# Analysis and assessment of functional stability of information systems supporting management processes

Oleksandr Dodonov[1], Olena Gorbachyk[1], Maryna Kuznietsova[1]

[1] *Institute for Information Recording of the National Academy of Sciences of Ukraine, 2, Mykoly Shpaka Street, Kyiv, 03113, Ukraine*

**Abstract**

The concept of functional stability of information systems and its characteristics are defined. We propose to build functionally stable information systems (IS) using the principles of multi-version diversity, multi-parameter adaptation, and multi-level management of system degradation. Features of specific implementation of multi-version of processes and products are described. A model of a multi-version system is presented. Indicators of functional stability are proposed, considering the functional capabilities of the system and its structural features. The problem of assessing functional stability taking into account the structure of IS is described. It is shown that the evaluation of the functional stability of the IS based on the parameters of the graph describing its structure is reduced to the clarification of the question of the viability of the graph or the structural viability of the IS. The methods and means of increasing the structural survivability of IS are outlined. Operational cycles at different levels of the IS hierarchy have been analyzed in order to ensure its functional stability. Ways of increasing the functional stability of information systems are considered, in particular, the technology of dynamic reconfiguration. The implementation of the dynamic IS reconfiguration management subsystem is proposed and the main functions of its components (monitoring, analysis, localization, selection and decision-making, implementation, database and knowledge modules) are analyzed. Measures to ensure the reliability of software and increase the functional stability of information systems are described

**Keywords**

Information system, functional stability, indicator of functional stability, dynamic reconfiguration.

## 1. Introduction

The functioning of most information systems (IS) involved in management processes takes place under the conditions of constant interaction with a permanently changing external environment. A significant part of such interactions are various informational conflicts that significantly affect the achievement of the system-wide goal. Information conflicts can lead to the destruction of information resources, violations of interaction regulations and regular information processes, as a result of which the performance of system and application functions is violated, and accordingly, management violations occur, which can represent a potential threat to human life and the environment in case of object criticality management.

To prevent emergency situations, it is important to assess the risks and develop means of forecasting, early detection, prevention, and countermeasures against destructive informational influences from the external environment. Otherwise, elimination of their consequences will require significant material and human resources.

XXII International Scientific and Practical Conference "Information Technologies and Security (ITS-2022)", November 16, 2022, Kyiv, Ukraine
EMAIL: dodonovua@gmail.com (O. Dodonov); bges@ukr.net (O. Gorbachyk); marglekuz@gmail.com (M. Kuznietsova)
ORCID: 0000-0001-7569-9360 (O. Dodonov); 0000-0001-8492-4478 (O. Gorbachyk); 0000-0001-6054-418X (M. Kuznietsova)

The development of functionally stable information systems involves several urgent tasks, including:

- the early detection and countermeasures against harmful informational influences of the external environment;
- practical assessment of the reliability of IS software;
- development and implementation of mechanisms for increasing the survivability of IS.

## 2. Basic provisions

Nowadays IS is the most important tool for supporting management processes and ensuring the safety of the operation of various management objects, in particular, critical infrastructures. In order to avoid violations in management processes, it is necessary to ensure that IS, which are components of automated organizational management systems, have such a property as functional stability, which allows IS to maintain and/or restore the performance of functions in conditions of various types of disturbing influences, minimizing the risks of transition to an emergency (dangerous) state of the control object [1].

Functional stability is a property that indicates the level of reliability, fault tolerance, survivability, and security in IS. Functional stability of IS is usually ensured by the introduction of a certain redundancy; implementation of the built-in control system; formation of a contour of protection against the negative effects of the external environment; using components with an increased level of security and reliability. However, additional redundancy leads to the deterioration of technical and economic characteristics of IS. Control systems may not always respond adequately to abnormal situations, and the probability of such situations is not reduced. The protection circuit minimizes the influence of external factors, but does not completely eliminate it. The choice of an element base with a higher level of security and reliability increases the fault tolerance of the IS, but does not ensure functional stability when the failure has already occurred. The functional security of IS is evaluated using various procedures and techniques [2,3], which are not sufficiently systematized and agreed on input and output parameters. To minimize the risks of inaccurate assessment, it is necessary to determine the order of their compatible and parallel application.

If in a time interval $\Delta t$ under any set of destructive influences $R_{\Delta_t}$ there is at least one workable distribution of IS resources, which will ensure the implementation of a defined set of functions $F = \{f_n\}$ with a quality level not lower than a given one in the presence of the state of communication in this interval $\alpha_{\lim}$, then the IC is functionally stable in the time interval $\Delta t$.

The operational failures flow, destructive effects of various origins (of diverse nature), intentional damage, and unauthorized interference, errors of service personnel can cause IS failures, leading to a decrease in functional stability or a complete loss of this property. But in the case of activation and application of mechanisms for ensuring survivability, such as reconfiguration and/or reorganization of resources, or by introducing additional resources from the outside, the lost property can be restored.

## 3. Assessment of functional stability

To reduce the probability of failures, IS can use methods and structural solutions based on multi-version diversity (diversity, N-version, multi-versity). This can prevent issues caused by physical hardware and software design defects, as well as external influences.

Multi-version systems use process and product diversity to minimize the impact of hardware and software defects. These systems have several product versions and backup channels created using different software and hardware versions.

A single-version system can be described like a four

$$S(1) = \{\hat{X}, \hat{Y}, \hat{Z}, F\},$$

where $\hat{X}$ – is the alphabet of input signals, $\hat{Y}$ – is the alphabet of internal states, $\hat{Z}$ – is the alphabet of output signals, $F = \{f_1, f_2, ..., f_n\}$ – is the set of performed functions. A multi-version system $S(N)$ is

described as follows:

$$S(N) = \left\{\hat{X}, \hat{Y}, \hat{Z}, F, \hat{V}, \Psi\right\},$$

where $\hat{X}$ – is the alphabet of input signals, $\hat{Y}$ – is the alphabet of internal states, $\hat{Z}$ – is the alphabet of output signals, $F = \{f_1, f_2, ..., f_n\}$ – is the set of executed functions, $\hat{V} = \{\hat{v}_1, \hat{v}_2, ..., \hat{v}_N\}$ – is the set of versions, $\Psi = \{\psi_1, \psi_2, ..., \psi_K\}$ – is the set of options for processing the results of version implementation.

There is a dependency between multi-version and single-version systems [2]:

$$S(N) = \left\{S(1), \hat{V}, \Psi\right\}.$$

A single-version system can be structurally redundant and have means $\Psi$, that process results from identical channels.

In multi-version systems, it is possible to apply several types of version redundancy [2]. Their set $R = \{r_1, r_2, ..., r_r\}$ can be decomposed into subsets by product versions and process versions. Various types of version redundancy $r \in R$ are accumulated, in turn, in the different product versions of the multi-purpose system. This is described by mapping using a Boolean matrix $\Xi = \|\Theta_{rj}\|$:

$$\Xi : R \to \hat{V}.$$

In this case, a multi-version system can be described as:

$$S(N, r) = \left\{\hat{X}, \hat{Y}, \hat{Z}, F, \hat{V}, \Psi, R, \Theta\right\}$$

or

$$S(N, r) = \left\{S(N), R, \Theta\right\} = \left\{S(1), \hat{V}, \Psi, R, \Theta\right\}.$$

For multi-version systems, the correspondence between multiple versions $\hat{V}_i$ of the backup channels of the system $C = \{c_1, c_2, ...c_q\}$ is also important [2], which can be specified by mapping using a Boolean matrix $Q = \|w_{ij}\|$:

$$Q : \hat{V} \to C$$

And accordingly, the model of the multi-version system will look as follows:

$$S(N, r, q) = \left\{\hat{X}, \hat{Y}, \hat{Z}, F, \hat{V}, \Psi, R, \Theta, C, Q\right\} = \left\{S(N.r), C, Q\right\}.$$

The principle of multi-version can be effectively supplemented in the design of functionally stable ICs by multi-parameter adaptation and multi-level controllability of degradation [2]. Multi-parameter adaptation involves the organization of several software-implemented and hardware-supported control loops of the reconfiguration procedure, considering the types and number of failed components, and multi-level control of degradation allows redistribution of excessive and non-excessive, but mobile resources (and correction, if necessary, goals of functioning) to minimize volumes of degradation.

Let's define the structure of the object Ŝ management system in the form of a triple:

$$\hat{\mathrm{S}} = \langle \mathrm{G}, \mathrm{R}, \mathrm{U} \rangle,$$

where G is the component set of the system, R is the set of rules by which the system Ŝ should function, U is the process of functioning, defined on the set G within the rules of R. IS is a part of the object's Ŝ management system .

Let's assume that in the absence of failures, the IS ensures the performance of a complex of object management tasks:

$$\Phi = (\phi_1, \phi_2, \dots, \phi_m).$$

The complex of tasks Φ is performed with the specified quality and the required efficiency, provided that the IS has resources to perform the functions of a certain set:

$$F = \bigcup_{i \in I} F_i = \{f_1, f_2, ..., f_n\}.$$

At the functional level IS, as a component of the object Ŝ management system, can be characterized as a set of functional modules performing functions from the set *F*:

$$\hat{\mathrm{s}} = (\hat{\mathrm{s}}_1, \hat{\mathrm{s}}_2, \dots, \hat{\mathrm{s}}_k), \ \hat{\mathrm{s}} \subset \hat{\mathrm{S}}.$$

Let us denote a set of critical management tasks, the failure of which will cause undesirable changes in the state of the management object, due to $\Phi^{\mathrm{kr}} = (\phi_1^{\mathrm{kr}}, \phi_2^{\mathrm{kr}}, \dots, \phi_{\mathrm{m}}^{\mathrm{kr}})$, $\Phi^{\mathrm{kr}} \subset \Phi$ .

Management tasks from a set of $\Phi^{kr}$ provide for the performance of a set of management tasks, independent or informationally related, in IS, which is possible if there are resources in the system for performing functions from a set *F*.

Suppose that $F^* \subseteq F$ – is some subset of the functions performed in IS. Then, if $d$ is such that:

(1) if any $d$ functional components $\hat{s}_j$ fail, the IS can perform any function from the set $F^*$ and, accordingly, the object $\hat{S}$ management system will ensure the performance of a set of critical management tasks $\Phi^{kr} \subset \Phi$.

(2) there will be such a set of $(d+1)$ functional components $\hat{s}_j$ and such a function $f_i \in F^*$, that in case of failure of $(d+1)$ functional components $\hat{s}_j$, the IS will not be able to ensure the performance of the function $f_i \in F^*$, then the value

$$P_{stab} = \frac{d}{n}$$

is defined as the coefficient of functional stability of IS and at the same time of functional stability of the object $\hat{S}$ management system.

It is clear that $0 \leq P_{stab} \leq 1$. . If it is impossible to fail any of the functional components $\hat{s}_j$ of the IS without losing the ability to perform management tasks from the set of $\Phi^{kr}$,, then $P_{stab} = 0$.
If IS resources are sufficient to perform all functions and, in addition, there is a sufficiently large functional redundancy or the possibility of interchangeability of functional components $\hat{s}_j$ in the IS, then $P_{stab} = 1$.

Different algorithms for calculating the functional stability index are possible for different classes of systems. And if the coefficient of functional overlap $K_f$ is known - a quantitative indicator (provided by the IS project) that determines the number of functions $f_i \in F^*$ performed by one functional component $\hat{s}_j$ of the IS, and characterizes the possibilities of functional interchangeability $\hat{s}_j$; $N$ – is the number of $f_i$, that must be implemented in IS to perform a set of critical management tasks by the object $\hat{S}$ management system, then it is possible to determine the upper estimate of the functional stability coefficient:

$$P_{stab} \leq \frac{K_f}{N} .$$

Since the upper estimate of the functional stability coefficient does not depend on the number of functional components in the IS, it is impossible to increase the functional stability of the system by simply reserving functional components.

Increasing the coefficient of functional stability is possible by increasing the coefficient of functional overlap or by "successful" selection of the set of functions *F* and the distribution of functions $f_i$ between the functional components $\hat{s}_j$ of the IS.

## 4. Evaluation of the functional stability of IS considering the properties of the system structure

IS, involved in management processes, to the class of complex organizational systems and are created on the basis of corporate computing technology. The main requirement for such systems is the guaranteed provision of users (managers) with access to distributed information resources, combined into a single information space, for solving object management tasks.

Assessing the functional stability of a distributed IS, provided that its main function is determined, such as data exchange between communication nodes, is usually reduced to solving the problems of graph connectivity analysis, assessing the probability of the existence of communication structures that make it possible to reach a specified node, and assessing the probability of the formation of a workable structure in IS in conditions of accumulation of damage to communication lines, etc.

An undirected graph $(V, L)$; $V = \{v_i\}$; $L = \{l_{ij}\}$; $i, j = \overline{1, M}$, serves as a model of the IC structure, where *V*– is the set of vertices of the graph corresponding to the set of communication nodes; *L* – a set of edges, corresponding to a set of communication lines between communication nodes. The graph is

described by the adjacency matrix:

$$A = \left\| a_{ij} \right\|, i, j = \overline{1, M}, \quad a_{ij} = \begin{cases} 1, if\ l_{ij} \in L \\ 0, if\ l_{ij} \notin L \end{cases}.$$

A distributed IS will be functionally stable if, when removing a vertex or an edge from the graph *G*, which is an IS model, a graph is obtained in which none of the following conditions is satisfied [4]:

- the graph consists of at least two components;
- there are no directed $(l_i - l_j)$ paths for defined sets of vertices;
- the number of vertices in the largest component of the graph G is less than some predetermined number;
- the shortest path $(l_i - l_j)$ is longer than some given value.

Evaluating the functional stability of the IS based on the parameters of the graph describing its structure is reduced to clarifying the issue of graph survivability or structural survivability of the IS. The construction of estimates of the structural survivability of IS is connected with the solution of the following problems:

- search for critical components of the network, in particular nodes, the removal of which leads to the disintegration of the network into unconnected parts;
- analysis of stability, elasticity and vulnerability of the network;
- estimations of load of arcs and capacity of the network;
- network resiliency analysis;
- search for options for construction and development of the network with the minimum value of the average path, the load of arcs and at the same time with the maximum throughput;
- construction of network connectivity indicators.

Calculation of connectivity indicators, such as the probability of connectivity under random graph edges, can be computationally challenging because it requires direct enumeration. At the same time, using the paths and sections of the graph simulating the communication network, it is possible to obtain fairly simple (compared to the exact methods of finding the relevant characteristics) marginal - upper and lower - estimates of the required indicator (Cesari-Proshan, Litvak-Ushakov estimates) [2].

The study of the connectivity of the majority of graphs, that is, the solution of the problem of whether a certain node - the source - can communicate with another certain node - the drain, in many cases does not provide an exhaustive criterion for answering the question about the quality of the functioning of the system with a network structure. Therefore, research has been conducted to find other relevant indicators of the quality of functioning of communication networks.

An important group of indicators of structural survivability, and accordingly structural stability, are the so-called "measures of survivability". When defining them, it was assumed that an intelligent adversary, knowing the structure of the network, tries to disrupt its functioning.

The network has a high survivability index, if it is necessary to "destroy" a large number of nodes and (or) edges in order to significantly impair or completely interrupt its functioning. Such a measure of survivability is conventionally called a "deterministic measure of reliability" and is used in the initial planning and development of communication networks, when there is a shortage of statistical data on the quality of the network's functioning.

In mathematical graph theory, survivability measures are often interpreted as quantitative measures of connectivity for the structure of a graph: minimum cut, nodal connectivity, generalized connectivity, path length, etc.

Assessing functional stability and analyzing structural survivability are crucial tasks at the initial stages of creating a distributed IS communication structure. It involves estimating the maximum flow that can be transmitted in the network when elements fail under permissible quality of functioning. When evaluating the survivability of communication networks, depending on the transmission technologies used in them, the presence of various types of traffic (audio and video information, data, compressed video and audio), various categories of services, the probability of data loss, and requirements for the quality of processing are considered. Packet data transmission can increase the structural survivability of systems. Information packages can take different routes between network nodes, depending on channel performance and load. In the event that part of the communication

channels is inoperable, destroyed, and the network remains survivable, information will still be delivered to the destination node via other working channels, and functional stability will be ensured.

The application of ring topology provides the possibility of automatic switching of channels to backup ones in the event of any emergency situations. For example, the SDH (Synchronous Digital Hierarchy) equipment provides the redundancy of lines and hardware units according to the 1+1 scheme. This allows for automatic traffic switching to the backup direction and restorative work without interrupting traffic.

Reserving frequently used arcs and nodes in the network is a known method of increasing the structural survivability and functional stability of distributed IS. Redundant networks can be created using RSTP technology (Real Time Spanning Tree Protocol), connection of network segments in pairs (Redundant Coupling), dual connection (Dual Homing), "trunk" connection (traunking) and technology of redundant ring structures (for example Hirschmann HIPER-Ring).

## 5. Reconfigurations in IS as a means of increasing functional stability

The functional stability of IS can be characterized through the operational cycle Ω, which includes such operations as: failures` prediction $w_1$, warning $w_2$, detection $w_3$, localization $w_4$, isolation $w_5$, parrying $w_6$, reconfiguration procedures $w_7$ and information recovery $w_8$. The sequence and time limits of operations depend on the specifics of the defects and features of the IS.

It is also possible to define appropriate operating cycles for various defects, for example, for software component vulnerabilities we have [2]: $w_{pr1}$ – prediction of possible characteristics of attacks on this vulnerability (possibility, method, time parameters); $w_{pr2}$ – intervention warning; $w_{pr3}$ – attack detection through input data control; $w_{pr4}$ – localization (isolation) of the component on which the attack was carried out; $w_{pr7}$ – reconfiguration that will compensate for a possible failure due to a vulnerability attack; $w_{pr8}$ – continuation of service operations and selection of stationary configuration.

The operational cycle $w_7$ - reconfiguration - is extremely important for increasing the functional and structural stability of the IS. Reconfiguration is, in general, a process of changing the structure, parameters, and technologies of functioning to restore the performance and efficiency of the system to the required level or minimize the decrease in these indicators during functional degradation. The main difficulty lies in the need to determine the moment of application of the reconfiguration procedure, rules and algorithms for the redistribution of available resources and the formation of new structural connections, ensuring the continuity and quality of object management.

IS means support the completion of management tasks by accumulating, processing, storing, and transmitting information. In the IS, a subsystem of information interaction arises, which is generated by the organizational structure of solving the management task. This subsystem consists of switching nodes and communication channels between individual elements. An undirected graph can serve as a mathematical model of such a support subsystem for the management task:

$$\hat{G}(V, L);\ V = \{v_i\};\ L = \{l_{ij}\};\ i, j = \overline{1, M},$$

where $V$ – is the set of vertices of the graph corresponding to the set of switching and information processing nodes; $L$ – is the set of edges corresponding to the set of connections between nodes.

Let's assume that the IS will perform the main function - ensuring information interaction, thanks to the exchange of data between nodes of switching and information processing - if there is at least one data transmission route. In this case, the requirement of the functional stability of the IS is transformed into the requirement of the connectivity of the graph $\hat{G}$, which forms the basis for the quantitative evaluation of the IS's functional stability, based on IS's topology.

For example, the failure of the switching and information processing node of the information interaction subsystem or the loss of connections between nodes due to their physical destruction or violation of data integrity can make it impossible to perform the tasks of providing information interaction for object management; deterioration in the functioning of IS (reduction in productivity, capacity of communication lines, etc.); distortions or defects in the functioning algorithms of communication and information processing nodes; reduction of structural redundancy, stock of resources; deterioration of the functioning of IS elements or degradation of the entire system; fatal

loss of operational efficiency of the IS, etc.

Each IS communication and information processing node $v_i$ is a functional component of the system, which is characterized by its functional purpose - a set of functions $F_i \subseteq F$. Functional specialization of nodes occurs by installing appropriate software and establishing the necessary connections for information exchange. The modular principle of software development allows to form the necessary configuration of the IS subsystem to perform the functions of accumulating, processing, storing and transmitting information to support the solution of a specific management task. The functionality of the $v_i$ node can be expanded, if necessary, by connecting new software modules.

Dynamic reconfiguration technology is implemented to ensure the continuity of the management process, the guaranteed performance of management tasks even in the presence of unwanted influences on the IS, which can lead to the failure of switching and information processing nodes, disruption of connections between them in the IS.

The implementation of this technology involves setting and solving various classes of problems of structural dynamics management, in particular, problems of planning and management of processes of processing and transfer of resources during restructuring of the structure. Therefore [1], dynamic reconfiguration is not only a technological solution to compensate for failures in the IS, but also an independent management process to ensure the operational redistribution of functions and resources between nodes (functional components of the IS) and increase the efficiency of the functioning of the IS.

Planning procedures for dynamic IS reconfiguration requires solving the following problems:

- construction of IS reconfiguration scenarios under conditions of destructive influences on the system or control object;
- development, analysis and multi-criteria synthesis of plans for functional and structural reconfiguration in conditions of permanent change in the operating environment;
- research through analytical and simulation modeling of the conditions for the reconfiguration procedures implementation.

Dynamic reconfiguration procedures in IS can be implemented by a separate IS subsystem, which should include a monitoring module, an analysis module, a localization module, a selection and decision-making module, a decision implementation module, and a database and knowledge base (DBKB) [1].

The monitoring module is designed to collect data on the state of the IS and its components. Data collection takes place in accordance with the defined list of monitoring indicators. The received data are recorded, indicating the exact time of their receipt, in the appropriate tables of the specialized DBKB.

The analysis module is designed to detect deviations in the functioning of the IS and its components. In the work of the analysis module, the values of the selected indicators (obtained at the monitoring stage and stored in the specialized DBKB) are used. The detection of critical deviations of the indicators from the normative or limit values allows to identify the IS components that are potentially dangerous for its stable functioning, as well as to assess their impact on the functioning of other components. The indicator values obtained at the monitoring stage are constantly checked for compliance with the limits set for each indicator. The analysis module forms a list of potentially dangerous IS components, which is passed on to the localization module for processing.

Therefore, the analysis module, based on the monitoring data, diagnoses the state of the IS, determines the criticality of the detected deviations and produces a command for further working out the situation. The results are recorded in DBKB. IS components identified at the analysis stage, which are vulnerable to unwanted influence, must be localized.

The localization module is designed to further work out the existing situation, in case of receiving a command from the analysis module to localize dangerous IS components that have critical deviations. Localization means actions directed against the spread and implementation of a threat to the functioning of the IS, therefore, the main task at the stage of localization is to perform actions that will exclude dangerous IS components that are under unwanted influence from the working configuration of the system in the shortest possible time.

The main functions of the localization module: calculation of the localization area; selection of localization methods; localization of components that showed critical deviations in functioning; formation of a list of users who need to be informed about localization results; forming and sending

messages to users about localization results; formation of a list of operational functional components of the IS.

After the localization of IS functional components vulnerable to unwanted influence is completed, a list of functional components that can potentially be used for reconfiguration - redistribution of functional tasks is formed. The list formed at this stage will be used at the stage of selection and decision-making regarding the application of the reconfiguration procedure.

Thanks to the localization of dangerous IS components vulnerable to unwanted influence, the functioning of the IS is stabilized and the further spread of unwanted influence is stopped. However, this step does not return the system to normal working condition.

The selection and decision-making module is designed to form a decision on the selection and application of the reconfiguration procedure to restore the process of information support for solving management tasks. The main functions of this module: definition of functional tasks for redistribution; determination of limitations for each of the functional tasks to be redistributed; definition of the list of functional components of IS and their properties for redistribution of functional tasks; solving the problem of redistribution of functional tasks taking into account the specified restrictions.

First, the selection and decision-making module prepares a list of functional tasks, in the performance of which functional components vulnerable to unwanted influence were involved at the time of localization, as well as tasks that must be performed by these functional components of the IS in the future. As a result of gathering information, two groups of functional tasks are formed: (1) – tasks, the execution of which was interrupted during the localization process and which were not completed in the normal mode, (2) – tasks that must be performed after the completion of the interrupted tasks according to the requirements of the technological process, which is realized. For each of the tasks from these two lists, you need to construct constraints that formalize the properties and requirements of the functional tasks. In particular, their priorities are determined; restrictions on user access rights, etc. are set [1].

The second block of input information for solving the problem of redistribution of functional tasks is a list of IS components that are operational at the moment and on which functional tasks from the list built earlier can potentially be performed. This list is formed on the basis of the list of components built in the last step of the localization module. Naturally, the list of components for redistribution may be smaller than the list of IS components generated by the localization module. The next step is to form a list of properties for each functional component of the IS from the last list (performance, current load, scheduled load, etc.). Prepared information about functional tasks, constraints, a list of functional components and their properties are input information for solving the reconfiguration task and obtaining a set of solutions that are ranked according to certain criteria, and the one with the highest rank is chosen. The obtained results are recorded in DBKB.

The implementation module is designed to implement the decision made at the previous stage. Functions of the solution implementation module: implementation of the selected reconfiguration procedure; formation of a list of users who need to be notified of the results of reconfiguration [1]. The implementation module starts the execution of tasks that were terminated in an emergency as a result of the localization process, on the specified IS components, makes the necessary changes in the structure of the sequence of execution of the set of tasks and makes these changes in the task manager, which monitors the sequence of their execution. Since part of the tasks with a low priority can be excluded from further execution, this leads to a change in the structure of relationships between tasks.

The data and knowledge base is a specialized database (DBKB) designed to store data and knowledge necessary for the functioning of the dynamic reconfiguration subsystem: a list of indicators and their parameters (limit and normative values); time series of indicator values obtained as a result of monitoring; a list of IS components to be localized; localization methods and mechanisms. The need to use DBKB is based on the fact that the amount of data received for storage per unit of time is quite large, it depends on the number of indicators that are monitored, as well as the frequency of data acquisition. For such purposes, an industrial relational DBMS cannot be used, as it is not designed to solve problems related to large volumes of data.

The dynamic reconfiguration subsystem should work in the background, performing a continuous process of monitoring the state of the IS and its components according to the specified indicators. In the case of detection of deviations from the set standard mode of operation, which may lead to or have

led to abnormal situations, the dynamic reconfiguration subsystem is activated.

Practical experience shows that the implementation of the dynamic reconfiguration procedure allows us to increase the functional stability of IS and, consequently, the quality of information support for the performance of management tasks, in particular, to ensure the safety of the functioning of critical infrastructure objects and infrastructure as a whole.

## 6. Functional stability and reliability of IS software

The analysis of the functional stability of IS involved in management processes is, first of all, based on a complex representation of systems as such, which must ensure the performance of specified functions (provision of relevant services) under conditions of occurrence or manifestation of physical, design defects of hardware and software, which lead to errors and failures; the appearance of failures as a result of violations of interaction with physical and informational environments with variable parameters of the influence of those interventions; a possible change in the requirements for IS services and the occurrence of unspecified failures.

The study of failures and disasters in complex management systems of the banking, administrative and industrial sectors revealed a variety of situations in which errors and failures in the functioning of IS were caused by software defects (software). Software defects may prevent the implementation of reconfiguration procedures, which are mostly performed by the corresponding software modules. Therefore, software reliability assessment has become an integral part of projects to create functionally stable IS.

IS software tools involved in management processes process a huge amount of information, and it is practically impossible to identify all connections and ways of data processing, even for standard rather simple programs. Therefore, implementing an experimental assessment of the real reliability of IS software, which is a complex combination of interacting software modules, is a rather time-consuming and difficult to automate task. Although it should be noted that today considerable experience has been accumulated in assessing the reliability of software used in IS of critical areas [5]. In such areas, it is impossible to use the functioning of real objects for testing and evaluating the reliability of software, and as a result, methods and means of modeling the external environment have become useful for the automated generation of tests for evaluating the reliability of IS software. On the basis of software models and components of real IS, simulation test benches are created, which provide an opportunity to evaluate the reliability of the functioning of specific software under conditions of regular and critical external influences that correspond to the real characteristics of the external environment.

Reliability of IS software is usually defined as the ability of a set of programs to perform specified functions while maintaining the value of certain indicators within specified limits over time. There are technologies for the development of reliable software complexes, proposed methods and models for researching their reliability and safety, but a single universal approach to solving the problem of creating reliable software has not been proposed [5]. The reason for this is the uniqueness of each software complex.

Each project should purposefully create a coordinated set of methods and means of ensuring the given reliability of the software under the conditions of a realistically possible reduction in the level of defects and program development errors. It is necessary to study the specific factors that affect the quality of the functioning of the software from the side of actually existing and potentially possible defects in the programs implemented in the IS [6].

The basic principles of creating reliable software can be divided into four groups: avoiding, detecting, correcting and making errors. Avoidance of errors combines principles, the observance of which will ensure the minimization of errors in the software development process. Error detection relies on error correction mechanisms. When, during the testing process, errors are detected in the components or the software as a whole, then the components or the software as a whole are refined. Testing continues until the specified reliability index is reached. Error tolerance involves the means and methods of ensuring the correct execution of a given function in the presence of errors.

In the general case, IS software is a collection of individual program modules (formally independent parts) connected by probabilistic links. Such modules can be software complexes,

individual programs, blocks or even operators. The number of software modules can be significant, so modules are grouped by type. Each type contains software modules that are similar in properties, in particular, in terms of reliability. Knowing the structure of IS software, which is a collection of software modules with known reliability indicators, applying the decomposition method, it is possible to assess the reliability of IS software. So, for example, let's assume that the IS software is a set of M program modules, then, taking into account the structure of the software and the operation of the IS, you can build a corresponding stochastic graph that will have M+2 vertices. Vertex 0 will be the initial (loop), and vertex M+1 will be the final (drain). Each software module is called to work with a given probability, which is determined based on the task of operation or initial data. The probability of error-free operation of IS software can be determined by the probability of error-free operation of all software modules and the probability of transitions between them. Usually, the reliability indicators of the software modules that are part of the software, as well as their probabilistic dependencies in the stochastic graph, are known. So, under these conditions, it is possible to assess, for example, the reliability of such complex software as in automated system of organization management, using the decomposition method.

Reliability control of software components is necessary at all stages of the IS life cycle. One of the effective ways to increase the reliability of software is the standardization of technological processes and objects of design, development and support of programs. Prevention of errors in software is possible thanks to high-quality documentation in the process of developing software modules and software as a whole.

## 7. Conclusions

Functional stability is an important characteristic of IS, especially for those involved in management processes. The indicator of the functional stability of the IS significantly depends on the basis on which the system is created, in particular, the use of multi-version, the principles of multi-parameter adaptation to failures, increasing the reliability of IS software etc. "Successful" functional overlapping and distribution of functions and resources between IS components allows to increase the indicator of functional stability, thanks to the organization and implementation of dynamic reconfiguration procedures.

## 8. Acknowledgements

The authors are grateful to colleagues who took part in discussions on research materials at scientific and scientific-technical seminars and conferences.

## 9. References

[1] Oleksandr Dodonov, Olena Gorbachyk, Maryna Kuznietsova. Dynamic Reconfiguration in Automated Organizational Management Systems. In: CEUR Workshop Proceedings vol. 2859, pp.129-141 (2020). http://ceur-ws.org/Vol-2859/paper11.pdf

[2] Kharchenko, V.S., Yakovlev, S.V. (Eds.): Provision of Functional Safety of Critical Information-control Systems. Konstanta, Kharkov (2019). 272 p. (in Ukrainian)

[3] Korolev, A.N.: Functional Stability of Navigation and Information Systems. In: University news. Instrument making, vol 61, no.7. Pp.559-565 (2018) (in Russian)

[4] H. Frank and I.T. Frisch Networks, Communication and Flows. Translation from English. Ed. D.A.Pospelov. Svyaz, Moscow (1978). 448 p. (in Russian)

[5] Letychevsky O.O., Peschanenko V.S., Hryniuk Y.V., Radchenko V.Yu., Yakovlev V.M. An Overview of the Modern Methods of Security and Protection of Software Systems // Cybernetics and system analysis, vol. 55, pp. 840-850 (2019).

[6] Shen Z., Chen S.: A Survey of Automatic Software Vulnerability Detection, Program Repair, and Defect Prediction Techniques. Security and Communication Networks, vol.2020. Article ID 8858010 (2020). https://doi.org/10.1155/2020/8858010.