MATEC Web of Conferences **76**, 04009 (2016) DOI: 10.1051/matecconf/20167604009
CSCC 2016

Method for assessing software reliability of the document management system using the RFID technology

Maciej Kiedrowicz[1], Tadeusz Nowicki[1], Robert Waszkowski[1], Zbigniew Wesołowski[1], Kazimierz Worwa[1,a]

[1]*The Faculty of Cybernetics, Military University of Technology, 00-908 Warsaw, Kaliskiego 2, Poland*

Abstract. The deliberations presented in this study refer to the method for assessing software reliability of the document management system, using the RFID technology. A method for determining the reliability structure of the discussed software, understood as the index vector for assessing reliability of its components, was proposed. The model of the analyzed software is the control transfer graph, in which the probability of activating individual components during the system's operation results from the so-called operational profile, which characterizes the actual working environment. The reliability structure is established as a result of the solution of a specific mathematical software task. The knowledge of the reliability structure of the software makes it possible to properly plan the time and financial expenses necessary to build the software, which would meet the reliability requirements. The application of the presented method is illustrated by the number example, corresponding to the software reality of the RFID document management system.

1 Introduction

The software reliability is one of the most important indicators of the software quality. One of the methods for modeling software reliability is to treat the software as monolithic entirety, while considering its interactions with the external environment, without trying to model the internal structure of the software. Such approach is inappropriate for modeling larger IT systems, with complex internal architecture. There is a demand for the modeling techniques of the software reliability, using the knowledge of its internal structure, e.g. module structure. The basic unit for determining the software structure is its component - often a module - understood as a logically independent software element, which performs well a specific function. It means that the modules may be designed, implemented and examined independently.

The behavior of the software, depending on the manner, in which different software modules affect each other, is defined by the software architecture. The interactions between different modules in the form of mutual control transfer are present only during the execution of the computer program. The software architecture may also contain some information on frequency of activation of the individual modules. When the software is working, some emergency situations related to software errors may occur. The software error may be connected with the implementation of any module or transfer of control between the two modules. A possibility of occurrence of some emergency situations during the use of the software is described by various software reliability indicators. The knowledge of the component (module) structure of the examined software allows using the indicators, whose values depend on such structure and the reliability indicators of its particular elements. Depending on the method applied in that respect, it is possible to distinguish two approaches to the reliability modeling process, on the basis of: the structure set by the source code instructions, structure based on logical paths [6].

The models based on the architecture set by the source code for modeling the software architecture use the so-called control transfer graph, illustrating a possibility of transferring control between the blocks of the source code. The nodes on the structure are indicated by the so-called program control guidelines. It is often assumed in the models of the discussed class that the control transfer between them may be described by the Markov chains. This assumption means that the control transfer after implementation of a specific module to other modules does not depend on previous activation of the program blocks. The software architecture is modeled by the Markov chain, e.g. discrete in states and in time. The representative examples of such models may be found in studies [1, 5, 9, 11, 13].

In the models based on the logical path concept, the software architecture is also created by separate blocks of the source code, e.g. by the modules, which are not - however - analyzed as independent components, but as sequences of components executed one by one in case of activation of a given path by an appropriate set of input data. Knowing the values of the reliability indicators of the components that create a given logical path, it is possible to determine the value of the reliability indicator

[a] Corresponding author: kazimierz.worwa@wat.edu.pl

of the whole software, using the knowledge of the architecture of the software. The representative examples of the models based on logical paths may be found in studies [8, 17, 23].

It is worth mentioning that apart from the above-mentioned methods for modeling the software reliability, models alternative to the models based on the software architecture and results of the software errors observed, e.g. during the software tests, are applied in practice. On such basis, the hypotheses concerning working time distribution between the moments of the occurrence of the two subsequent errors are formulated. The representative examples of such models may be found in studies [4, 10, 12, 14, 20, 22, 24].

On the basis of the software engineering practice, it is evident that the software quality is the function of the time and expenses incurred during the production process, regardless of the applied methodology and production technology. Due to the fact that the process is usually executed with limited budget, the available funds should be used in a manner that maximally improves the quality of the produced software. A method often used in practice to achieve this objective is identification of the key components - for the purpose of proper functioning of the entire software - and paying special attention to the process of their production. It is worth mentioning that the solution of the issue of reducing the time and expenses required in the software production process may be significantly facilitated by applying formal methods of mathematical modeling and operational research. The examples of such approach may be found, for example, in [3-4,6-7,15, 21].

The subject of the deliberations in this study is the allocation of funds for the production of the document management software, using the RFID technology, with a known component structure, assuming that the software production process covers independent production of its components. In further deliberations, the components of the discussed software will be called software modules or just modules. In accordance with the previous idea, the allocation of the aforesaid funds includes the impact of the particular modules on the correct functioning of the whole software. The model of the discussed software is the control transfer graph, in which the probability of activating particular component modules during the operation of the system results from the so-called operation profile of the software, characterizing its actual working environment. Identification of the modules, whose execution has the largest impact on the proper functioning of the entire software system, will be made on the basis of an analysis of the so-called software reliability structure, understood as the components. The optimum structure will be established as a result of the solution of a specific mathematical software task. The knowledge of the software reliability structure of the document management system, with the RFID technology, allows proper planning of time and expenses required for its production. It should be emphasized that the approaches found in the literature [1], for example to determine the meaning of particular component modules of the software, are based on the results of the so-called sensitivity analysis, i.e. the analysis of the impact of changing the values of the reliability indicators of particular modules on the value of the adopted indicator of the software quality.

2 Characteristics of the examined software

It is assumed that the examined software has a module structure, where the term "module" has the meaning generally given in the software engineering, i.e. a logically independent software component, which usually executes one function. The logical independence means, among other things, technological independence, i.e. the module may be designed, implemented and tested independently. The typical production process of the software with the above-described module structure consists in identification and separation of the component modules, independent performance of the individual modules and their subsequent integration (usually during the integration testing) into one.

For the purpose of this study, the analyzed software will be represented by the graph, with a set of vertices corresponding to the numbers of the selected component modules and a set of arcs, characterizing a possibility of transferring control between the modules during the execution of the software. Without compromising the deliberations herein, it is possible to assume that the discussed software has one input and one output module. In the literature, the graph with the provided values is called the control transfer graph of the software [1, 16].

Let $\boldsymbol{M} = \{1,2,3,...,m,...,M\}$ mean a set of the module numbers of the examined software, where - for the purposes of simplification - the input module has number *1*, and the output module – number *M*.

The reliability structure of the examined software will be understood as vector $\boldsymbol{R} = (R_1, R_2, R_3, ..., R_m, ..., R_M)$, whose particular components are the reliability indicators of the component modules, where the reliability indicator of the module constitutes probability of its correct, single execution during the execution of the software. The execution of the software for the specific set of its input data consists in activating a certain, corresponding sequence of modules, which created the so-called logical path of the software. It is assumed that the measure of reliability of the examined software is probability of its proper execution for a single set of input data. The probability is equal to a product of probabilities of the proper execution of the modules, which are part of the path from the initial to the final module, activated by a given set of input data. The reliability of the software module is a function of many factors, such as qualifications and experience of its designers, applied production technology, amount of time and expenses incurred, etc. It is assumed in the study that the process of transferring control between the modules during the software execution is the Markov chain. In particular, it means that the probability of the p_{ij} event consisting in the fact that after the execution of the*i*-th module, the *j*-th module will be next, does not depend on what happened before the control transfer to the*i*th module. The probability of transfers p_{ij},

$i, j \in \boldsymbol{M}$ between the software component modules in the process of its execution results from the so-called operational profile of the software, characterizing its actual working environment. When adding to the software control graph two additional statuses corresponding to the proper and improper completion of the software execution, respectively, the Markov chain is obtained with two absorbing statuses, whereas - in accordance with the previous assumptions - the reliability indicator of the software, defined as the probability of its proper execution for a single set of input data, may be calculated as the probability of achieving by the discussed Marvov process the absorbing status corresponding to the proper and improper completion of the software execution. The examined Markov chain explicitly characterizes the transfer matrix, which - in accordance with the adopted assumptions - is of the form

$$\overline{\boldsymbol{P}}(\boldsymbol{R}) = \begin{bmatrix} & C & F & 1 & 2 & \dots & m & \dots & M \\ C & 1 & 0 & 0 & 0 & \dots & 0 & \dots & 0 \\ F & 0 & 1 & 0 & 0 & \dots & 0 & \dots & 0 \\ 1 & 0 & 1-R_1 & 0 & R_1 p_{12} & \dots & R_1 p_{1m} & \dots & R_1 p_{1M} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ m & 0 & 1-R_m & 0 & R_m p_{m2} & \dots & R_m p_{mm} & \dots & R_m p_{mM} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ M-1 & 0 & 1-R_{M-1} & 0 & R_{M-1} p_{M-1,2} & \dots & R_{M-1} p_{M-1,m} & \dots & R_{M-1} p_{M-1,M} \\ M & R_M & 1-R_{M-1} & 0 & 0 & \dots & 0 & \dots & 0 \end{bmatrix}$$

in which the first two lines and the first columns correspond to the proper and improper completion of the software execution, respectively, for a single set of input data.

Let ***P(R)*** mean the matrix obtained by rejecting from matrix $\overline{\boldsymbol{P}}(\boldsymbol{R})$ the first two lines and the first two columns. It may be shown that the reliability indicator of the software ***R(R)*** with reliability structure ***R***, understood as the probability of its proper execution for a single set of input data, is defined by formula [6]

$$R(\boldsymbol{R}) = S_{1M}(\boldsymbol{R}) R_M , \quad (1)$$

where S_{1M} is an element located at the intersection of the first line and the *M*-th column of matrix *S(R),* defined in the following manner:

$$\boldsymbol{S}(\boldsymbol{R}) = \boldsymbol{I} + \boldsymbol{P}(\boldsymbol{R}) + \boldsymbol{P}^2(\boldsymbol{R}) + \dots = \sum_{k=o}^{\infty} \boldsymbol{P}^k(\boldsymbol{R}) , \quad (2)$$

where ***I*** is a unit matrix of the same category as matrix ***P(R).***

3 Optimization of the software reliability structure

Let K_m mean the cost of production of the *m*-th software component module, with the reliability indicator of R_m, $m \in \boldsymbol{M}$.

When assuming that the production costs of the particular modules are the known functions of their reliability indicators $K_m = f_m(R_m)$, $m \in \boldsymbol{M}$, it is possible to determine - by way of an appropriately formulated mathematical programming - the software reliability structure, which may be understood as a kind of the production plan, and form the practical point of view, especially two tasks might be useful:

- one-criterion optimization task consisting in maximizing the value of the software reliability indicator as the function of objective, with the maximum, allowed cost of the software production being the limitation;
- two-criteria optimization task, with the software reliability indicator and the cost of its production as the vector function of objective, ensuring determination of the allowed software reliability structure, i.e. ensuring achievement of the required, minimum reliability level and not exceeding of the maximum, allowed cost of the software production.

The knowledge of the optimum software reliability structure, at the stage of planning its production, allows to identify the modules of the greatest importance for the reliability of the whole software and hence allows to allocate the available funds in such a manner that the produced software fully meets the assumed reliability requirements.

When considering the introduced designations and assumptions made, the first of the aforementioned optimization tasks may be formulated in the following manner:

$$\text{maximize} \quad R(\boldsymbol{R}) = S_{1M}(\boldsymbol{R}) R_M \quad (3)$$

considering limitation

$$K(\boldsymbol{R}) = \sum_{m=1}^{M} K_m(R_m) \le K_{max} , \quad (4)$$

where

$$\boldsymbol{R} = (R_1, R_2, R_3, \dots, R_m, \dots, R_M) , \; 0 \le R_m \le 1 , \; m \in \boldsymbol{M} .$$

The formulation of the second of the aforementioned optimization tasks may be in the following form:

$$(\mathbf{R}, F, \Psi), \quad (5)$$

where:

R - set of allowed solutions, defined in the following manner:

$$\mathbf{R} = \{ \boldsymbol{R} = (R_1, R_2, R_3, \dots, R_m, \dots, R_M) , \; 0 \le R_m \le 1 , \; m \in \boldsymbol{M} \},$$

F - solution quality indicator:

$$F(\boldsymbol{R}) = (K(\boldsymbol{R}), R(\boldsymbol{R})) , \quad (6)$$

where values***R(R)***, ***K(R)*** are defined in the following manner:

$$R(\boldsymbol{R}) = S_{1M}(\boldsymbol{R}) R_M , \quad K(\boldsymbol{R}) = \sum_{m=1}^{M} K_m(R_m) \quad (7)$$

Ψ - the dominance relationship is as follows:

$$\Psi = \{ y_1, y_2) \in \boldsymbol{Y} \times \boldsymbol{Y} : \; y_1^1 \le y_2^1, \; y_1^2 \ge y_2^2 \} , \quad (8)$$

where $y_1 = (y_1^1, y_1^2), \; y_2 = (y_2^1, y_2^2)$,

where***Y*** is the so-called criterial space defined in the following manner:

$$\boldsymbol{Y} = \{ y = (K(\boldsymbol{R}), R(\boldsymbol{R})) : \quad \boldsymbol{R} \in \mathbf{R} \} . \quad (9)$$

The first of the formulated tasks is the one-criterion optimization. Its nature, and hence - the manner of solution - strongly depend on the class (analytical form) of the

software reliability indicator $R(\boldsymbol{R})$ and the cost of its production $K(\boldsymbol{R})$. The second of the formulated tasks is the non-linear two-criteria optimization task. Its solution may be determined in accordance with the generally adopted methodology for solving the optimization tasks [18]. In accordance with this methodology, the solution of task (5)-(9) may, in particular, require the determination of the following data:

- set of dominating strategies,
- set of non-dominated strategies,
- set of compromise strategies.

Due to the nature of correlations (7), which define the component critertial functions $R(\boldsymbol{R})$ and $K(\boldsymbol{R})$ the subject task, it should be expected that the set of the dominant solutions would be empty. Therefore, in the discussed situation, the issue of determining the set of non-dominated solutions and its potential narrowing, e.g. by setting the set of compromise solution, becomes of practical importance [18]. The reliability structure $\boldsymbol{R}^*$, being the solution to task (3) - (7), additionally maximizes the value of the software reliability indicator $R(\boldsymbol{R})$ and minimizes the value of the function of the cost of $K(\boldsymbol{R})$ its production.

4 Case study - software reliability of the document management system, using the RFID technology

The presented method for determining the software reliability structure will be illustrated by a number example, which is in accordance with the reality of the document management system software, using the RFID technology. The control transfer graph of the discussed software will be defined as in *Fig. 1*.

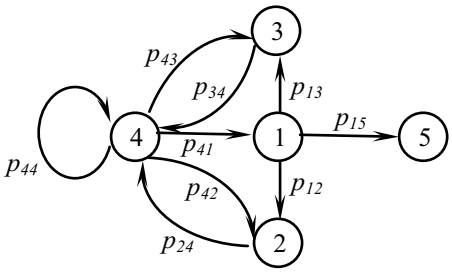

Figure 1. Control transfer graph of the discussed software

According to Fig. 1, it is evident that the set of numbers of the modules of the discussed software is in the form of $\boldsymbol{M}=\{1,2,3,4,5\}$, where module 1 is the initial module, and module 5 - the final module. The module structure of the software is defined by vector $\boldsymbol{R}=(R_1,R_2,R_3,R_4,R_5)$. Let probabilities p_{ij} of the control transfer between the modules, during the execution of the software be in the form of:

$p_{11}=0,\ p_{12}=0.6,\ p_{13}=0.3,\ p_{14}=0,\ p_{15}=0.1,$
$p_{21}=0,\ p_{22}=0,\ p_{23}=0,\ p_{24}=1,\ p_{25}=0,$
$p_{31}=0,\ p_{32}=0,\ p_{33}=0,\ p_{34}=1,\ p_{35}=0,$
$p_{41}=0.3,\ p_{42}=0.2,\ p_{43}=0.4,\ p_{44}=0.1,\ p_{45}=0.$

Using correlation (1), the software reliability indicator with the reliability structure$\boldsymbol{R}$ may be defined based on formula $R(\boldsymbol{R})=S_{15}(\boldsymbol{R})R_5$, where value S_{15} may be set on the basis of correlation (2) or programming package *Mathematica* [19], which allows performing symbolic calculations.

The use of the above-mentioned package for the module structure from *Fig.1* and the above-mentioned probabilities of transfers p_{ij} provides the following dependency on probability S_{15}

$$S_{15}=\frac{0{,}1R_1-0{,}01R_1R_4-0{,}02R_1R_2R_4-0{,}04R_1R_3R_4}{1-0{,}1R_4-0{,}2R_2R_4-0{,}18R_1R_2R_4-0{,}4R_3R_4-0{,}09R_1R_3R_4} \quad (10)$$

For the purpose of formulating the previously discussed optimization tasks, it is assumed that cost $K_i(R_i)$ of the production of the *i*-th module with the reliability indicator R_i is in the form of

$$K_i(R_i)=KS_i+\alpha_i e^{\beta_i R_i},\quad i\in\boldsymbol{M}=\{1,2,3,4,5\} \quad (11)$$

where:

KS_i – fixed component of the production cost of the *i*-th module, independent of the level of its quality R_i,

α_i, β_i – co-efficient of the curve shape of the production cost of the *i*-th module, depending i.a. on its complexity, used technology, qualifications of the design and programming team, etc.

Let figures of value KS_i, α_i i β_i and K_{max} be defined as in *Table 1*.

For the module structure of the software in *Fig. 1* and numerical data in *Table 1*, the solutions of the optimization tasks, formulated in the previous sub-chapter, i.e. the one-criterion optimization tasks (3) - (4) and polyoptimization tasks (5) - (9), will be determined.

Table 1. Figures of value KS_i, α_i, β_i, K_{max} used for calculations

i	*1*	*2*	*3*	*4*	*5*
KS_i	1000.0	400.0	1500.0	2500.0	900.0
α_i	50.0	35.0	25.0	30.0	40.0
β_i	5.0	2.0	8.0	10.0	4.0
Kmax	160 000.0				

When using the possibilities created by modern computer technology, both tasks may be solved by the full review method. For that purpose, a set of all possible reliability structures will be created $\boldsymbol{R}=(R_1,R_2,R_3,R_4,R_5)$, for $R_i\in[0.95,1.00]$, $i\in\{1,2,3,4,5\}$, to two decimal places. The number of vectors in this set is $6^5=7776$, out of which *7770* meet the restriction (4).

Table 2. Solution of the one-criterion optimization task

R_1	R_2	R_3	R_4	R_5	$\boldsymbol{R(R)}$	K_1	K_2	K_3	K_4	K_5	$K(\boldsymbol{R})$
0.98	1.00	1.00	1.00	1.00	0.8305	7,714.49	658.62	76023.95	76023.95	3,083.93	163,504.93 €
1.00	1.00	0.99	1.00	0.96	0.8362	8,420.66	658.62	70,294.28	76,023.95	2,761.02	158,158.52 €
1.00	1.00	0.99	1.00	0.97	0.8449	8,420.66	658.62	70,294.28	76,023.95	2,836.97	158,234.47 €
1.00	0.99	1.00	1.00	0.95	0.8488	8,420.66	653.50	76,023.95	76,023.95	2,688.05	163,810.10 €
1.00	1.00	0.99	1.00	0.98	0.8536	8,420.66	658.62	70,294.28	76,023.95	2,916.02	158,313.52 €
1.00	0.99	1.00	1.00	0.96	0.8578	8,420.66	653.50	76,023.95	76,023.95	2,761.02	163,883.07 €
1.00	1.00	0.99	1.00	0.99	0.8623	8,420.66	658.62	70,294.28	76,023.95	2,998.29	158,395.79 €
0.99	1.00	1.00	1.00	0.95	0.8628	8,058.75	658.62	76,023.95	76,023.95	2,688.05	163,453.31 €
1.00	0.99	1.00	1.00	0.97	0.8667	8,420.66	653.50	76,023.95	76,023.95	2,836.97	163,959.02 €
1.00	**1.00**	**0.99**	**1.00**	**1.00**	**0.8711**	**8,420.66**	**658.62**	**70,294.28**	**76,023.95**	**3,083.93**	**158,481.43 €**
0.99	1.00	1.00	1.00	0.96	0.8719	8,058.75	658.62	76,023.95	76,023.95	2,761.02	163,526.28 €
1.00	0.99	1.00	1.00	0.98	0.8756	8,420.66	653.50	76,023.95	76,023.95	2,916.02	164,038.07 €
0.99	1.00	1.00	1.00	0.97	0.8810	8,058.75	658.62	76,023.95	76,023.95	2,836.97	163,602.23 €
1.00	0.99	1.00	1.00	0.99	0.8846	8,420.66	653.50	76,023.95	76,023.95	2,998.29	164,120.35 €
0.99	1.00	1.00	1.00	0.98	0.8901	8,058.75	658.62	76,023.95	76,023.95	2,916.02	163,681.28 €
1.00	0.99	1.00	1.00	1.00	0.8935	8,420.66	653.50	76,023.95	76,023.95	3,083.93	164,205.98 €
0.99	1.00	1.00	1.00	0.99	0.8992	8,058.75	658.62	76,023.95	76,023.95	2,998.29	163,763.56 €
0.99	1.00	1.00	1.00	1.00	0.9083	8,058.75	658.62	76,023.95	76,023.95	3,083.93	163,849.19 €
1.00	1.00	1.00	1.00	0.95	0.9500	8,420.66	658.62	76,023.95	76,023.95	2,688.05	163,815.22 €
1.00	1.00	1.00	1.00	0.96	0.9600	8,420.66	658.62	76,023.95	76,023.95	2,761.02	163,888.19 €
1.00	1.00	1.00	1.00	0.97	0.9700	8,420.66	658.62	76,023.95	76,023.95	2,836.97	163,964.14 €
1.00	1.00	1.00	1.00	0.98	0.9800	8,420.66	658.62	76,023.95	76,023.95	2,916.02	164,043.19 €
1.00	1.00	1.00	1.00	0.99	0.9900	8,420.66	658.62	76,023.95	76,023.95	2,998.29	164,125.47 €
1.00	1.00	1.00	1.00	1.00	1.0000	8,420.66	658.62	76,023.95	76,023.95	3,083.93	164,211.10 €

Table 2 shows the values of the software reliability indicator (1) and the production cost of the model software, indicated in accordance with dependency $K(\boldsymbol{R}) = \sum_{i=1}^{5} K_i(R_i)$, where the cost of $K_i(R_i)$ is defined in formula (11), for the most important (from the point of view of the solution sought) fragment of the above set of the reliability structures of the analyzed software. The lines corresponding to the forbidden solutions, i.e. vectors $\boldsymbol{R} = (R_1, R_2, R_3, R_4, R_5)$, in case of which restriction (4) is not met, are marked in lighter color. Vector $\boldsymbol{R}^* = (1.00, 1.00, 0.99, 1.00, 1.00)$ is the optimum solution of the one-criterion optimization task (3) - (4). The corresponding line in *table 1* was shaded. The value of the software reliability indicator corresponding to the optimum solution is $R(\boldsymbol{R}^*)=0.87$, with the production cost of the software $K(\boldsymbol{R}^*)=158\ 481.43$ €.

For the purpose of the analysis to obtain optimum solution $\boldsymbol{R}^*$, the sensitivity evaluation of the software reliability indicator was performed: $R(\boldsymbol{R})$ with respect to a single change of the value of the reliability indicator of particular modules for the two levels of values of indicator $R(\boldsymbol{R})$. The results are presented in *Table 3*. It is noticeable that the ratio of the growth value of indicator $R(\boldsymbol{R})$ to the cost of its achievement is the least beneficial for module number *3*. It means that increasing the value of indicator R_3 is the most costly. It is worth mentioning that this fact is reflected in the value structure of optimum solution $\boldsymbol{R}^*$.

Due to the fact that the component criterial functions present in the two-criteria optimization task (5)-(9) are of opposite nature (the cost of the software production is reduced, while the value of the software reliability indicator is maximized), the set of dominating solutions to this task is empty. Therefore, the solutions must be sought among the non-dominated solutions.

Table 3. The unitary increase in the value of the software reliability indicator and the cost of its production for ΔR_i=0,01 and: a) R_i=0,95; b) R_i=0,99

	i	*1*	*2*	*3*	*4*	*5*
a)	$\Delta R(\Delta R_i)$	0.0073	0.0055	0.0063	0.0136	0.0026
	$\Delta K(\Delta R_i)$	296.31 €	4.73 €	4.160,60 €	4.160,60 €	72.97 €
b)	$\Delta R(\Delta R_i)$	0.0394	0.0436	0.0548	0.1230	0.0061
	$\Delta K(\Delta R_i)$	361.91 €	5.12 €	5.729.67 €	5.729.67 €	85.63 €

The set of the non-dominated solutions to the polyoptimization task (5)-(9) is presented in *Fig. 2*. Every point

in bold fragment of the contour of *Fig. 2* is a non-dominated solution.

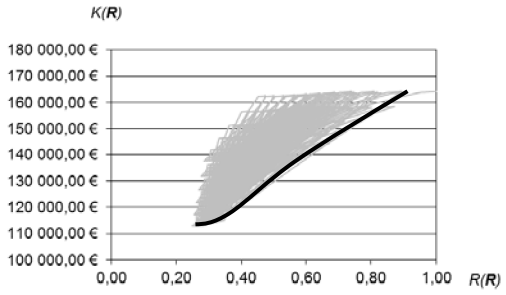

Figure 2. The non-dominated solution set of the optimization task (5)-(9)

To narrow the set to one solution only, the compromise solution, which is the closest (in terms of the Euclidean space) to the so-called ideal point [18] will be indicated. For that purpose, both components of the vector criterial function $(K(\boldsymbol{R}), R(\boldsymbol{R}))$ will be standardized, in result of which both indicators will have values in the range of *[0, 1]:*

$$\overline{K(\boldsymbol{R})} = \frac{K(\boldsymbol{R}) - K_{min}(\boldsymbol{R})}{K_{max}(\boldsymbol{R}) - K_{min}(\boldsymbol{R})},$$

$$\overline{R(\boldsymbol{R})} = \frac{R(\boldsymbol{R}) - R_{min}(\boldsymbol{R})}{R_{max}(\boldsymbol{R}) - R_{min}(\boldsymbol{R})}, \qquad (12)$$

where

$$K_{min}(\boldsymbol{R}) = \min_{\boldsymbol{R}\in\Re} K(\boldsymbol{R}), \quad R_{min}(\boldsymbol{R}) = \min_{\boldsymbol{R}\in\Re} R(\boldsymbol{R}) \qquad (13)$$

$$K_{max}(\boldsymbol{R}) = \max_{\boldsymbol{R}\in\Re} K(\boldsymbol{R}), \quad R_{max}(\boldsymbol{R}) = \max_{\boldsymbol{R}\in\Re} R(\boldsymbol{R}). \qquad (14)$$

In terms of the discussed task, the ideal point $(\overline{K^*(\boldsymbol{R})}, \overline{R^*(\boldsymbol{R})})$, being an element of the so-called extended standardized criterial space, is in the form of $(\overline{K^*(\boldsymbol{R})}, \overline{R^*(\boldsymbol{R})}) = (0,1)$.

The solution of the polyoptimization problem (5)–(9), with cost limitation $K(\boldsymbol{R}) \le 160\,000$ € is presented in *Table 3.* The shaded line corresponds to vector $\boldsymbol{R}^* = (1.00, 1.00, 0.95, 1.00, 1.00)$, being the compromise solution, i.e. the closest to the ideal point *(0, 1)*, where distance $d[(\overline{K(\boldsymbol{R})}, \overline{R(\boldsymbol{R})}), (0,1)]$ is defined by the following formula

$$d[(\overline{K(\boldsymbol{R})}, \overline{R(\boldsymbol{R})}), (0,1)] = \sqrt{[\overline{K(\boldsymbol{R})}]^2 + [\overline{R(\boldsymbol{R})} - 1]^2}. \qquad (15)$$

The following

$$\boldsymbol{R}^* = (1.00, 1.00, 0.95, 1.00, 1.00)$$

values of the component criterial functions correspond to the compromise solution: $K(\boldsymbol{R}^*) = 139\,42.05$ € and $R(\boldsymbol{R}^*) = 0.5872$.

Table 4 shows the solution of the analyzed polyoptimization problem, with additional limitation to the minimum, allowed level of the software reliability in the form of $R(\boldsymbol{R}) \ge 0{,}8$. In this case, the compromise solution is vector $\boldsymbol{R}^* = (1.00, 1.00, 0.99, 1.00, 1.00)$, with the values of the component criterial functions: $K(\boldsymbol{R}^*) = 158\,481.43$ € and $R(\boldsymbol{R}^*) = 0.8711$. It is apparent that the solution coincides with the solution of the one-criterion optimization problem (3) - (4).

Table 4. Solution of the two-criteria optimization task K(**R**) ≤ 160 000 €

R_1	R_2	R_3	R_4	R_5	$\overline{R(\boldsymbol{R})}$	$\overline{K(\boldsymbol{R})}$	$d[(\overline{K(\boldsymbol{R})}, \overline{R(\boldsymbol{R})}), (0,1)]$
1.00	1.00	0.95	0.99	0.99	0.3425	0.41	0.773068
1.00	1.00	0.95	0.99	1.00	0.3493	0.41	0.768180
1.00	1.00	0.95	1.00	0.95	0.4116	0.51	0.780266
1.00	1.00	0.95	1.00	0.96	0.4194	0.51	0.775334
1.00	1.00	0.95	1.00	0.97	0.4272	0.52	0.770492
1.00	1.00	0.95	1.00	0.98	0.4350	0.52	0.765742
1.00	1.00	0.95	1.00	0.99	0.4428	0.52	0.761089
1.00	**1.00**	**0.95**	**1.00**	**1.00**	0.4506	0.52	0.756537
1.00	1.00	0.96	0.95	0.95	0.1251	0.11	0.882228
1.00	1.00	0.96	0.95	0.96	0.1299	0.12	0.877656
1.00	1.00	0.96	0.95	0.97	0.1347	0.12	0.873097
1.00	1.00	0.96	0.95	0.98	0.1395	0.12	0.868551
1.00	1.00	0.96	0.95	0.99	0.1443	0.12	0.864019
1.00	1.00	0.96	0.95	1.00	0.1491	0.12	0.859502
1.00	1.00	0.96	0.96	0.95	0.1647	0.20	0.857755

Table 5. Solution of the two-criteria optimization task K(**R**) ≤ 160 000 €, R(**R**) ≥ 0,8

R_1	R_2	R_3	R_4	R_5	$\overline{R(\boldsymbol{R})}$	$\overline{K(\boldsymbol{R})}$	$d[(\overline{K(\boldsymbol{R})}, \overline{R(\boldsymbol{R})}, (0,1)]$
1.00	1.00	0.99	0.99	0.99	0.5888	0.77	0.876889
1.00	1.00	0.99	0.99	1.00	0.5981	0.78	0.874058
1.00	1.00	0.99	1.00	0.95	0.7705	0.88	0.909792
1.00	1.00	0.99	1.00	0.96	0.7821	0.88	0.908320
1.00	1.00	0.99	1.00	0.97	0.7936	0.88	0.907053
1.00	1.00	0.99	1.00	0.98	0.8052	0.88	0.905995
1.00	1.00	0.99	1.00	0.99	0.8168	0.89	0.905147
1.00	1.00	0.99	1.00	1.00	0.8284	0.89	0.904515
1.00	1.00	1.00	0.95	0.95	0.2088	0.51	0.942643
1.00	1.00	1.00	0.95	0.96	0.2145	0.51	0.938661
1.00	1.00	1.00	0.95	0.97	0.2201	0.52	0.934730
1.00	1.00	1.00	0.95	0.98	0.2258	0.52	0.930854
1.00	1.00	1.00	0.95	0.99	0.2315	0.52	0.927034
1.00	1.00	1.00	0.95	1.00	0.2372	0.52	0.923273
1.00	1.00	1.00	0.96	0.95	0.2720	0.59	0.939384

5 Summary

The references include a description of the method for determining the optimum software reliability structure of a known module structure, assuming that the software production process covers independent creation of its component modules, where the software reliability structure is understood as the vector of the reliability indicators of the software component modules. The knowledge of the optimum reliability software structure allows proper planning of time and expenses required for the purpose of the software production, if the assumed reliability and cost requirements are met. The optimum reliability software structure is determined based on the solution to a specific one- or multi-criteria optimization problem. To establish the criterial functions or limitations of such mathematical programming
problem, the program analyzed by the control transfer graph is modeled, where the probability of activation of the particular component modules in the software execution process result from the so-called operational profile of the software, which characterized the actual working environment of the software. Such graph model of the software allows, in particular, obtaining analytical dependency, defining the probability of the correct, single execution of the software on the basis of the Marvon chain theory.

The numerical example, which illustrates the above-described deliberations, confirmed a possibility of obtaining the optimum reliability software structure by way of solving the mathematical programming problem, whereas - in accordance with the expectations - the structure, indicated as the solution to the on-criterion optimization problem, turned out to be the same as in case of the solution to the polyoptimization problem.

References

1. R.C. Cheung, *A user-oriented software reliability model*. IEEE Transactions on Software Engineering, **SE-6**, 2, 118-125 (1980)
2. D.W. Coit, *Economic allocation of test times for subsystem level reliability growth testing*. IEEE Transactions on Software Engineering, **30**, 12, 1143-1151 (1998)
3. Y. S. Dai, M. Xie, K.L. Poth, B. Tang, *Optimal testing-resource allocation with genetic algorithm for modular software systems*. The Journal of Systems and Software, 66, 47-55 (2003)
4. W. Everett, *Software component reliability analysis*, in: Proceedings of the Symposium on Application-specific Systems and Software Engineering Technology (ASSET'99), 204–211 (1999)
5. S. Gokhale, W.E. Wong, K. Trivedi, J.R. Horgan, *An analytical approach to architecture based software reliability prediction*, in: Proceedings of the Third International Computer Performance and Dependability Symposium (IPDS'98), 13–22 (1998)
6. K. Goseva-Popstojanova, K.S.Trivedi, *Architecture-based approach to reliability assessment of software systems*. Performance Evaluation, 45, 179-204 (2001)
7. C.Y. Huang, J.H. Lo, *Optimal resource allocation for cost and reliability of modular software systems in the testing phase*. The Journal of Systems and Software, 79, 653-664 (2006)
8. S. Krishnamurthy, A.P. Mathur, *On the estimation of reliability of a software system using reliabilities of its components*, in: Proceedings of the Eighth International Symposium on Software Reliability Engineering (ISSRE'97), 146–155 (1997)

9. P. Kubat, *Assessing reliability of modular software*, Operations Research Letter, 8, 35–41 (1989)
10. W. Kuo, V.R. Prasad, *An annotated overview of system reliability optimisation*. IEEE Transactions on Software Engineering, **49**, 2, 176-187 (2000)
11. J.C. Laprie, *Dependability evaluation of software systems in operation*, IEEE Transactions on Software Engineering, **10**, 6, 701–714 (1984)
12. Y.W. Leung, *Dynamic resource-allocation for software-module testing*. The Journal of Systems and Software, **37**, 2, 129-139 (1997)
13. B. Littlewood, *Software reliability model for modular program structure*, IEEE Transactions on. Reliability, **28**, 3, 241–246 (1979)
14. J.D. Musa, *Software–Reliability-Engineered Testing Practice*. McGraw-Hill, New York (1998)
15. H. Ohtera, S. Yamada, *Optimal allocation & control problems for software-testing resources*. IEEE Transactions on Software Engineering, **39**, 2, 171-176 (1990)
16. J. Rajgopal, M. Mazumdar, *Modular operational test plans for inferences on software reliability based on a Markov model*. IEEE Transactions on Software Engineering, **28**, 4, 358-363 (2002)
17. M. Shooman, *Structural models for software reliability prediction*, in: Proceedings of the Second International Conference on Software Engineering, 268–280 (1976)
18. J. Stadnicki, *Teoria i praktyka rozwiązywania zadań polioptymalizacji*, WNT (2006).
19. S. Wolfram, *The Mathematica book*. Cambridge University Press and Wolfram Media (1996)
20. K. Worwa, *Application of mathematical modelling to the solution of software testing problems*, in: Proceedings of the XII International Conference System-Modelling-Control (2009)
21. K. Worwa, *Estimation of the program testing strategy*. Postępy Cybernetyki, 3-4, 155-188 (1995)
22. M. Xie, C. Wohlin, *An additive reliability model for the analysis of modular software failure data*, in: Proceedings of the Sixth International Symposium on Software Reliability Engineering, 188–194 (1995)
23. S. Yacoub, B. Cukic, Ammar H., *Scenario-based reliability analysis of component-based software*, in: Proceedings of the 10th International Symposium on Software Reliability Engineering, 22–31 (1999)
24. S. Yamada, T. Ichimori, M. Nishiwaki, *Optimal allocation policies for testing-resource based on a software reliability growth model*. International Journal of Mathematical and Computer Modelling, **22**, 10, 295-301 (1995)