1000-9825/2005/16(01)0008 ©2005 Journal of Software 软 件 学 报 Vol.16, No.1

用 HOQ 拓展概念的软件描述及其定量结构化方法*

熊 伟[1+], 渡边喜道[2], 新藤久和[2]

[1](浙江大学 工商管理系,浙江 杭州 310027)

[2](Department of Computer Science and Media Engineering, Yamanashi University, Kofu 4008510, Japan)

A Description Approach to Software by HOQ Extension Concept and Its Quantitative Structuralization

XIONG Wei[1+], Yoshimichi WATANABE[2], Hisakazu SHINDO[2]

[1](Department of Business Administration, Zhejiang University, Hangzhou 310027, China)

[2](Department of Computer Science and Media Engineering, Yamanashi University, Kofu 4008510, Japan)

+ Corresponding author: Phn: +86-571-87951770, E-mail:wxiong@zju.edu.cn, http://zjux.org, http://www.zju.edu.cn/

Received 2003-01-23; Accepted 2003-09-09

Xiong W, Watanabe Y, Shindo H. A description approach to software by HOQ extension concept and its quantitative structuralization. *Journal of Software*, 2005,16(1):8−16. http://www.jos.org.cn/1000-9825/15/8.htm

Abstract: In order to improve the quality and productivity of software, the software structure which is deeply related to them, is observed in this paper. Firstly, the problems of software structure analysis and design of current methods are examined. Then, the HOQ (house of quality) concept of the QFD (quality function deployment) is extended as a tool for software description based on the similarity between HOQ in quality design and engineering drawing in product design. By applying Simon's idea of system's near decomposability, the quantitative structuralization method is proposed based on the QMT3 (quantification method of type 3). Furthermore, the features of this method are clarified by comparison with the current methods. Finally, the effectiveness of the method for software structure analysis and design is verified by an application. Therefore, the method can help successfully for assurance of software quality.

Key words: software engineering; software quality assurance; structure design; quality function deployment (QFD); house of quality (HOQ)

摘 要: 为了改善软件的质量和生产性,注目于与其密切相关的软件结构.首先,讨论了现有方法中软件结构分析和设计的问题;然后根据质量设计中的质量屋 HOQ(house of quality)与产品设计中的工程图之间的相似性,将质量功能展开 QFD(quality function deployment)中的质量屋 HOQ 概念扩展成一种软件描述工具,并引入 H.A.西蒙的系统准分解可能性原理,提出了基于数量化理论 3 类 QMT3(quantification method of type 3)的软件定量结构化方法,再通过与现有方法的比较分析,阐明了该方法的特点;最后,通过应用实例验证了该方法对软件结构分析和设计的有效

* Supported by the National Natural Science Foundation of China under Grant No.70472056 (国家自然科学基金)

XIONG Wei was born in 1963. He is a professor at Zhejiang University. His research interests are software quality control and software engineering. **Yoshimichi WATANABE** was born in 1964. He is an associate professor at Yamanashi University, Japan. His current research area is computer science. **Hisakazu SHINDO** was born in 1947. He is a professor and doctoral supervisor at Yamanashi University, Japan. His current research areas are Total Quality Management (TQM) and software development methodology.

性.因而为软件质量保证提供了有力的支持.
关键词: 软件工程;软件质量保证;结构设计;质量功能展开(QFD);质量屋(HOQ)
中图法分类号: TP311 **文献标识码**: A

With large-scale software production, its complexity has become a prominent problem, and it is more important to improve the quality and productivity of software. We focus our attention on the software structure which is deeply related to them. Structure is one of the major factors which affects the quality and the productivity of artificial product. In designing hardware products, physical laws and principles exist, and natural structure is drawn based on them. These laws and principles can be regarded as what gives the standard of structuralization. In designing software, it is important to pay attention to structure so that it may be symbolized by the SASD (structured analysis & structured design) method[1]. Various techniques[1,2], such as prototype technique, data structure oriented method and object-oriented method, have also been proposed in software engineering. However, the deriving of the software structure is deeply dependent on the skill of a designer. As a result, the reusing of software and quality assurance have not been effectively promoted.

On the other hand, QFD (quality function deployment)[3] was put forward by Akao, and it has been developed and used in various companies in the world. In the beginning, the objective of the QFD was focused on quality assurance for a new product development. Later, QFD has been extended to deploy not only quality in narrow sense but also other factors to be considered in the processes of planning and design. In fact, the present QFD involves engineering, cost, and reliability as well as quality. QFD has kept deeply relations to designing. This implies that QFD itself can be regarded as a method of developing a new product.

From the above viewpoint, we examine the analogy between quality design in QFD and product design. A quality table called house of quality (HOQ) has played a key role in the former, and an engineering drawing in the latter. Therefore, we believe that HOQ is equivalent to a tool, which enables one to design something shapeless or invisible such as quality and software. In other words, HOQ can describe a product in such a way as it employed in the engineering drawing.

Thus, in this paper, we put forward a method of software description in the form of HOQ or a set of linked HOQ. By applying Simon's idea of system's near decomposability[4], we put forward a method to structuralize software description. The method uses quantification method of type 3 (QMT3)[5] and leads to a more understandable structure of description. This indicates that the method is useful to analyze or design the structure of software system, and is meaningful towards the assurance of software quality.

1 Present Research State of Software Structure Analysis and Design

In software analysis and design, various methods[1,2] have been proposed until now. In order to improve maintenance and reliability, an initial method of software analysis and design was proposed by D. Parnas in 1972. The Warnier method proposed in 1974 is one focusing on the input data structure. M.A. Jackson proposed a data structure-oriented method in 1975, which is often used even now. The SASD (Structured Analysis and Structured Design) method was proposed by Yourdon and Constsntine in 1978 and has been improved by Tom Demarco in 1979. The Yourdon method consists of structured analysis (SA), structured design (SD), and structured programming (SP), and was most widely used for the 80s. PAM (Problem Analysis Method) was proposed by the HITACHI company at the end of the 80s, and is still a popular method in Japan. Since the 90s, object-oriented method[3] has become the mainstream method of software analysis and design, and the object-oriented development method OMT (Object Modeling Technique) is formed by the development from Object Oriented Programming (OOP) into Object Oriented Design (OOD) and Object Oriented Analysis (OOA).

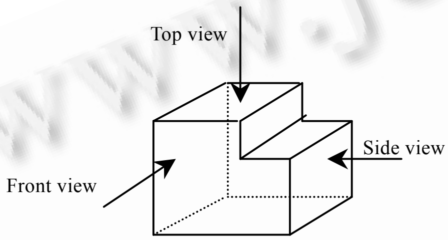
By all the above methods, the structuralization or hierarchical decomposition is performed in a certain form. For example, the Yourdon method performs functional decomposition by the top down formula, and the data structure oriented methods, such as the Jackson method and the Warnier method, perform structuralization centering on data. However, the standard of structuralization of these methods is not very clear. It points out that modeling of an object system is dependent on a designer's viewpoint, and it is also about the object-oriented method OMT, which has attracted attention in recent years. Therefore, the structure of software is easy to be dependent on a designer's subjectivity, and it has also become the reason for affecting the result greatly.

In order to design software equipped with good structure, the software which is going to be realized poses problems. Those problems are important for assurance of software quality.

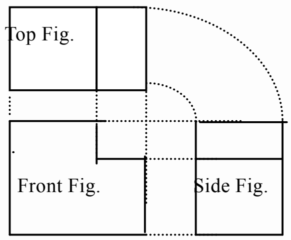
2 Extension of HOQ Concept as Software Description Method

2.1 Engineering drawing in product design

Engineering drawing has already been established and standardized specifically in the field where the object of designing is something shaped. Therefore, it is significant to review the conventional drawing system.



(a) Something shaped



(b) A description of the something shaped

Fig.1 Engineering drawing in product design

Basically, the principles used in the conventional drawing system can be illustrated in Fig.1. As shown in Fig.1(a), a solid body or something shaped is wrapped by a cube and is projected to each side of the cube in three orthogonal directions. Then, the cube is developed in two-dimensional form, and eventually expressed as in Fig.1(b) called a "drawing". In other words, "drawing" consists of three orthogonal profiles and their relations. It is of theoretical and practical importance that we can examine the shape, size and structure of the solid body indirectly based on the "drawing". It should be noticed that designing the object directly as it is in three dimensions is more difficult even if it is a solid body or something shaped. In this sense, drawing is most helpful to design.

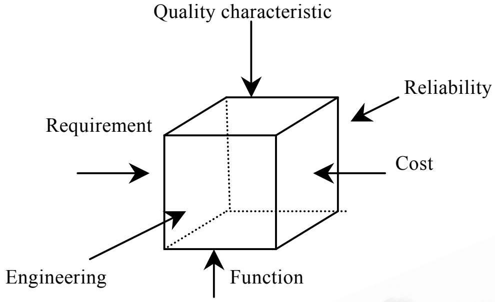
2.2 HOQ in quality design of QFD

Quality Function Deployment (QFD) is defined as converting the customer's requirements into "quality characteristics" and QFD develops design quality of the product by systematically deploying the relationships between the requirements and the characteristics, starting with the quality of each functional component and extending to the quality of each part and process. The overall quality of the product will be formed through this network of relationships.

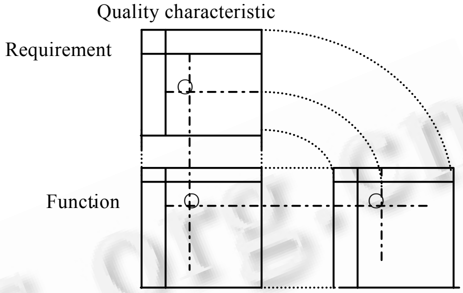
QFD is basically implemented by a set of tables consisting of different aspects or factors of a product and their relationships. The most important table is called house of quality (HOQ), which includes users' requirements and quality characteristics. Relationships among them can be used for transforming users' requirements into quality characteristics. This is one of the key activities for quality design.

As shown in Fig.2, it can be recognized that HOQ in QFD describes something shapeless from two points of view. One is from requirements for quality; the other is from quality elements or characteristics. These two factors and their relationships constitute a quality house.

In an embodiment design of a product, three different views and their relationships are naturally employed in engineering drawings. The engineering drawing is sufficient to describe the shape of the product. In this sense, it can be said that a quality house and an engineering drawing are established on the basis of the similar concept.



(a) Hypercube model of something shapeless



(b) HOQ like description

Fig.2 HOQ in quality design of QFD

2.3 Software description by HOQ

Quality assurance should start with quality planning and design. It shows that the usual HOQ consists of requirements for quality and quality elements or characteristics. The concept of HOQ is extended as a software description method and a description of software in the same manner of HOQ-like description. However, it comes into a question that what kind of factors should be selected to describe a software system. Fortunately, software engineering has shown that functions and data contained in the software might be of theoretical importance for design software.

Then, we can employ both functions and data for software description. Although most software design techniques[1,2] have paid attention to only one factor, the description proposed here is characterized by the fact that both functions and data are simultaneously treated as coordinate factors. This results come from the extended the concept of HOQ.

3 The Quantitative Structuralization Method

3.1 Meaning of structuralization

Software description is in the form of HOQ cannot be always ready to understand. There may be some function and data, which are generally overlooked, not noticed, and their relationship may not necessarily be exact, either. It is the purpose of structuralization to translate into the form, which can be enough understood based on such an imperfect description. That is, the meaning of structuralization enables it to understand, while making imperfect description perfect and decreasing complexity. In order to reduce the complexity, we utilize the idea of system's being near decomposability proposed by Simon. Simon who was awarded the Nobel Prize of economy, he states that the complexity has a hierarchy and it comes from the limit of a human's simultaneous understandability. Therefore, we should accomplish not arranging information but also discovering a structure from the complexity. At least such a recognition is significant as it has been carried out in the designing of a solid body. Moreover, Simon also proposed an idea of system’s being nearly decomposable which seems to be very closely related to discovering a structure of systems or products. Unfortunately, Simon did not give the definite procedures to achieve the system’s near decomposition.

Suppose that a software description shown in Table1 (a) is originally obtained, where functions and data are denoted by symbols A-F and a-h, respectively, and a symbol "O" represents a relationship between them. Alphabetical orders appearing in the row and column items have no specific meanings. Therefore, exchanges between the row and column can be allowed while the content determined by Table 1(a) is kept as it is. For example, apply the exchange sequence (a,f), (b,h), (c,g), (d,g), (e,g) and (f,g) yields Table 1(b). Furthermore, exchanges (A,D), (B,E), (C,E)and (C,F) transform Table 1(b) into 1(c) from which we can understand the structure with relative ease.

Table 1 Software description and its structuralization

(a)

	a	b	c	d	e	f	g	h
A	•	•	•	•	O	O	•	•
B	O	O	O	•	•	O	•	•
C	O	O	O	•	•	•	•	•
D	•	•	•	O	•	•	O	O
E	•	•	•	O	O	O	•	•
F	•	•	•	O	•	•	O	O

(b)

	g	h	d	e	f	a	c	b
A	•	•	•	O	O	•	•	•
B	•	•	•	•	O	O	O	O
C	•	•	•	•	•	O	O	O
D	O	O	O	•	•	•	•	•
E	•	•	O	O	O	•	•	•
F	O	O	O	•	•	•	•	•

(c)

	g	h	d	e	f	a	c	b
D	O	O	O	•	•	•	•	•
F	O	O	O	•	•	•	•	•
E	•	•	O	O	O	•	•	•
A	•	•	•	O	O	•	•	•
B	•	•	•	•	O	O	O	O
C	•	•	•	•	•	O	O	O

Table 1(c) indicates that the system described by the proposed method consists of three subsystems, (D,F)-(g,h,d),(E,A)-(e,f)and (B,C)-(a,c,b). And the (E)-(d) and (B)-(f) can be regarded as interfaces between the subsystems. This transformation process in itself is a structuralization process and reduces the complexity of the description. In addition, the process is equivalent to the system's near decomposition proposed by Simon. If a description is small, transformation may be easily carried out by observation. However, larger-scale description requires a more appropriate method for transformation. Thus, we utilize the QMT3 to achieve the transformation, and QMT3 enables us to entirely or nearly decompose a HOQ, which can be regarded as a description of software. It facilitates discovering a structure of software and gives the definite procedures of system's near decomposition.

Table 2 Eigenvalues and eigenvectors of QMT3

Eigen vectors \ Eigen values	(1) 1.00	(2) 0.910	(3) 0.668	(4) 0.049
a	1.000	1.181	0.728	0.110
b	1.000	1.181	0.728	0.110
c	1.000	1.181	0.728	0.110
d	1.000	–0.940	0.004	1.980
e	1.000	–0.042	–1.988	–0.071
f	1.000	0.327	–1.203	–0.729
g	1.000	–1.290	0.801	–1.068
h	1.000	–1.290	0.801	–1.068
A	1.000	0.149	–1.952	–1.813
B	1.000	1.014	0.300	–0.451
C	1.000	1.238	0.891	0.500
D	1.000	–1.230	0.655	–0.230
F	1.000	–0.229	–1.300	1.782
G	1.000	–1.230	0.655	–0.236

3.2 The quantitative structuralization method

The QMT3 is devised as one of the statistical methods for categorical data analysis together with other 3 methods. This is referred to "correspondence analysis" by French statisticians. QMT3 is effective to extract the inside structure latently existing in a table.

Suppose that a table consists of two factors X and Y. The i-th element of X and the j-th element of Y have scores Xi and Yj respectively. The matrix P represents relationships between each item of X and Y. QMT3 determines these scores so that the correlation coefficient of the table takes a maximum value. Specifically, this can be carried out by solving an eigen value problem.

In this method, the new HOQ matrix is obtained by applying QMT3 to HOQ, and the structure of this new HOQ is examined, and the QMT3 is applied again. Bye repeating such a cycle, the software structure is drawn gradually. In the structuralized HOQ matrix, the items are rearranged so that the related information might gather on the main diagonal line, and the relation of the items becomes stronger as they are nearer. The lump of the related information is cut down so that the independence may become high, and the block (lump) is regarded as a subsystem. The HOQ matrix does not become a perfect diagonal procession but the place where each blocks overlap can be considered to be an "interface". Here, by using these characters, software is nearly decomposed into subsystems and the structure is analyzed. The scores of the functions and data obtained by QMT3 are utilizable for the standard of decomposition. For example, applying QMT3 to the data given by Table 1(a), we have eigenvalues and eigenvectors shown in Table 2.

It is noticed that the first eigenvector (scores) should be excluded by the definition of QMT3, and the items were rearranged in ascending order with respect to the values of the second eigenvector (scores) or others, we can easily obtain the result of Table 3 which are the same as that of Table 1(c).

The decomposition of a matrix (HOQ) as mentioned above can also be achieved by the Dulmage- Mendelsohn decomposition (DMD) for a bipartite graph[6]. Since a software description proposed here is a matrix (HOQ), it can be represented as a correspondent bipartite graph. DMD is a technique and it is effective in decomposing a bipartite graph into partial graphs, and in finding out the layered structure.

Table 3 Quantified structuralization by eigenvectors of QMT3

Second		g	h	d	e	f	a	c	b
eigenvectors		−1.290	−1.290	−0.940	−0.042	0.327	1.181	1.181	1.181
D	−1.230	○	○	○	•	•	•	•	•
F	−1.230	○	○	○	•	•	•	•	•
E	−0.229	•	•	○	○	○	•	•	•
A	0.149	•	•	•	○	○	•	•	•
B	1.014	•	•	•	•	○	○	○	○
C	1.238	•	•	•	•	•	○	○	○

Both methods yield almost the same results. The QMT3 requires more computing time to solve an eigen value problem. However, QMT3 can give more helpful information to make the description more structuralized. The DMD requires less computing time because it is a deterministic method. Therefore, it can be said that DMD is appropriate for globally examining the structure and QMT3 is useful for locally examining the structure in detail. That is, for a larger software, a detailed structure should be examined by the QMT3 after a rough structure is grasped by DMD.

3.3 The features of the quantitative structuralization method

The quantitative structuralization method put forward has the following characteristics, comparied with the present methods[1,2].

(1) Clarification of the structuralization standard

In software design, the principles or laws used as the foundation of deriving natural structure are not clarified enough. In the method of quantitative structuralization of using QMT3, the structuralization of description is performed by rearranging each item of the function and data into a large order (or small order) on the basis of the eigen values and the eigen vectors which are obtained by the quantification method of type 3. That is, it is a feature of this method to clarify the standard of structuralization and to perform quantitative structuralization.

(2) Simultaneous and concurrent structuralization

Conventionally, in the structural analysis and design of software, structuralization is performed through one of the function and data. For example, the Yourdon method tends to decompose by the function and the Jackson method etc. and tends to derive the program structure from the data structure. In this paper, the characteristic of the ways is that the function and data are decomposed at the same time. In Jackson method and PAM method (Problem Analysis Method) etc. when the data structure changes, the whole software must do it again. In these methods, when there is such a change, there is the change in a small wide by adding (or removing) some items.

(3) Thoroughness of requirement analysis

In the process of structuralization, if the requirements of users are not enough, the times of repetition will increase, HOQ is still decomposed because “diagonal line” is finally regarded as the last target. From the above, it can be used even in the condition of inadequate requirement. In the usual methods, it is very difficult to examine requirement analysis is enough or not. In this method, it is easy to discover the leak of requirements, the mistake, etc. by paying attention to the structure.

(4) Visualization of the analysis and design process

Since a software design has the tendency of brain activity of a designer, the process is unclear for the third person. Then, visualization of the analysis and design activity is important. By method of this paper, a series of HOQ are used as the visualization tool of the structuralization process. The procedure of structuralization is given clearly. Moreover, the designer can catch the state of the relation of the decomposed subsystems visually.

(5) Fusion of the bottom up formula and the top down formula

In structuralization process, the lump which the related information in HOQ Matrix gathered is cut down so that independence may become high, and a block is regarded as a subsystem. Since this is the conclusion process of a bottom up formula, the difficulty and uncertainty of functional decomposition which are seen by methods such as SASD are avoided. Moreover, since scores of the functions and data obtained by QMT3 are utilized, it can be operated for designers who have no experience about software development. By this method, the structure of software is gradually derived by applying QMT3 to HOQ Matrix repeatedly. Since this is the decomposition process of a top down formula, it also has the advantage of the top down method. That is, the modular complexity is effectively controllable.

In addition, as a range of using this method, it is aimed at the range of using this method from the requirement analysis to structure design or the functional design stage.

4 Applications

Here, we apply the description method of software to a small library management system used in Ref.[6] and analyze the structure of the software by decomposing it into subsystems. The library keeps over six thousand books and about thirty five thousand journals. The users are about twenty faculty members and three hundred students belonging to a certain department.

A procedure here writes together the concrete step about this library management system with a general step.

(1) Acquisition of the primitive information. Various requirements to the library management system are investigated and gathered from users and librarians.

(2) Conversion and arrangement of the primitive information. The primitive information is changed into the language information on a brief expression, which does not include two or more meanings. They are classified by using the KJ method[3].

(3) Extraction of functions. Functional requirements are considered and the low rank functions are extracted. At this time, it learns from the functional definition of value engineering and makes the expression “a verb” + “a

noun” as much as possible.

(4) Enumeration of data. What kinds of input or output data are required for function examination. We enumerate the relevant data to each function one by one. The software system description is made based on these functions and data.

(5) Creation of HOQ. The functions and the data corresponding to function are arranged, and the relationship is attached, then HOQ is created to describe the software.

Table 4 A structured library management software description

Function \ Data			1st	Borrower information										
			2nd	Borrower message		Day of lending	Quantity lent			Reminder		Day of return		
			3rd	Decide Borrower status	Borrower Division	Day of lending	Lending limit	Quantity lent	Number of borrowers	Stop lending	Reminder Message	Day of borrowing	Day of return	Name of borrower
1st	2nd	3rd level												
Borrower management	Borrower status register	Check borrower status		○	○									○
		Output borrower status			○									○
		Delete borrower			○									○
		Update on borrower			○									○
		Input on borrower			○									○
	Lending information	Day of lending				○								○
		Lender information			○	○								○
	Quantity of books lent	Output quantity that can be lent					○	○	○					○
		Output quantity lent					○	○	○					○
		Input name and number of borrower							○					○
	Reminder& punishment	Output message to stop lending								○	○		○	○
		Output reminder message									○		○	○
	Returning information	Print memo of return information										○	○	○
		Extend return time										○	○	○
		Compute time of return										○	○	○
		Output day of return										○	○	○

(6) Structuralization of HOQ. The first application of QMT3 to the software description indicates that the rearranged description could not be well structured. This results from the fact that a few data items are related to almost all the functions in the software system. For example, a registration code of a book and a user identification number are referred to by most of the functions. Thus, we temporarily remove these data items and try to apply QMT3 to the description. The result shows somewhat vague blocks, which seems to be subsystems of the software system. It can be considered that the vagueness comes from the complicated interfaces among functions and from the lack of functions and data. Consequently, we discover unidentified functions and data items and apply QMT3 to the revised description. This process is repeatedly done until the structure appears clear. After the temporarily removed items are put back, we can obtain the final result of the nearly decomposed description as shown in Table 4.

Applying the DMD to the bipartite graph in correspondent with the software description, almost the same structure as given by the QMT3 method can be obtained. In the example reported here, it is analyzed by a VC language program made by ourselves using the desktop type PC of 500MHz CPU. The calculation time is about 4 minutes using QMT3 and about 30 seconds using DMD. By larger-scale and complicated software, it is thought that the difference between the calculation times will become larger.

Therefore, in large-scale and complicated software, if the detailed structure about each portion is examined when it is using QMT3, and after the overall structure has been grasped when it is using DMD, the complexity of algorithm can be controlled and the efficiency can be raised.

5 Conclusions

In this paper, we introduce an idea that HOQ can be regarded as a software description tool. This idea clarifies that the principles involved in QFD have a deep relation with the usual design method. Especially, it is important to recognize that HOQ enables one to describe something shapeless or invisible from the two viewpoints such as the front and top views as they are employed in the engineering drawing.

Even if one can obtain such a software description, it cannot be always easy to understand. It follows that the reduction of the complexity existing inside the description would be required. The reduction of the complexity is equivalent to the stucturalization of the description. Then, we introduce the method called ” the Quantification Method of Type 3(QMT3)”. The quantitative structuralization method is proposed based on QMT3. The method is the important support technology of software requirement analysis and structure design process.

When using this method in an object-oriented development environment, it is necessary to take the block divided in HOQ not as a subsystem but as an object. That is, this method can be used as the quantitative object classification method.

Application by the library management system shows that the method offers a means effective in realization of a quantitative design of software structure. Since the software structure deeply relates to software quality, therefore, the method can help successfully for software quality assurance.

In order to practically utilize the method for software structure analysis and design, we consider making a support system by computer. Moreover, we want to do the future work for handling the time factor about a real time system, and for the research on software quality assurance methodology based on this method.

Acknowledgement We thank the researchers in Yamanashi, University Japan for their valuable discussions and comments.

References:

[1] Roger SP (written), Mei H (translated). Software Engineering: A Practitioner’s Approach. 5th ed., Beijing: Machine Industry Publishing Company, 2002 (in Chinese).

[2] James R (written), Hanyuda E (translated). Object Oriented Modeling and Design. Tokyo: Toppan Press, 1992.

[3] Akao Y. Introduction to Quality Deployment. Tokyo: JUSE Publishing Company, 1991.

[4] Xiong W, Shido, H. An analysis of software structure based on system’s near decomposability. Trans. of Information Processing Society of Japan, 1994,36(3):742−752.

[5] Kobayashi R. Introduction to Quantification Theory. Tokyo: JUSE Publishing Company, 1981. 63−90.

[6] Xiong W. Study on software system description and its structuralization [Ph.D. Thesis]. Japan: Yamanashi University, 1996 (in Japanese).

附中文参考文献:

[1] Roger SP(著),梅宏(译).软件工程−−实践者的研究方法.第 5 版,北京:机械工业出版社,2002.