Available online at www.sciencedirect.com

ScienceDirect

Procedia Manufacturing 44 (2020) 318–325

Procedia MANUFACTURING

www.elsevier.com/locate/procedia

1st International Conference on Optimization-Driven Architectural Design (OPTARCH 2019)

Method for Designing Optimal Usable Structures of Computer System for its Emergency Reconfiguration

Stefan Rozmus*, Andrzej Kozina

Institute of Computer and Information Systems, Faculty of Cybernetics, Military University of Technology, ul. gen. Sylwestra Kaliskiego 2, 00 – 908 Warsaw 46, Poland
Department of Public Administration, Cracow University of Economics, ul. Rakowicka 27, 31-510 Cracow, Poland

Abstract

The work deals with the reconfiguration of computer system, which was designed to provide services from a strictly defined set. We assume that the system is fully usable, as long as each randomly appearing request to perform any service from this set will be handled. When such a request occurs, the technical resources required for its fulfillment must be allocated to the service. In the general case, the same technical resources can be dedicated to perform various services at different moments of time. Therefore, it is advisable to introduce abstract resources, so-called functional ones, representing particular types of actual technical resources. Functional resources mediate in linking the services and technical resources required to perform them. One functional resource can be associated with only one service, whereas it can be performed by various technical resources. Each set of pairs <*service*, *functional resource*>, constituting a subset of the full set of such pairs, determined at the design stage of the system, creates the so-called the system's functional structure. In the case of damage of technical resource, it cannot be ruled out that it is impossible to maintain full usability of the system, i.e. to create a complete usable structure. However, as a rule, it will be possible to create more than one of its substructures, each of them provides support for service requests from a subset of the established set of services. If it is permissible to provide services to a limited extent for established subsets of services and for each of them a loss function resulting from the inability to provide a given service was specified, the system should be reconfigured (using available technical resources) so as to create a usable structure, at which potential losses will be the smallest..

The results of previous works are as follows:

- an algorithm for interactive fixing acceptable substructures by a user, reducing to the necessary minimum the number of required indications of service subsets (n services is 2^n possible subsets). Indeed, the choice comes down to establish vectors of minimal reliability functions. That algorithm has been implemented,

* Corresponding author.
E-mail address: stefan.rozmus@wat.edu.pl (S. Rozmus)

10.1016/j.promfg.2020.02.237

- a general algorithm for determining optimal functional substructures at a given state of fitness of the system. It can be the basis for the design of the layer for controlling automated system reconfiguration.

The presented approach allows to reduce losses in the case of a lack of full possibilities of supplying services by the system. The proposed method, after some extensions, can be applied to reconfigure non-computer systems, e.g. the state security system.

Keywords: Usable structure; system; reliability state; functional resource; technical resource; service; usability;

1. Introduction

From the user's point of view, the computer system is perceived as an object intended to provide specific services [1], [2], for the implementation of which it was designed. For the systems considered in the paper, the full set of services is strictly determined. By default, service requests will flow into the system randomly during its operation. For the current performance of each service, it is necessary to allocate appropriate technical resources. As a rule, such allocation may be made in many different ways. For the user, it is not significant which technical resources will be utilized for a given service. It is important, however, that it was done to the extent that fully satisfies the user. Thus, it can be assumed that the technical resources are perceived by the user as virtual elements of the computer system, capable of performing the assigned operations in the frame of the services. The technical resources seen in such a way will be further referred to as functional resources. Talking about a functional resource, we abstract from its physical form. Indeed it is obvious that every functional resource should have its technical representation.

At the design stage of the computer system, it is necessary to decide on the basis of which technical resources will be performed the services predicted for implementation in the system. It is convenient to make such decisions in two stages. In the first one, functional resources necessary for the implementation of each service should be specified. In the second stage, after selecting the technical resources, the links between functional resources and appropriate technical resources should be defined. It is not difficult to notice that in the general case there may be many variants of such links. As a result, the possible technologies of service implementation considered by the designer determine for each service reasonable subsets of functional resources ensuring its performance. In this way, a set of potential, proposed usable structures of the system is established. The usable structure constitutes a set of services with functional resources linked to each of them. At the stage of technical feasibility study, one of these potential usable structures [3] is generally selected for further design work, considered the best in the sense of the adopted selection criterion, determined by the possibilities of technical implementation of the system. They may result, for example, from the current financial condition of the system contractor, IT competences of future users or the availability of technical resources necessary to implement the system. The usable structure is a convenient initial concept for formulating the system's performance properties, because such a structure constitutes the link between the future user of the designed system and its designer. The user, without going into technical details, can specify potential usable structures. The task of the designer is then to find the answer to the question whether the proposed usable structures can be implemented with a fixed set of technical resources or not.

2. Problem description

If in a given situation (environmental conditions, physical state of system resources) the system is able to provide any service from the full set of services to be performed, then the user should declare the system's usability. In this case, there is often talk of the so-called full system usability. Often, however, even if it is possible to provide only a subset of these services, the user may considered the system as usable (partial system usability [4],[5]). Therefore in some emergency situations it is allowed to operate the system in a limited scope, but still satisfying the user.

The system's ability to perform subsets of services is determined, in fact, by its physical condition and thus, in our interpretation, its possible usable structure. For each subset of services that the system can perform (and thus for each usable structure), the user should be able to decide whether to consider the system usable or not. There must therefore be a binary function which assigns bivalent user decisions to each subset of services to be performed (1 - system still useable, 0 - system not useable), hereinafter referred to as the function of usability, denoted by φ. Applying it, a set of acceptable, but only from the user's point of view, system usable structures will be created.

When considering the reliability of any object, in the general case, the following three factors should be considered: services that the object will be able to perform, its predicted physical states, and external conditions during the performance of services. If the object is treated as a system, it is necessary to introduce the so-called generalized structural reliability function (let us denote it by f), whose arguments correspond to the three factors mentioned above. The values of f determine the distinguished reliability states of the entire system. In practice, it is assumed that the physical states of system elements are represented by their reliability states, while the external conditions are divided into a small quantity of numbered normative areas. The set of potential services is finite and strictly defined (only such are taken into account in this work). The formal arguments of the considered function $f(\mathbf{L},\mathbf{Y},\mathbf{Z})$ may be expressed in the following form:

$$\mathbf{L}\langle \mathrm{l}_1,\ldots,\mathrm{l}_i,\ldots,\mathrm{l}_I\rangle, \quad \mathbf{Y}=\langle \mathrm{y}_1,\ldots,\mathrm{y}_n,\ldots,\mathrm{y}_N\rangle, \quad \mathbb{Z}\in\{\mathrm{z}_1,\ldots,\mathrm{z}_\xi,\ldots,\mathrm{z}_\Xi\} \tag{1}$$

where: $\mathrm{l}_i=1$ means that the service numbered by i belongs to the current subset of services to be performed, $\mathrm{l}_i=0$ – that it does not belong, y_n denotes the reliability state of the nth element of the system (technical resource); it is assumed that the elements are divalent in the sense of reliability, i.e. $\mathrm{y}_n\in\{0,1\}$ wherein $\mathrm{y}_n=1$ means that the nth element is useable and $\mathrm{y}_n=0$, that damaged, z_ξ means the number of the ξth distinguished normative area of external operating conditions of the system.

It is assumed that the system is bivalent in the sense of reliability, i.e. the function f takes its values from the set $\{0,1\}$, wherein the value 1 means the system's usability, and the value 0 the opposite state. It may be noticed that the generalized structural reliability function f can be used to determine a set of acceptable usable structures, because it associates through its arguments, the user's requirements for usability (appropriate **L** vectors) with the possibility of providing it under given conditions and the physical state of the system (arguments $\mathbb{Z}$, **Y**). However, it may be assumed that the system will be designed for a fixed nominal area of external conditions (for the given value of $\mathrm{z}\in\mathbb{Z}$). So practically the function f will come down to the partial function of only two arguments (**L** and **Y**), which will slightly simplify the problem of its identification. To reduce the number of symbols used, we also denote this partial function using the symbol f, i.e.:

$$f(\mathbf{L},\mathbf{Y},\mathrm{z})=f_{\mathrm{z}}(\mathbf{L},\mathbf{Y})=f(\mathbf{L},\mathbf{Y}) \tag{2}$$

where the value "z" indicates the established nominal area of the external conditions. Therefore, for further considerations it was assumed that the generalized structural reliability function is a binary function $f(\mathbf{L},\mathbf{Y})$, where **L** and **Y** are binary vectors. It is convenient to assume that f is a Boolean function, because it will allow to use known methods of studying such functions [6],[7],[8].

As already mentioned, the change in the reliability status of technical resources usually leads to a change in the system's usable properties, which in turn necessitates the selection of a new usable structure. This should be the best structure in the sense of a fixed quality measure, feasible with changed reliability of technical resources. It is the user's responsibility to define this quality measure in detail.

It can be seen that for each subset of services that can be performed, described by the **L** vector, there is exactly one usable structure being a substructure of the complete usable structure. Thus, the vector **L** uniquely identifies the corresponding usable structure. In view of the above, and assuming that there is no technical impact of the usable structure implementation on the quality of the system performance, the objective function within the problem of choosing the optimal usable structure (lets denote it W) will specify the expected losses for any **L** vector. It may be noted that the expected losses will be, generally speaking, inversely proportional to the number of services possible to perform with the usable structure determined by the **L** vector. If $\mathbb{L}$ denotes the set of all **L** vectors and the symbol $\prec$ means the corresponding dyadic preceding relation in $\mathbb{L}$, then formally this feature of the proposed objective function can be expressed as follows:

$$\forall \mathbf{L}_1, \mathbf{L}_2 \in \mathbb{L}: \quad \mathbf{L}_1 \prec \mathbf{L}_2 \Rightarrow W(\mathbf{L}_1) \geq W(\mathbf{L}_2) \tag{3}$$

wherein $\mathbf{L}_1 \prec \mathbf{L}_2$ when the subset of services described by the $\mathbf{L}_1$ vector is included in the subset of services described by the $\mathbf{L}_2$ vector (the relation $\prec$ will be described in detail later in the paper). This property of the W function causes that the area of searching for the optimal structure can be significantly narrowed. Thus it is sufficient to consider only those of the acceptable usable structures that are not substructures of any other admissible usable structure.

3. Formulation of the research task

The research task presented in this work comes down to developing a method for the designer to choose the best usable structures for all possible reliability states of the technical resources of the system. This task can be accomplished in the three stages: stage 1 - developing a method for the user to determine acceptable usable structures (the identification of function φ); stage 2 - elaborating a method for determining the generalized structural reliability function (the identification of function f); stage 3- developing a method for selecting the best usable structure in the sense of the objective function W. Usable structures determined in such a way will form the basis for the development of the so-called table of automatic system reconfiguration, assigning each system reliability state to the best usable structure to be used at this state.

4. Results obtained

4.1. The method of the identification of function φ

As a rule determining (identifying) this function is not a simple task because in relation to the set of I services one should ask the user 2^I questions (there are so many possible subsets). With the increase of I the number of these questions increases rapidly (exponentially) so determining the function φ becomes practically unreal. Therefore it has become necessary to elaborate a conversational method that minimizes the number of user inquiries and responses to identify the φ function. Formally, this function can be described as follows:

$$\forall \mathbf{L} \in \mathbb{L}: \quad \varphi(\mathbf{L}) = \begin{cases} 1 - \textit{the user considers the system useable} \\ 0 - \textit{the user considers the system not useable} \end{cases} \tag{4}$$

Thus the function φ defines the whole set of usability states of the system. The essence of function φ follows that it is a non-decreasing monotonic function [9]. Meanwhile, if the user considers the system to be useable for the ability to perform a certain subset of services, then, if it is possible to perform the superset of this subset, it should also be consider useable. Furthermore, let us note that the function φ is a coherent one [4]. Determining the states of usability can be reduced to the task of determining the minimum vectors of function φ . It should be noticed that the developed method is essentially different from the methods known in the literature for determining the minimum vectors of monotonic Boolean functions determined algebraically using Boolean formulas, e.g. formulas minimizing [6],[7],[8]. Before moving on to further considerations, it is necessary to define the relationship $\prec$, which was mentioned in chapter 2.3.

Let $\mathbb{B} = \{0,1\}$ mean the set of Boolean values (false and true, respectively), $\mathbb{L}$ still denotes the set of all I dimensional Boolean vectors **L**, defining subsets of services, and let the relation $\propto \subset \mathbb{B} \times \mathbb{B}$ be the relation

$$\propto = \{(0,0),(0,1),(1,1)\} \tag{5}$$

Then the relations $\prec \subset \mathbb{L} \times \mathbb{L}$ will be the relation determined as follows:

$$\forall \mathbf{L}', \mathbf{L}'' \in \mathbb{L}: \quad [\mathbf{L}' \prec \mathbf{L}''] \Leftrightarrow [\forall 1 \leq s \leq I : l'_s \propto l''_s] \tag{6}$$

where $\mathbf{L}'=\langle l_1',...,l_i',...,l_I'\rangle$, $\mathbf{L}''=\langle l_1'',...,l_i'',...,l_I''\rangle$. The vector $\mathbf{L}'$ is called the predecessor of the vector $\mathbf{L}''$, and the vector $\mathbf{L}''$ - the successor of the vector $\mathbf{L}'$. The interval $[\mathbf{L}',\mathbf{L}'']$ specified in the space $\mathbb{L}$ is the set of these vectors $\mathbf{L}$ that are simultaneously the successors of the vector $\mathbf{L}'$ and the predecessors of the vector $\mathbf{L}''$. It should be noted here that both $\mathbf{L}'\prec\mathbf{L}'$, and $\mathbf{L}''\prec\mathbf{L}''$. The function φ is a monotonic non-decreasing Boolean function if and only if for each ordered pair of vectors $\mathbf{L}^{'},\mathbf{L}^{''}\in\mathbb{L}$ the following implication is true:

$$[\mathbf{L}'\prec\mathbf{L}'']\Rightarrow[\varphi(\mathbf{L}')\le\varphi(\mathbf{L}'')] \tag{7}$$

As the minimum vector of the function φ we will call any such vector $\mathbf{L}\in\mathbb{L}$, for which $\varphi(\mathbf{L})=1$, and for all predecessors of this vector different from $\mathbf{L}$ the value of the function φ is equal to 0. If the set of minimum vectors we denote by $\mathbb{L}_{\min}$, then the formal record of the minimum vector of φ function is as follows:

$$[\mathbf{L}\in\mathbb{L}_{\min}]\Leftrightarrow\left[(\varphi(\mathbf{L})=1)\wedge\{\forall\mathbf{L}_p\in\mathbb{L}:[(\mathbf{L}_p\prec\mathbf{L})\wedge(\mathbf{L}_p\ne\mathbf{L})]\Rightarrow(\varphi(\mathbf{L}_p)=0)\}\right] \tag{8}$$

Let the vectors $\mathbf{a},\mathbf{b}\in\mathbb{L}$, $\mathbf{a}=\langle a_1,...,a_i,...,a_I\rangle$, $\mathbf{b}=\langle b_1,...,b_i,...,b_I\rangle$ denote, respectively, the lower and upper boundaries of any interval specified in space $\mathbb{L}$, and the vectors $\mathbf{1},\mathbf{0}\in\mathbb{L}$, $\mathbf{1}=\langle 1_1,...,1_i,...,1_I\rangle$, $\mathbf{0}=\langle 0_1,...,0_i,...,0_I\rangle$ mean, respectively, a vector representing the full set of services of the designed system and a vector representing empty set. Each interval $[\mathbf{a},\mathbf{b}]$ in space $\mathbb{L}$, can be clearly defined by the appropriate characteristic function $\prod$ [4],[10] which can be written as follows:

$$\prod\nolimits_{\mathbf{a},\mathbf{b}}^{(I)}(\mathbf{L})=\begin{cases}1, & \textit{when } \mathbf{a}=\mathbf{0}\wedge\mathbf{b}=\mathbf{1}\\ \prod_{i\in\mathbb{D}_1} l_i\prod_{i\in\mathbb{D}_0}\tilde{l}_i, & \textit{when } \mathbf{a}\ne\mathbf{0}\vee\mathbf{b}\ne\mathbf{1}\end{cases} \tag{9}$$

where: l_i - Boolean variables (the components of the vector of Boolean variables $\mathbf{L}$), a_i,b_i - Boolean constants (the components of vectors $\mathbf{a}$ and $\mathbf{b}$), $\mathbb{D}_1=\{i\in\mathbb{D}:a_i=b_i=1\}$, $\mathbb{D}_0=\{i\in\mathbb{D}:a_i=b_i=0\}$ $\mathbb{D}=(\mathbb{D}_1\cup\mathbb{D}_0)\subset\{1,...,i,...,I\}$. Using the characteristic function of the interval (9) one can determine whether a given vector belongs to the interval or not. Let's note that the set of vectors for which the non-decreasing Boolean function takes the value 1 is covered by (inseparable) intervals, whose lower boundaries are the minimum vectors of this function and the upper end of each of these intervals is vector **1** (in this case of the number of components equal to *I*). Hereinafter these intervals will be called intervals generated by minimum vectors. This means that when identifying the function φ the consideration of space $\mathbb{L}$ can be carried out sequentially eliminating entire intervals in which the function φ takes the value of 1.

The entire space of the vectors of the states of usability (and not usability) $\mathbb{L}$, determined by the interval $[\mathbf{a},\mathbf{b}]$ can be dichotomously divided into two intervals $[\mathbf{0},\mathbf{b}_1]$ and $[\mathbf{a}_1,\mathbf{1}]$, where $\mathbf{b}_1=\langle 1_1,...,1_i,...,0_I\rangle$ while $\mathbf{a}_1=\langle 0_1,...,0_i,...,1_I\rangle$. By dividing in an analogous way each of the intervals obtained in the next steps, after *I*-1 steps we get a binary tree [1],[2],[11], whose vertices are intervals of the form $[\mathbf{a}_k,\mathbf{b}_k]^n$, where: *k* is the step number (the level of binary tree), $k=\overline{0,I-1}$; *n* denotes the interval number for *k*th level, $n=\overline{1,2^k}$. The general rule for determining the intervals for the *k*th level of the tree on the basis of each of the intervals obtained at the $k-1$ level $([\mathbf{a}_{k-1},\mathbf{b}_{k-1}])$ at $k>0$, is as follows:

- In the vector $\mathbf{b}_{k-1}$ we insert the value of the component 0 at the position numbered *k*, obtaining the vector $\mathbf{b}_k$ (upper boundary of the first interval).
- In the vector $\mathbf{a}_{k-1}$ we insert the value of the component 1 at the position numbered *k,* obtaining the vector $\mathbf{a}_k$ (lover boundary of the second interval).
- By replacing the upper boundary of the interval $[\mathbf{a}_{k-1},\mathbf{b}_{k-1}]$ by the determined vector $\mathbf{b}_k$ we obtain the first interval $[\mathbf{a}_k,\mathbf{b}_k]^{'}$, wherein $\mathbf{a}_k=\mathbf{a}_{k-1}$.
- By replacing the lower boundary of the interval $[\mathbf{a}_{k-1},\mathbf{b}_{k-1}]$ by the determined vector $\mathbf{a}_k$ we obtain the second interval $[\mathbf{a}_k,\mathbf{b}_k]^{''}$, wherein $\mathbf{b}_k=\mathbf{b}_{k-1}$.

Determining the minimum vectors of the function φ comes down to searching for these vectors within the intervals of thus formed binary tree. Using the properties of the non-decreasing Boolean monotonic function (7) and the characteristic function of the interval (9) it can be stated that:

$$\begin{aligned}&\left[(\mathbf{a} \prec \mathbf{b}) \wedge \varphi(\mathbf{b}) = 0\right] \Rightarrow \left[\forall \mathbf{L} \in [\mathbf{a},\mathbf{b}] : \varphi(\mathbf{L}) = 0\right]\\ &\left[(\mathbf{a} \prec \mathbf{b}) \wedge \varphi(\mathbf{a}) = 1\right] \Rightarrow \left[\forall \mathbf{L} \in [\mathbf{a},\mathbf{b}] : \varphi(\mathbf{L}) = 1\right]\end{aligned} \tag{10}$$

It follows from the above that to identify the function φ it is not necessary to consider all determined intervals. If the value of the function φ is equal to 0 for the upper end of the interval, then there is no need to consider this interval. This corresponds to removing the whole subspace of vectors from the considerations (the subtree, whose node is the interval $[\mathbf{a},\mathbf{b}]$). Similarly, we remove a vector subspace if for the lower boundary of the interval φ is equal to 1. This allows us to minimize the number of necessary inquiries about the system's usability.

To search the binary tree discussed in the paper, the so-called *pre-order* strategy was applied [1],[11]. There still remains the problem of establishing minimum vectors in the currently considered interval $[\mathbf{a},\mathbf{b}]$. Remembering that this interval is the vertice of the previously formed binary tree and based on (10), it may be noted that the sub-trees of this vertice should be considered only when $\varphi(\mathbf{a}) = 0$ and $\varphi(\mathbf{b}) = 1$. Therefore, the minimum vector of the function φ in the considered interval can be only one of its boundaries.

The upper boundary $\mathbf{b}$ of the considered interval $[\mathbf{a},\mathbf{b}]$ is the minimum vector of the function φ if it does not belong to any of the subsets generated by already established minimum vectors, and this is the only vector from this interval at which the user considers the system useable. The lower boundary $\mathbf{a}$ of the considered interval $[\mathbf{a},\mathbf{b}]$ is the minimum vector of the function φ if it does not belong to any of the subsets generated by already established minimum vectors and the user considers the system to be useable for this vector. The function φ identified on the basis of the user's answer can be visualized (in a reasonable dimension) in the form of a binary cube [12].

4.2. The method of determining optimal usable substructures for a given system usability condition

The system designer setting the function f for the system being designed (the values $f(\mathbf{L},\mathbf{Y})$) must know, on the one hand, the values of the function $\varphi(\mathbf{L})$, set by the user, and on the other hand the possibility (or inability) specified in the project to provide a subset of services specified by $\mathbf{L}$ for given vector $\mathbf{Y}$ of reliability states of technical resources of the system. If $\varphi(\mathbf{L}) = 1$ and there is a possibility to ensure performing the subset of tasks determined by $\mathbf{L}$ (with the configuration chosen for $\mathbf{Y}$ – such configurations for a given $\mathbf{Y}$ may, in the general case, be numerous), then the designer should determine that $f(\mathbf{L},\mathbf{Y}) = 1$. In other cases $f(\mathbf{L},\mathbf{Y}) = 0$. Therefore, in order to determine the value of $f(\mathbf{L},\mathbf{Y})$, the designer, in addition to the function φ must, theoretically, have binary feasibility functions of the subsets of tasks [13] set for all possible vectors $\mathbf{Y}$, defined as follows:

$$g_{\mathbf{Y}} : \mathbb{L} \rightarrow \{0,1\}, \mathbf{Y} \in \mathbb{Y}, \quad g_{\mathbf{Y}}(\mathbf{L}) = \begin{cases} 1 & \textit{when, with the reliability states the elements of the} \\ & \textit{system specyfied by the vector } \mathbf{Y} \textit{ there is the possibility} \\ & \textit{of performing a subset of tasks specified by the vector } \mathbf{L} \\ 0 & \textit{otherwise} \end{cases} \tag{11}$$

where $\mathbb{L}$ i $\mathbb{Y}$ are the sets of all $\mathbf{L}$ and $\mathbf{Y}$ vectors, respectively. In view of the above it can be seen that $f(\mathbf{L},\mathbf{Y}) = \varphi(\mathbf{L}) \wedge g_{\mathbf{Y}}(\mathbf{L})$. Let:

$$\mathbb{ZF} = \{zf_1, ..., zf_j, ..., zf_J\}, \quad \mathbb{ZT} = \{zt_1, ..., zt_n, ..., zt_N\}, \quad \mathbb{ZG} = \{zg_1, ..., zg_s, ..., zg_S\} \tag{12}$$

means, respectively, a set of required functional resources, a set of technical resources (determined by the designer) and a set of technical resource groups, i.e. a set of all these subsets of technical resources ($zg_s \in \mathbb{ZG}$), which the designer has deemed appropriate to implement the determined functional resources. In the general case, such subsets are not disjoint. Therefore, if we assume that each technical resource zt_n can be simultaneously allocated to the implementation of only one functional resource, then after choosing any group of technical resources, which element is zt_n, for the implementation of this functional resource, no other group of technical resources whose element is zt_n can no longer be used to implement any other functional resource. Connections of functional resources with possible groups of technical resources, ensuring their disjunctive implementation, it is convenient to present in the form of a graph:

$$GT = \langle \mathbb{XT}, \mathbb{UT}, PT \rangle \tag{13}$$

where: $\mathbb{XT}$ – set of graph vertices, $\mathbb{XT} = \mathbb{ZF} \cup \mathbb{ZG}$; $\mathbb{UT}$ – set of graph edges, $\mathbb{UT} = \overrightarrow{\mathbb{UT}} \cup \mathbb{UT}$ wherein $\overrightarrow{\mathbb{UT}} \subset \mathbb{ZF} \times \mathbb{ZG}$, $\mathbb{UT} \subset \mathbb{ZG} \times \mathbb{ZG}$; PT – a ternary relationship meeting the following conditions:

$$\begin{gathered} a)\ \forall u \in \mathbb{UT}\ \ \exists x, y \in \mathbb{XT} : \langle x, y, u \rangle \in PT \\ g_{\mathbf{Y}}\ b)\ \forall u \in \mathbb{UT}\ \ \forall x, y, z, v \in \mathbb{XT} : \\ \{[\langle x, y, u \rangle \in PT \wedge \langle v, z, u \rangle \in PT] \Rightarrow [(x = v) \wedge (y = z) \vee (x = z) \wedge (y = v)]\} \end{gathered} \tag{14}$$

Graph GT will be referred to as the graph of full technical capabilities of functional resources implementation. The edge connecting two groups of technical resources means that both groups have at least one common element (but it is not a group of identical technical resources). The lack of edges between groups means separation of these groups. The graph of the full technical capabilities of implementing functional resources GT will be used to determine the value of $g_{\mathbf{Y}}(\mathbf{L})$ function with the determined reliability status of technical resources described by the vector **L**. Let's denote by $\mathbb{ZF}_{\mathbf{L}}$ the set of functional resources necessary to perform the services described by the vector **L**, $\mathbb{ZF}_{\mathbf{L}} \subset \mathbb{ZF}$. Let $\mathbb{ZG}_{\mathbf{Y}} \subset \mathbb{ZG}$ be the set of those groups of technical resources for which all their elements are useable with this **Y** vector. Thus, with the current state of technical resources, each group $zg \in \mathbb{ZG}_{\mathbf{Y}}$ consists only of those technical resources for which the corresponding component in the vector **Y** is of value 1. If we now remove from the GT graph all vertices that do not belong either to the subset $\mathbb{ZF}_{\mathbf{L}}$ or to the subset $\mathbb{ZG}_{\mathbf{Y}}$ then we will obtain the $GT_{\mathbf{Y}}^{\mathbf{L}}$ graph (GT subgraph), which will determine the possibilities of technical feasibility of functional resources necessary to carry out the tasks described by the vector **L**, with a fixed state of technical resources, described by the vector **Y**. To determine the function $g_{\mathbf{Y}}(\mathbf{L})$ it is sufficient to determine the most numerous association of the graph $GT_{\mathbf{Y}}^{\mathbf{L}}$. With the user-defined function φ determining the value of the function $g_{\mathbf{Y}}$ for each of the **Y** vectors is equivalent to determining the function f.

From the essence of the function $g_{\mathbf{Y}}$ (11) it follows that it is a non-growing monotonic binary (Boolean) function and is a coherent one. Thus the function $g_{\mathbf{Y}}$ can be uniquely determined by the set of its maximum vectors. To determine the maximum vectors of the $g_{\mathbf{Y}}$ function the identification method of the function φ, can be successfully used, after its appropriate modification. To this end, let us introduce several symbols and dependencies. As the negated vector of the vector **L** (1) we call the vector $\tilde{\mathbf{L}} = \langle \tilde{l}_1, ..., \tilde{l}_i, ..., \tilde{l}_I \rangle$ where the sign $\sqcup$ means Boolean negation. The following relationships can be proved to be true:

1. For each pair of vectors $\mathbf{L}_i, \mathbf{L}_j \in \mathbb{L}$ the relationship $[\mathbf{L}_i \prec \mathbf{L}_j] \Leftrightarrow [\tilde{\mathbf{L}}_j \prec \tilde{\mathbf{L}}_i]$ is true.
2. Let the Boolean function $g: \mathbb{L} \rightarrow \{0,1\}$ be a non-growing monotonic function. Then the function $g': \mathbb{L} \rightarrow \{0,1\}$, whose values are determined by the equality $g'(\mathbf{L}) = g(\tilde{\mathbf{L}})$ is a non-decreasing monotonic function.
3. Let g denote a non-growing monotonic Boolean function, and the function g' defined by the equation $g'(\mathbf{L}) = g(\tilde{\mathbf{L}})$ will be, by relationship 2, a Boolean non-decreasing monotonic function. Then the negated minimum vectors of the non-decreasing monotonic Boolean function g' are the maximum vectors of the non-growing monotonic Boolean function g.
4. Each maximum vector of the function $f(\mathbf{L}, \mathbf{Y}) \equiv f_{\mathbf{Y}}(\mathbf{L})$ is the maximum vector of the function $g_{\mathbf{Y}}(\mathbf{L})$, i.e. $\mathbf{L} \in \mathbb{L}_{\max}^{f_{\mathbf{Y}}} \Rightarrow \mathbf{L} \in \mathbb{L}_{\max}^{g_{\mathbf{Y}}}$.

Therefore, based on the relationships 2 and 3 as well as using the method of determining the minimum vectors of the function φ, we can determine the maximum vectors for each considered function $g_{\mathbf{Y}}(\mathbf{L})$. Based on the relationship 4, it can be observed that the set of maximum vectors $f_{\mathbf{Y}}(\mathbf{L})$ is a subset of the set of maximum vectors of the function $g_{\mathbf{Y}}(\mathbf{L})$, for which $\varphi(\mathbf{L})=1$. This subset is a set of admissible solutions $\Omega_{\mathbf{Y}}$ (permissible usable structures) with the reliability state of the technical resources of the system described by the **Y** vector. Therefore, it is enough to select from the set $\Omega_{\mathbf{Y}}$ such a vector $\mathbf{L}^*$ (e.g. using the genetic algorithm [14]) for which the objective function *W* adopts the smallest value. It should be added that the above operations will be carried out without the intervention of the designer, because the answers to questions about the usability of the system will be generated automatically (presentation of the algorithm for checking the possibility of implementing a subset of tasks based on the $GT_{\mathbf{Y}}^{\mathbf{L}}$ graph exceeds the scope of this work). Particular $GT_{\mathbf{Y}}^{\mathbf{L}}$ graphs will also be created automatically based on the specified, for each functional resource, possible groups of technical resources, ensuring its disjunctive implementation.

5. Conclusions

The method presented in this paper allows to determine, at the design stage, such computer system configurations that will ensure the best, from the user's point of view, system performance in the event of failure of its technical resources. By assigning the best system configuration to each distinguished emergency situation, obtainable at a given state of technical resources, we get an automatic reconfiguration table (we abstract here from its physical implementation), based on which a properly programmed mechanism will auto-reconfigure the system. The development of such a table at the design stage of the system reduces the time of elaboration a new system configuration during its operation, almost to zero. This is particularly important in highly time-dependent systems where the response time to emergencies can have a decisive impact on the further usability of the system.

The proposed method, according to the authors' opinion, can be applied not only in relation to computer systems. It can be used to solve a number of problems from the class, let's call it, "tasks-resources" where specific resources are required to perform tasks, and which of these tasks can be performed depends on whether and which resources will be allocated. For example, this may be the issue of selecting projects for their portfolio. It can also be a matter of building a state security system, but as an objective function certainly it will be necessary to take into account the costs of performing system reconfiguration.

References

[1] Cabinet Office, ITIL 2011 Service Design, The Stationery Office (2011).
[2] Cabinet Office, ITIL 2011 Service Operation, The Stationery Office (2011).
[3] K. Bausea, A. Radimerskya, M. Iwanickia, A. Albersa, Feasibility Studies in the Product Development Process, Procedia, CIRP 21 (2014), 473 – 478.
[4] B. Korzan, Elementy teorii niezawodności, WAT, (1986).
[5] H Pham, Handbook of reliability engineering, Springer, (2003).
[6] O. Coudert, Two-level logic minimization: an overview, INTEGRATION 17-2 (1994), 97–140, http://www.ocoudert.com/papers/pdf/int94.pdf (access 12.08.2019).
[7] T. Łuba, Synteza układów logicznych, Oficyna Wydawnicza Politechniki Warszawskiej (2005).
[8] R. L. Rudell, Logic Synthesis for VLSI Design, University of California at Berkeley (1989), http://www1.cs.columbia.edu/~cs6861/handouts/rudell-PhDthesis.pdf (access 12.08.2019).
[9] A. Kapica, S. Rozmus, Metoda operatywnej konfiguracji komputerowego system zautomatyzowanego dowodzenia, Postępy Cybernetyki, 3 (1993).
[10] J. Karpiński, E. Korczak, Metody oceny niezawodności dwustanowych systemów technicznych, IBS PAN (1993).
[11] N. Wirth, Algorytmy + struktury danych = program, WNT (2004).
[12] B. Korzan, Elementy teorii grafów i sieci. Metody i zastosowania, WNT (1978).
[13] B. Korzan, S. Rozmus, Metoda ustalania uogólnionej strukturalnej funkcji niezawodnościowej projektowanego stałoprogramowego system komputerowego, Biuletyn WAT, 6 (1993), 15-24.
[14] J. Arabas, Wykłady z algorytmów ewolucyjnych, WNT (2016).