LogForum

> Scientific Journal of Logistics <

2018, 14 (2), 209-219

http://dx.doi.org/10.17270/J.LOG.2018.259

http://www.logforum.net p-ISSN 1895-2038 e-ISSN 1734-459X

ORIGINAL PAPER

OPTIMIZING THE STRUCTURE OF SOFTWARE SYSTEMS SUPPORTING LOGISTICS AT THE DESIGN STAGE

Kazimierz Worwa

Military Technical University, Warsaw, **Poland**

ABSTRACT. **Background:** Computer software, widely used to support a broad range of logistics activities, is characterized both by increasing functionality and increasing complexity. For this reason, the process of software production, including the stages of specification of requirements, design, programming and testing, is time-consuming and expensive. The main goal of the software design phase is to determine the software architecture that identifies all software components and defines links and connections between them. The design phase also includes the development of the so-called internal logic of all extracted components, that is, detailed elaboration of algorithms for their operation and defining the structure of data used. It should be emphasized that the results of the software design process depend greatly on the knowledge and experience of the designer, because there are no universal behavioral patterns in this area. The main goal of the approach proposed at work is to reduce the role of the subjective factor in the results of the software development process. The focus of this work is on this software development process within logistics processes.
Methods: The basic research method used in the work is mathematical modeling. The paper proposes a formal method of assigning the modular structure of the computer program by formulating and solving the corresponding double-criterion optimization problem. The module strength coefficient and module-coupling coefficients were established as modularization criteria of the program.
Results: The main result of the work is the method of determining the modular structure of the designed program by determining the solution of the two-criterion optimization problem. The numerical example developed to illustrate this entirel confirms the possibilities of its practical application. The modular structure of the program, based on the solution of the formulated polyoptimization task, is characterized by the maximum value of the modular power coefficient and the minimum value of the modular strength coefficient. According to the latest trends in software engineering, it is the optimal structure. The method can be useful in the process of designing software for systems supporting the implementation of logistics processes.
Conclusions: The author's method of determining the modular structure of the program, presented in the article, is an unprecedented attempt in the literature to use formal methods in the software design process, which could be implemented practically in the logistics processes. The lack of similar attempts probably results from the very low compliance of the software design stage with attempts to formalize it. In order to increase the possibility of practical application of the proposed method, it seems reasonable to conduct further work to implement the methods of developing software requirements specifications in a formalized form, e.g. with the use of mathematical notation.

Key words: program design, modular structure, modularization criteria, program structure optimization.

INTRODUCTION

Logistics is one of those sectors that could not function effectively without information technology (IT) support. The growing importance of competition in the area of logistics and the constant emphasis on increasing the efficiency of enterprises' operation drives the search for new solutions and technologies and increases the importance of using modern IT systems. Computer software, widely used to support logistics activities understood in the broad sense, is characterized by increasing functionality and also by increasing complexity. For this reason, the process of producing software is time-consuming and expensive. The traditional

Citation: *Worwa K., 2018. Optimizing the structure of the software systems supporting logistics on the stage of its design. LogForum 14 (2), 209-219, http://dx.doi.org/10.17270/J.LOG.2018.259*
Received: 26.10.2017, Accepted: 12.03.2018, on-line: 29.03.2018.

organization of the software production process includes 4 main stages of work: specification of requirements, project development, implementation and testing. Practical methods of implementing these stages, including the techniques and tools used for design and implementation, as well as methods of organizing the work of executive teams, are of interest to software engineering. Among all the stages of a complex computer program development process, the design stage is one of the most important due to the possibility of shaping usability. According to the analytical design [Pressman 2015, Hohmann 2006, Hohmann 2013], the essence of a program design stage is to design a modular structure of the program under construction, defining in particular its decomposition into component modules and their interrelations. What has a very important influence on the form of modular structure of the program are the criteria assumed for its division into modules, henceforth referred to as modularization criteria.

Numerous modularization methods are used in practice, depending on the specificity and purpose of the programs being designed. Designing a program, like any design activity, is essentially about the invention or creative activity of the designer. Due to the fact that both the design work and its results depend greatly on the designer's knowledge and experience (the subjective conditioning of the design process), the software design stage is hardly susceptible to formalization. It should be emphasized that the formalization of design work, such as the extensive use of mathematical methods for finding specific design solutions, could be the basis for undertaking work to reduce the role of the subjective factor in design work, for example through their partial automation.

The paper presents an attempt to define the modular structure of a program by solving a suitably defined two-criterion optimization problem, using both maximization of the modular strength coefficient and minimization of the inter-module coupling of the program as the modularization criteria.

THE ROLE OF IT METHODS AND MEANS IN LOGISTICS

It is well known that modern logistics could not function effectively without using IT methods and means. The logistics departments include cells that deal with storage, transport, planning and shopping. In these areas of logistics, IT is indispensable. Proper use of IT tools is very often a condition for the implementation of the logistics systems. These tools are perceived as an essential component of the logistics systems infrastructure. In the area of modern logistics, various IT solutions are used. Some of them are typical solutions, also used in other industries, others are strictly specialized, dedicated to the logistics sector. Among the logistics functions for which IT resources are commonly used, one can mention: planning logistics processes in various cross-sections and time horizons, coordination of events, operations and logistics processes, monitoring and control of logistics operations, and operational control of logistics processes.

What are particularly important in the activities of modern enterprises, including logistics companies, are integrated IT management systems. Increasingly, operate in a distributed system, allowing logistics management of both large and small enterprises [Bartkowiak, Rutkowski 2016]. The most important ones used in logistics include: ERP (Enterprise Resource Planning), WMS (Warehouse Management System), CRM (Customer Relationship Management), MRP (Material Requirements Planning) and SCM (Supply Chain Management) systems. A broader description of the architecture and functionality of these systems can be found, for example, in Rezapour et. al, 2009.

ERP systems provide support for managing the entire enterprise. The basis for the implementation of such systems is the integration of all areas of the company's operations, through a common database, which all the cells use (production planning, sales, transport, distribution, human resources or accounting). ERP logistics systems can be adapted to almost any type of activity. Great convenience is afforded by the possibility of limiting or extending the access of individual

units to the database. Data obtained and collected in this way can be repeatedly used and processed without fear of it being lost. ERP systems are also recommended for companies that operate in many places, branches, countries, because thanks to them the whole organization operates equally, and the exchange of data between individual cells takes place in real time.

WMS systems are an effective tool for supporting all processes related to storage and warehouse functioning. Typical business processes occurring in the warehouse are: receiving and sending large batches of goods, their proper segregation, order preparation, and invoicing. WMS class systems allow for quick and efficient implementation of these processes, ensuring minimization of the likelihood of making a mistake. These systems allow the use of modern technologies in warehouse processes, based on automatic identification technologies, among which those based on bar codes or RFID technologies are becoming increasingly popular.

CRM systems are dedicated to managing customer relations. They enable continuous collection of data about current and potential clients. They increase the chances of acquiring new customers, but also for maintaining relationships with clients already using the services of a given company. As part of CRM information systems, it is also possible to conduct service and consulting activities, as well as to collect information on the needs of customers and their satisfaction with the services offered. Better contact with the customer is not only a guarantee of their loyalty, but is also an additional source of feedback on the company and can serve to build its strong position in a competitive market.

MRP systems are designed to manage the production process, enabling, among other things, optimization of production of different elements and their components, optimization of inventories, and estimation of production costs, as well as better use of the enterprise infrastructure.

The functionality of SCM class systems includes procurement, production and distribution, and, to a greater extent, the integration of the enterprise with suppliers and recipients. SCM systems often require the integration of several IT subsystems, which significantly increases the efficiency of information flow, facilitating, among other things, the adjustment of demand and supply to each other, more efficient customer service, increased competitiveness through the optimal flow of materials and lower storage costs, as well as improved cooperation with suppliers and recipients.

Areas of logistics create many opportunities for the introduction of new technological innovations in the field of information systems. New, advanced technologies create many opportunities for improving logistics management. It is worth paying attention to the possibilities offered by RFID technology in this area. A description of an exemplary use of these possibilities of this technology in a logistics information system is presented in the study by Nowicki et. al [2017]. Current technological progress in the design, construction and operation of integrated information management systems is a basic condition for the development of logistics.

PROGRAM MODULARIZATION CRITERIA

Designing the modular structure of the program is based on the specification of software requirements. For the purposes of further consideration, it is assumed that these specifications are expressed by a specially-designed decision table (the 'cause-effect table') describing in a precise and unambiguous way the program input and the actions that it performs. The method of constructing such a table is described in [Myers 2012]. In the papers cited, cause-effect tables are constructed for the purpose of designing a set of test cases, which is the basis for the testing phase. Cause-effect table describes the relationship between the so-called causes that are possible combinations of input data from the program and their effects, understood as actions taken by the program following the occurrence of these cases.

Let be the set of numbers - resulting from the specification of requirements - possible combinations of causes and - the set of numbers of their effects.

Let $\boldsymbol{J} = \{ 1, 2, ..., j, ..., J \}$ be the set of numbers - resulting from the specification of requirements - possible combinations of causes and $\boldsymbol{I} = \{ 1, 2, ..., i, ..., I \}$ - the set of numbers of their effects.

With reference to the previously mentioned cause-effect table, the value I is the number of lines of the "effect" part of the table, and J denotes the number of their columns.

In the remainder part of this article, the cause-and-effect table describing the specifications of the analized program will be represented by the matrix T, as follows:

$$T = \left[t_{ij} \right]_{I \times J}, \qquad (1)$$

where

$$t_{ij} = \begin{cases} 1 & \text{if the } i\text{-th efect is a consequense of the } j\text{-th combination of causes,} \\ 0 & \text{otherwise.} \end{cases}$$

According to previous remarks, the particular effects represented in the matrix T as "one" are understood as actions performed by the program following the occurrence of specific combinations of causes (input data).

There are a number of mutually related modules (in the sense of sequence for example) in the program design process, which represent the programming implementation of these actions. The complexity and number of implemented actions determines the number and nature of reciprocal links of the specified modules. Let $\boldsymbol{M} = \{ 1, 2, ..., m, ..., M \}$ denote the set of program module numbers, wherein the range of values that number M can take is as follows:

$$1 \le m \le I \ ,$$

where $M = 1$ means no modularization (the so-called one-module program), while $M = I$ is the maximum modularization that takes place when every program module implements exactly one action (function).

Determining the number of modules M as well as their interrelationships is a fundamental difficulty in the process of designing the modular structure of the program. Using the matrix T, which is a representation of the cause-effect table of the designed program, the problem of determining the modular structure of the program can be reduced to the problem of determining the "allocation" of each action to particular modules, i.e. to determine the number of modules and actions that they will execute. The mentioned assignment, hereinafter referred to as the letter X, is defined as follows:

$$X = \left[x_{im} \right]_{I \times M}, \qquad (2)$$

where

$$x_{im} = \begin{cases} 1 & \text{if the } i\text{-th action is realized by the } m\text{-th module,} \\ 0 & \text{otherwise.} \end{cases}$$

Elements of the matrix X fulfill the following constraints:

$$\sum_{m \in \boldsymbol{M}} x_{im} \ge 1, \quad i \in \boldsymbol{I}, \qquad (3)$$

where equality in dependence (3) takes place in a situation in which every action (function) can be executed (implemented) by exactly one module.

Assignment X assigns the division of the set of effect numbers I into subsets defined as follows:

$$\boldsymbol{I}_m(X) = \{ i \in \boldsymbol{I} : x_{im} = 1 \}.$$

According to the above definition set $\boldsymbol{I}_m(X)$, $m \in \boldsymbol{M}$, contains the numbers of these actions, which according to assignment X implements the m-th module.

In cases where dependencies (3) are equations sets $\boldsymbol{I}_m(X)$, $m \in \boldsymbol{M}$, are disjoint sets.

The nature of the mutually relations between modules is determined by the interrelationships of the particular effects - in the sense of their temporal consequence - during the execution of the program. Mutual effect relationships will be characterized by functions Γ^j, $j \in \boldsymbol{J}$, defined as follows:

$$\Gamma^j : I \to 2^I \quad j \in \boldsymbol{J} \tag{4}$$

The value of the function for the *i*-th action is the set of numbers of those program actions that for the *j*-th combination of causes the program can execute next. For example, if $\Gamma^j(i)=\{k,l\}$ then for the *j*-th combination of causes, after execution of the *i*-th action as the next one can be executed the *k*-th or *l*-th action.

Designing a modular structure of a program involves determining the number of modules (by determining their "content", i.e. the actions (functions) that they will implement) and their interrelations. This structure - dependent on assignment *X* - will be characterized by the zero-one matrix *Γ(X)*, defined as follows:

$$\Gamma(X)=[\gamma_{mn}(X)]_{M\times M}, \tag{5}$$

where

$$\gamma_{mn}(X)=\begin{cases} 1 & \text{if according to assignment s X wfter execution the m - th module the n - th module can be executed as next,} \\ 0 & \text{otherwise.} \end{cases}$$

Using the knowledge of function Γ^j, $j \in \boldsymbol{J}$, quantities $\gamma_{mn}(X)$, $m, n \in \boldsymbol{M}$, can be defined as follows:

$$\gamma_{mn}(X)=\begin{cases} \underset{j\in J}{\mathrm{X}} \underset{h\in I_m(X)}{\mathrm{X}} \underset{i\in I_m(X)}{\mathrm{X}} \gamma_{hi}^j(X) & \text{for } m \neq n, \\ 0 & \text{for } m = n, \end{cases} \tag{6}$$

and

$$\gamma_{hi}^j(X)=\begin{cases} 1 & \text{if } i \in \Gamma^j(h) \\ 0 & \text{othetwise,} \end{cases}$$

where operator X means logical summation of zero-one values, i.e.

$$\underset{i\in \boldsymbol{I}_m(X)}{\mathrm{X}} \gamma_{hi}^j(X) = 1 - \prod_{i\in \boldsymbol{I}_m(X)} (1-\gamma_{hi}^j(X)) \cdot$$

As mentioned earlier, modularization criteria have decisive importance in the process of designing the modular structure of the program. In particular, these criteria determine the form of the matrix *X*, describing the way of grouping actions into modules. From practical experience it follows that the modular structure of the program - among other things, to ensure its high reliability, ease of use and possible modifications - should be characterized by maximum simplicity. In this work, as a way to achieve this simplicity, simultaneously maximization of the modular strength coefficient and minimization of the inter-module coupling of the program is proposed. It should be stressed that this approach is consistent with the latest trends in modern software engineering [Pressman 2015, Hohmann 2006, Hohmann 2013].

The strength of a module is a measure of the nature and strength of inter-modular (internal) links between particular parts (elements) of a module. The inter-module coupling of the program is a measure of the number and types of inter-module (external) links, that is, between the highlighted program modules. The term "link" that is present in the above quoted definitions most commonly means the data coupling in practice [Myers 1975].

There are several types of module strength and module coupling categories in literature. For example, Myers [Myers 1975] distinguishes 7 module strength categories (random, logical, classic, algorithmic, communication, information) and 6 categories of module coupling (content, common, external, control, features, data).

Let $\boldsymbol{A} = \{1, 2, ..., a, ..., A\}$, $\boldsymbol{B} = \{1, 2, ..., b, ..., B\}$ denote sets of module strength categories and module coupling respectively, and these numbers are assumed to be assigned to each category in such a way that the higher number

corresponds to a higher strength or coupling category.

The strength of the individual modules and the strength of their interconnections depends on the described by the matrix X, grouping method of actions realized by the program.

Let $A(X)$ denote the M-element vector, the components of which determine the strength of the individual program modules for the assignment X:

$$A(X)=(a_1(X),a_2(X),...,a_m(X),...,a_M(X)),\ a_m(X)\in \boldsymbol{A},\ m\in \boldsymbol{M}$$

Respectively, let $B(X)$ denote the matrix characterizing the strength of module coupling of the considered program for the assignment X:

$$B(X)=[b_{mn}(X)]_{M\times M},$$

where $b_{mn}(X)$ is the number of the m-th and n-th module coupling categories, for the assignment X.

The following two coefficients will be used as criteria for program modularization:

- module strength coefficient of the program:
$$F_1(X)=\min_{m\in \boldsymbol{M}} a_m(X) \quad (7)$$
- module coupling coefficient of the program:
$$F_2(X)=\max_{(m,n)\in \boldsymbol{M}\times\boldsymbol{M}} b_{mn}(X) \quad (8)$$

The value of the coefficients (7) and (8) in a particular way characterize the modular structure of the program, wherein the way of their construction shows that the module strength value is equal to the "weakest" number - among all component modules - the module strength category, while the module coupling value in the program is determined by the category number of the pair of modules "most strongly" interrelated.

FORMULATING A PROBLEM TO OPTIMIZE THE ALLOCATION OF ACTIONS TO INDIVIDUAL PROGRAM MODULES

Based on the introduced denotations and the assumed program modularization criteria it is possible to formulate the following two-criterion optimization problem of assignment of actions (functions) to particular modules of designed program can be formulated:

$$F(\boldsymbol{X}, F, R) \quad (9)$$

where:

X – a set of permissible solutions, defined as follows:

$$\boldsymbol{X}=\{X=[x_{im}]_{I\times M}: X \text{ satisfies constraint (2) and (3)}\}; \quad (10)$$

F - two-criterion quality coefficient of the solution quality of the form

$$F(X)=(F_1(X),\ F_2(X)), \quad (11)$$

wherein the component criteria F_1, F_2 are defined by the relationships (7) and (8) respectively;

R - the dominance relation in the set of values of the quality coefficient, defined as follows:

$$R=\{(y_1,y_2)\in \boldsymbol{Y}\times\boldsymbol{Y}: y_1^1\geq y_2^1,\ y_1^2\leq y_2^2\} \quad (12)$$

where $\boldsymbol{Y}$ is so called criteria space [Eschenauer, Koski, Osyczka 1990], defined as follows:

$$\boldsymbol{Y}=F(X)=\{y=(F_1(X),\ F_2(X)): X\in \boldsymbol{X}\} \quad (13)$$

where in

$$y_1=(y_1^1,\ y_1^2),\ y_2=(y_2^1,\ y_2^2).$$

In the presented formulation of the problem of determining the optimal assignment, the dimension of the matrix X is set to $I\times M$. In cases where the number of modules M cannot be determined in advance $M = I$ must be pre-determined. After determining the solution $\overline{X}$ of the task (9-13), the real number of modules

of the program under consideration is obtained by summing these columns of the matrix $\overline{X}$ in which there is at least one "1".

According to previous remarks, the knowledge of the assignment X allows to define the modular structure of the designed program, while the assignment $\overline{X}$, which is the solution of the two-criterion optimization problem (9-13), corresponds to the optimal modular structure in the sense of the assumed criteria. The solution of the optimization problem (9-13) can be determined in accordance with accepted methodology of solving multi-criterion optimization tasks [Eschenauer, Koski, Osyczka 1990].

NUMERICAL EXAMPLE

For the illustration of the considerations that have been discussed, a simple numerical example will be presented.

Let the sets $\boldsymbol{I}$, $\boldsymbol{J}$, the matrix T and the functions Γ^j, $j \in \boldsymbol{J}$, describing the specifications of the sample requirements of the designed program have the following form:

$$\boldsymbol{I} = \{1, 2, 3, 4\},$$

$$\boldsymbol{J} = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\},$$

$$T = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \end{bmatrix},$$

$$\Gamma^1(1) = \Gamma^2(3) = \{4\},$$

$$\Gamma^5(1) = \{3, 4\}, \Gamma^5(3) = \{4\},$$

$$\Gamma^6(2) = \{4\},$$

$$\Gamma^9(2) = \{1, 3\},$$

$$\Gamma^{10}(2) = \{1\},$$

wherein in description of the function Γ^j, $j \in \boldsymbol{J}$, the cases in which their values are empty are omitted.

According to the accepted assumptions, the set of permissible solutions (assignments) $\boldsymbol{X}$, defined by the relation (10), has the following form:

$$\boldsymbol{X} = \{X_1, X_2, X_3, X_4, X_5\},$$

where:

$$X_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}, \quad X_2 = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad X_3 = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix},$$

$$X_4 = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad X_5 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Above permissible assignments correspond to the following five cases:

1. all four program actions are implemented by one module (X_1);
2. one module executes three actions and the other one (X_2);
3. each of two modules executes two actions (X_3);
4. one module executes two actions, the other two - one action (X_4);
5. each module executes exactly one action (X_5).

Corresponding to the particular permissible assignments matrices $\Gamma(X_i)$, $i = \overline{1,5}$, describing the possible modular structures of the analyzed program are defined as follows:

$$\Gamma(X_1) = [0], \; M = 1, \; \boldsymbol{I}_1(X_1) = \{1, 2, 3, 4\};$$

$$\Gamma(X_2) = \Gamma(X_3) = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \; M = 2, \; \boldsymbol{I}_1(X_2) = \{1, 2, 3\}, \boldsymbol{I}_1(X_3) = \{1, 2\}$$

$$\Gamma(X_4)=\begin{bmatrix}0&1&1\\0&0&1\\0&0&0\end{bmatrix},\ M=3,\ \boldsymbol{I}_1(X_4)=\{1,2\},\ \boldsymbol{I}_2(X_4)=\{3\},\ \boldsymbol{I}_3(X_4)=\{4\}$$

$$\Gamma(X_5)=\begin{bmatrix}0&0&1&1\\1&0&1&1\\0&0&0&1\\0&0&0&0\end{bmatrix},$$

$$M=4,\ \boldsymbol{I}_1(X_5)=\{1\},\ \boldsymbol{I}_2(X_3)=\{2\},\ \boldsymbol{I}_3(X_3)=\{3\},\boldsymbol{I}_4(X_3)=\{4\}.$$

Existing in analyzed sample program internal and external modular links will be evaluated based on the categories proposed by Myers [Myers 1975]), mentioned in chapter 2 i.e. $\boldsymbol{A}=\{1, 2, 3, 4, 5, 6, 7\}$, $\boldsymbol{B}=\{1, 2, 3, 4, 5, 6\}$.

Let matrices $A(X_i)$, $B(X_i)$, $i=\overline{1,5}$, will be defined as follows:

$$A(X_1)=[1],\ B(X_1)=[0];$$

$$A(X_2)=[1\ \ 2],\ B(X_2)=\begin{bmatrix}0&4\\4&0\end{bmatrix};$$

$$A(X_3)=[3\ \ 3],\ B(X_3)=\begin{bmatrix}0&3\\3&0\end{bmatrix};$$

$$A(X_4)=[3\ \ 2\ \ 2],\ B(X_4)=\begin{bmatrix}0&3&2\\3&0&4\\2&4&0\end{bmatrix};$$

$$A(X_5)=[3\ \ 5\ \ 2\ \ 2],\ B(X_5)=\begin{bmatrix}0&5&2&2\\5&0&3&1\\2&3&0&4\\2&1&4&0\end{bmatrix}.$$

The values of the coefficients, assumed as component criteria for modularization, according to (7) - (8), have the form:

$$F_1(X_1)=1,\ F_1(X_2)=1,\ F_1(X_3)=3,\ F_1(X_4)=2,\ F_1(X_5)=2$$
$$F_2(X_1)=0,\ F_2(X_2)=4,\ F_2(X_3)=3,\ F_2(X_4)=4,\ F_2(X_5)=5$$

Accordingly, the criterion space $\boldsymbol{Y}$, defined by the relation (13), forms the following set of pairs

$$F(X)=(F_1(X_i),\ F_2(X_i)),\ i=\overline{1,5}$$

$$\boldsymbol{Y}=\{(1,0),\ (1,4),\ (3,3),\ (2,4),\ (2,5)\}.$$

According to the assumed dominance relation (12) there are two non-dominated [Eschenauer, Koski, Osyczka 1990] elements in the set $\boldsymbol{Y}$, as pairs (1, 0) and (3, 3). These elements are "better" - in the sense of the relation R - from the other elements of the set $\boldsymbol{Y}$. These pairs of points are images - in the transformation F – of permissible solutions (assignments) X_1 and X_3 respectively. These solutions are therefore non-dominated solutions of the two-criteria optimization problem (9) - (13).

Solution X_1 corresponds to such situation in which all actions are implemented by only one module. In this case strength module coefficient is low, but there are no external links between modules. In turn, the solution X_3 corresponds to the case of a program in which two modules exist. In this situation value of the strength module coefficient is increased, but due to the appearance of certain links between modules, the value of the module coupling coefficient in the program it is getting worse.

Statement that solutions X_1 and X_3 are non-dominated solutions of the two-criteria optimization problem (9) - (13) means that, based on the assumed dominance relationship (12), it cannot be decided which one of them is a better solution. In general, for example for purely practical reasons, aimed at e.g. to reduce the complexity of the design-implementation task and improve the utility of the program, the designer decides probably to take the solution X_3.

CONCLUSIONS

The formalized method of designing the structure of application software presented here can be useful in the processes of designing a wide range of information systems, including designing software supporting the planning and implementation of logistics processes. The proposed method of determining the modular structure of the computer program by solving the two-criteria

optimization problem (9)-(13) requires representation of program requirements specifications in the form of matrix (1) and (4). It should be emphasized that, besides the possibility of applying the method described, this form of describing program requirements specifications also enables effective verification of their completeness and consistency, and the application of modern test data design methods, e.g. cause-effect graph methods [Myers 2012].

The modular structure of the program, based on the solution of the formulated task of polyoptimization, is characterized by the maximum value of the module strength coefficient (7) and the minimum value of the module coupling (8). According to the latest software engineering trends, this is the optimal structure.

The practical application of the proposed method requires the determination of the solution to the two-criteria optimization problem (9) - (13), i.e. determining the dominant solution set, and in the absence of it, which is very often the case, the non-dominated solutions set [Eschenauer, Koski, Osyczka 1990]. If this set contains more than one element, its "representative" must be chosen as part of the design decisions, e.g. by designating a compromise solution [Eschenauer, Koski, Osyczka 1990]. Consequently, the efficiency of the proposed method can be significantly increased by equipping the designer with the appropriate software to enable computer-aided determining of the solution to the two-criteria optimization problem (9) - (13).

In many practical situations, the set of permissible solutions X of optimization problem (9) - (13) will not be too numerous, i.e. the number of design variants will be relatively small. In such cases, the solution to this problem can be determined by the full review method.

REFERENCES

Adamczak M., Domanski R., Hadas L. et al., 2016. The integration between production-logistics system and its task environment chosen aspects. Conference: 8th IFAC Conference on Manufacturing Modelling, Management and Control (MIM), IFAC Papersonline, 49, 12, 656-661.

Ansaria S., Başderea M., Lib X., Ouyangc Y., Smilowitza K., 2018. Advancements in continuous approximation models for logistics and transportation systems: 1996–2016. Transportation Research Part B: Methodological. 107, 229-252. http://dx.doi.org/10.1016/j.trb.2017.09.019

Cyplik P., Hadas L., Adamczak M. et al.. 2014. Measuring the level of integration in a sustainable supply chain. 19th World Congress of the International-Federation-of-Automatic-Control (IFAC), Cape Town, IFAC Papersonline, 47, 3, 4465-4470.

Eschenauer H., Koski J., Osyczka A., 1990, Multicriteria design optimization: procedures and applications. Springer-Verlag, Berlin.

Felix T.S. Chan, Nan Li, Chung S.H., Saadat m., 2017. Management of sustainable manufacturing systems-a review on mathematical problems. International Journal of Production Research, 55, 4, http://dx.doi.org/10.1080/00207543.2016.1229067

Fernandes A.C., Sampaio P., Sameiro M., Truong H.Q., 2017. Supply chain management and quality management integration: A conceptual model proposal. International Journal of Quality & Reliability Management, 34, 1, 53-67. http://dx.doi.org/10.1108/IJQRM-03-2015-0041

Hohmann L., 2006, Beyond Software Architecture: Creating and Sustaining Winning Solutions. Addison Wesley.

Hou H., Chaudhry S., Chen Y. et al., 2017. Physical distribution, logistics, supply chain management, and the material flow theory: a historical perspective. Information Technology and Management, 18, 2, 107-117.

http://dx.doi.org/10.1007/s10799-015-0229-1

Myers J.G., 1975, Reliable Software Through Composite Design. New York: Petrocelli/Charter.

Myers J.G., 2012, The art of software testing. Wiley, New York.

Pressman R.S., 2015, Software engineering: a practical approach, Mc Grow-Hill, New York.

Rezapour S., Moghadam M.S., Dehkordi M.A., 2009, Logistics and Supply Chain Management Information Systems. Springer-Verlag Berlin Heidelberg.

Vaughn V., 2013, Implementing Domain-Driven Design, Addison Wesley.

Zavadskasa E.K., Turskisa Z., Vilutienėa T., Lepkovab N., 2017. Integrated group fuzzy multi-criteria model: Case of facilities management strategy selection, 82, 317-331. Expert Systems with Applications. http://dx.doi.org/10.1016/j.eswa.2017.03.072

OPTYMALIZACJA STRUKTURY OPROGRAMOWANIA WSPIERAJĄCEGO LOGISTYKĘ NA ETAPIE JEGO PROJEKTOWANIA

STRESZCZENIE. **Wstęp:** Oprogramowanie systemów komputerowych, powszechnie wykorzystywanych do wspomagania szeroko rozumianej działalności logistycznej, charakteryzuje się coraz większą funkcjonalnością, ale także coraz większą złożonością. Z tego powodu proces jego wytwarzania, obejmujący etapy: specyfikacji wymagań, projektowania, programowania oraz testowania, jest przedsięwzięciem czasochłonnym i kosztownym. Głównym celem fazy projektowania oprogramowania jest określenie architektury oprogramowania, która identyfikuje wszystkie komponenty oprogramowania i definiuje łącza i połączenia między nimi. Faza projektowania obejmuje również opracowanie wewnętrznej logiki wszystkich wyodrębnionych komponentów, czyli szczegółowe opracowanie algorytmów ich działania i określenie struktury wykorzystywanych danych. Należy podkreślić, że wyniki projektowania oprogramowania silnie zależą od wiedzy i doświadczenia projektanta, ponieważ nie ma uniwersalnych wzorców zachowań w tym obszarze. Głównym celem proponowanego podejścia jest ograniczenie wpływu wspomnianego czynnika subiektywnego na wyniki procesu projektowania oprogramowania.
Metody: Podstawową metodą badawczą zastosowaną w pracy jest modelowanie matematyczne. W pracy proponuje się formalną metodę określania struktury modułowej projektowanego programu, poprzez wyznaczenie rozwiązania odpowiednio sformułowanego zadania optymalizacji dwukryterialnej. Jako kryteria modularyzacji przyjęto wskaźnik mocy modułowej oraz wskaźnik siły powiązań międzymodułowych programu.
Rezultaty: Głównym rezultatem pracy jest autorska metoda wyznaczania struktury modułowej projektowanego programu, poprzez wyznaczenie rozwiązania formalnego problemu optymalizacji dwukryterialnej. Ilustrujący proponowaną metodę przykład numeryczny w pełni potwierdza możliwości jej praktycznego zastosowania. Struktura modułowa programu, oparta na rozwiązaniu sformułowanego zadania polioptymalizacji charakteryzuje się maksymalną wartością współczynnika tzw. mocy programu i minimalną wartością modułowego współczynnika powiązań międzymodułowych. Zgodnie z najnowszymi trendami inżynierii oprogramowania, jest to zatem struktura optymalna. Metoda może być przydatna, m.in. w procesie projektowania oprogramowania systemów wspierających realizację procesów logistycznych.
Wnioski: Przedstawiona w artykule autorska metoda określania struktury modułowej programu jest – nie mająca precedensu w literaturze przedmiotu – jest próbą wykorzystania metod modelowania matematycznego w procesie projektowania oprogramowania. Brak podobnych prób w literaturze przedmiotu prawdopodobnie wynika z bardzo niskiej podatności etapu projektowania oprogramowania na jego formalizacji. W celu zwiększenia możliwości praktycznego stosowania proponowanej metody wydaje się zasadne prowadzenie dalszych prac, służących wdrożeniu metod opracowywania specyfikacji wymagań na oprogramowanie w sformalizowanej formie, np. z wykorzystaniem zapisu matematycznego.

Słowa kluczowe: projektowanie programu, struktura modułowa, kryteria modularyzacji, optymalizacja struktury programu.

Worwa K., 2018. Optimizing the structure of the software systems supporting logistics on the stage of its design. LogForum 14 (2), 209-219. http://dx.doi.org/10.17270/J.LOG.2018.259

OPTIMIERUNG DER DIE LOGISTIK UNTERSTÜTZENDEN SOFTWARE-STRUKTUR IN DER PHASE DEREN PROJEKTIERUNG

ZUSAMMENFASSUNG. Einleitung: Die Software von Computersystemen, die im allgemeinen zur Unterstützung der breit verstandenen, logistischen Aktivitäten in Anspruch genommen werden, charakterisiert sich nicht nur durch die immer größere Funktionalität, sondern auch durch die immer höhere Kompliziertheit. Daher ist der Prozess deren Erzeugung, die folgende Etappen wie: Auflistung von Anforderungen, Projektierung, Software-Erstellung und Testen umfasst, ein zeit- und kostenaufwendiges Vorhaben. Das Hauptziel der Phase von Projektierung der Software ist es, die Software-Architektur, die alle Software-Komponenten bestimmt und die Schnittstellen zwischen ihnen definiert, festzulegen. Die Projektierungsphase umfasst auch die Ermittlung der zwischen den betreffenden Komponenten bestehenden Logik, das heißt die Bearbeitung von Algorithmen deren Funktionalitäten und die Bestimmung der Struktur von in Anspruch genommenen Daten. Da in diesem Bereich keine Handlungsmuster vorliegen, muss es hervorgehoben werden, dass die Ergebnisse der Software-Projektierung jeweils stark vom Wissen und der Erfahrung des Projektanten abhängen. Das Hauptziel der vorgeschlagenen Vorgehensweise ist die Einschränkung des Einflusses des erwähnten, subjektiven Faktors auf die Resultate des Software-Projektierungsprozesses.
Methoden: Die grundlegende, im Rahmen der vorliegenden Arbeit in Anspruch genommene Forschungsmethode ist die mathematische Modellierung. Dabei schlägt man eine formelle Methode für die Bestimmung der Modul-Struktur des projektierten Programms anhand der Festlegung einer dementsprechend formulierten Aufgabe der Zweikriterien-Optimierung vor. Als Modularisierungskriterien wurde die Kennziffer einer Modul-Power und die Kennziffer der Stärke von intermodularen Bindungen innerhalb des Programms angenommen.
Ergebnisse: Die Autoren-Methode der Kennzeichnung der Modul-Struktur eines projektierten Programms anhand der Festlegung einer dementsprechend formulierten Aufgabe der Zweikriterien-Optimierung stellt das Hauptresultat der Forschungsstudie dar. Das nummerische Beispiel, das die vorgeschlagene Methode projiziert, bestätigt in vollem Umfange die Möglichkeiten deren praktischen Anwendung. Die Modul-Struktur des Programms, die sich auf die Lösung der Aufgabe einer Polyoptimierung stützt, charakterisiert sich durch den maximalen Wert des Koeffizienten der sog. Programm-Power und den minimalen Wert des modularen Koeffizienten der intermodularen Bindungen. Laut der neuesten Trends innerhalb des Software-Ingenieurwesens gilt die betreffende Struktur als eine optimale Struktur. Die Methode kann unter anderem im Prozess der Projektierung von Software-Systemen, die die Ausführung von Logistikprozessen unterstützen, brauchbar sein.
Fazit: Die im Artikel projizierte, in der Gegenstandsliteratur einzigartige Autoren-Methode für die Bestimmung der Modul-Struktur des Programms stellt einen Versuch der Inanspruchnahme von Methoden zur mathematischen Modellierung im Prozess der Software-Projektierung dar. Der Mangel an ähnlichen Versuchen in der betreffenden Fachliteratur resultiert wahrscheinlich aus der sehr niedrigen Brauchbarkeit der Projektierungsphase in Bezug auf deren Formalisierung. Zwecks der Erhöhung von Möglichkeiten einer praktischen Anwendung für die vorgeschlagene Methode scheint die Fortsetzung weiterer Forschungsarbeiten, die der Einführung von Methoden zur Ermittlung von Software-Anforderungen in einer formalisierten Form, z.B. unter Anwendung einer mathematischen Aufzeichnung, dienen, durchaus zweckmäßig zu sein.

Codewörter: Projektierung des Programms, Modul-Struktur, Kriterien der Modularisierung, Optimierung der Software-Struktur.

Kazimierz Worwa
Military Technical University
Faculty of Cybernetics
gen. Witolda Urbanowicza 2, 00-908 Warszawa 46, **Poland**
e-mail: kazimierz.worwa@wat.edu.pl
Funding Source Declaration
Faculty of Cybernetics statutory fund