} \sum_{i=1}^{n_н} \Delta t^{(i)} = \frac{1}{10} \cdot 89.08 = 8.908. \quad (9)$$

$$\left[\sigma^2_{\Delta t}\right]^* = \frac{1}{n_н - 1} \sum_{i=1}^{n_н} \left[\Delta t^{(i)} - m^*_{\Delta t}\right]^2 = \frac{1}{9} \cdot 33.75 = 3.75. \quad (10)$$

As an example of using the survivor function, an interval, the probability of which is more than 85%, is considered:

$$0.85 = 0.5 - \Phi\left(\frac{\tau - 97.88}{6.423}\right), \qquad (11)$$

$$\tau \approx 91.3 .$$

Thus, with a probability of 85%, it can be argued that the software will work reliably for more than 91.3 hours. This value can be increased by further collecting statistical data on failures, which will result in the recalculation of statistical quality indicators.

The software (SW) error-proof feature lies in its ability to perform functions under abnormal conditions. To determine abnormal conditions, it is necessary to highlight precedents that require the presence of third-party software and/or hardware. The highlighted precedents, their requirements for the availability of third-party tools and the reaction of SPS monitoring and control software are given in the table 3.

Table 3. Reaction of software to abnormal conditions

User actions	Software and / or hardware	Software response to failure
use-of-data precedents	hardware automation tools	indication of absence of connection and exclusion of request from the queue
forecasting result of load connection	Matlab Engine	inform. notification and automatic rejection of the action
preservation / restoration of the mimic and software environment; analysis of data on congestion; automation testing	operating system tools for accessing files	inform. notification
accumulation and analysis of data on the work of the DGS and the state of the loads	DBMS PostgreSQL	inform. notification possibility to change settings

The restorability of SPS monitoring and control software lies in restoration of the mnemonic diagram and component settings. This requirement is one of the main requirements for the software, so it is tested during functional testing.

The modern software development process is iterative and includes several stages: requirements analysis, architecture development, coding, testing and debugging, documentation, implementation and maintenance. Calculation of the key quality indicators for the considered category of software determines the need to return to earlier stages of development (analysis of requirements – coding) or transition to the following (documenting and further). After selecting the features in the direction of which it is necessary to make priority changes, the software interface is corrected and then the survey is conducted again and its results are processed in this way. This process is repeated until the "coverage" of requirements and the value of practicality are not satisfactory.

4 Conclusion

In work the methodology of a quantitative estimation of the basic indicators of the specialized software quality for monitoring and management of marine electric power systems – functionality, practicality and reliability is considered. Using this methodology allowed to obtain the composition of software attributes and quantitative values of metrics, specify the directions of priority changes and evaluate the results of their implementation, determine the reliability of software. Based on the analysis of all stages of testing, it is established that the functionality of the software can be enhanced by strengthening system testing and changing some algorithms of the software.

The software functionality enhancement is related to the strengthening of system testing (enhancement of suitability and interoperability) and to the upgrade of interaction algorithms between hardware and software automation (improving of precision). The software usability can be improved by upgrade of the means of specifying incorrect setting up for data exchange and by simplifying of the succession of the user moves when performing complex operations (data flow analysis etc.). Temporary software efficiency depends on topology and electrical power system composition as well as on the rate of traffic from software components, so the attributes should be calculated for each automated control system of the marine electric generating plant.

References:

[1] K. Esaki, System Quality Requirement and Evaluation, *Global Perspectives on Engineering Management*, Vol.2, No.2, 2001, pp. 52-59.

[2] T. Danilina, Software Quality Evaluation With Respect to International Standards, *Electronic and computer systems*, Vol.7, 2012, pp. 266-269.

[3] V. Kuliamin, A. Petukhov, Review of the Techniques for Constructing of Covering Sets, *Programming*, Vol.3, 2011, pp. 3-11.

[4] V. Kuliamin Component Architecture of Model-based Testing Environment, *Programming and Computing software*, Vol.36, No.5, 2010, pp. 289-305.

[5] Ashwin B. Tomar, Dr.Vilas. M. Thakare, A Systematic Study of Software Quality Models, *International Journal of Software Engineering & Applications*, Vol.2, No.4, 2011, pp. 61-70.

[6] Al-Suod Mahmoud M., A. Ushkarenko, O. Dorogan, Automatic Approach for Estimating the Protection Elements of Electric Power Plants, *World Academy of Science, Engineering and Technology, International Science Index, Energy and Power Engineering*, Vol.2, No.12, 2015, pp. 3935-3937.

[7] Al-Suod Mahmoud M., A. Ushkarenko, O. Dorogan, Monitoring and Automatic Control for Ship Power Plants Based Logical Algorithms, *International Journal of Advanced Computer Research (IJACR)*, Vol.4, No.17, 2014, pp. 966-972.

[8] O. Tarasiuk, V. Kharchenko, Dynamic Radial Metric Diagrams in Tasks of Software Quality Management, *Collection of studies of Pukhov Institute for Modelling in Energy Engineering*, Vol.22, 2003, pp. 202-205.

[9] Hussain Azham, Mkpojiogu Emmanuel, Usability metrics and methods for public transportation applications: A systematic review, *Journal of Engineering Science and Technology*, Vol.12, 2017, pp. 98-105.

[10] H. Rex Hartson, Terence S. Andre, Robert C. Williges, Criteria for Evaluating Usability Evaluation Methods, *International Journal of Human-Computer Interaction*, Vol.15, No.1, 2003, pp. 145-181.

[11] Premal B. Nirpal, K. V. Kale, A Brief Overview Of Software Testing Metrics, *International Journal on Computer Science and Engineering*, Vol.3, No.1, 2011, pp. 204-211.