RESEARCH
IN LOGISTICS
& PRODUCTION

OPTIMIZING THE MODULE STRUCTURE OF THE PROGRAM ON THE STAGE OF ITS DESIGN

Kazimierz Worwa

Faculty of Cybernetics, Military Technical University, Warsaw, Poland,
Email: kazimierz.worwa@wat.edu.pl

Abstract The paper proposes a formal method of determining the modular structure of the computer program by formulating and solution of the corresponding two-criterion optimization problem. A module strength coefficient and a module coupling coefficients were established as modularization criteria of the program. For the illustration of the considerations that have been discussed, a simple numerical example will be presented.

Paper type: Research Paper

Published online: 31 January 2018
Vol. 8, No. 1, pp. 91–102
DOI: 10.21008/j.2083-4950.2018.8.1.7

ISSN 2083-4942 (Print)
ISSN 2083-4950 (Online)

Keywords: *program design, program modular structure, program structure optimization*

1. INTRODUCTION

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 its broadly understood usability. According to the so-called analytical design (Pressman, 2001) 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. Very important influence on the form of modular structure of the program have assumed criteria of its division into modules, further 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 realization of design works work and their results strongly depend on the designer's knowledge and experience (the so-called subjective conditioning of the design process), the stage of software design 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.

2. 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 (so-called 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 cited papers, 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 the possible combinations of input data of the program and their effects, understood as actions taken by the program following occurrence of these cases.

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:

$$\boldsymbol{T} = \lfloor t_{ij} \rfloor_{I \times J}, \tag{1}$$

where

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

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:

$$X = [\, x_{im}]_{I \times M} \,, \tag{2}$$

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:

$$\sum_{m \in \boldsymbol{M}} x_{im} \geq 1, \quad i \in \boldsymbol{I} \,, \tag{3}$$

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 $\Gamma^j, \; j \in \boldsymbol{J}$, defined as follows:

$$\Gamma^j : I \rightarrow 2^I \;\; j \in \boldsymbol{J} \tag{4}$$

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 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

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

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

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

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.

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

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, 2001).

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*:

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

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:

$$F_1(X) = \min_{m \in \boldsymbol{M}} a_m(X) \tag{7}$$

- module coupling coefficient of the program:

$$F_2(X) = \max_{(m,n) \in \boldsymbol{M} \times \boldsymbol{M}} b_{mn}(X)\ . \tag{8}$$

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.

3. 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:

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

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

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

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

$$F(X) = (\ F_1(X),\ F_2(X)\ ), \tag{11}$$

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:

$$R = \{\ (y_1, y_2) \in \boldsymbol{Y} \times \boldsymbol{Y} : y_1^1 \ge y_2^1,\ y_1^2 \le y_2^2\ \}, \tag{12}$$

where ***Y*** is so called criteria space (Eschenauer, 1990), defined as follows:

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

wherein

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

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 can not 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, 1990).

4. 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 $\Gamma^j,\ j \in \boldsymbol{J}$, 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 $\Gamma^j,\ j\in\boldsymbol{J}$, the cases in which their values are empty are omitted.

According to the accepted assumptions, the set of permissible solutions (assignments) ***X***, defined by the relation (10), has the following form:

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

where:

$$X_1=\begin{bmatrix}1\\1\\1\\1\end{bmatrix},\ X_2=\begin{bmatrix}1&0\\1&0\\1&0\\0&1\end{bmatrix},\ X_3=\begin{bmatrix}1&0\\1&0\\0&1\\0&1\end{bmatrix},\ X_4=\begin{bmatrix}1&0&0\\1&0&0\\0&1&0\\0&0&1\end{bmatrix},\ X_5=\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}.$$

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 part 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:

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

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

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

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

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

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

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

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, 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 can not be decided which one of them is a better solution. In general, for example for purely practical reasons, aimed at eg. 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.

5. CONCLUSION

The proposed method of determining the modular structure of the computer program by determining the solution of the two-criteria optimization problem (9)–(13) requires representation of program requirements specification in the form of matrix (1) and (4). It should be emphasized that – besides the possibility of application of the described method – this form of description of program requirements specification also enables, among others effective verification of their completeness and consistency and 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 of the two-criteria optimization problem (9)–(13), i.e. determining the dominant solution set, and in the absence of it – which is very common case – the non-dominated solutions set (Eschenauer, 1990). If this set contained more than one element, it must be chosen – as part of the design decisions – its "representative", e.g. by designation of so-called a compromise solution (Eschenauer, 1990). Consequently, the efficiency of the proposed method can be significantly increased by equipping the designer with the appropriate software to enable the computer aided determining the solution of 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 of this problem can be determined by the full review method.

REFERENCES

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

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

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

Pressman R.S. (2001), Software engineering: a practical approach, Mc Grow-Hill, New York.

BIOGRAPHICAL NOTES

Kazimierz Worwa is an Associate Professor at the Cybernetics Faculty of Military Technical University in Warsaw. His research interests include software reliability modelling, software testing methods and software efficiency. He is an author and co-author of many scientific publications. His paper appear in numerous journals, e.g. Control and Cybernetics or Polish Journal of Environmental Studies and proceedings of international conferences, eg. Depcos-Relcomex, Information Management or System, Modelling and Control.