Concurrent Design in Software Development Based on Axiomatic Design

Ruihong Zhang[1], Jianzhong Cha, Yiping Lu

Beijing Jiaotong University, Beijing, PR China.

Abstract. To shorten the lead-time of software, the design tasks should be arranged reasonably. Development process reconfiguration is the key to the concurrent design. Axiomatic design builds the functional-structure model of products by zigzag mapping among domains. The independence axiom demands that the independence of the functional requirements should be maximized. The relationship between tasks is established by analyzing the design matrix. The diagonal matrix shows that the design tasks are mutually independent, and can be concurrently processed so that the overall developing time can be greatly shortened. The triangular matrix shows that the design tasks should be processed sequentially so that the whole process can be managed effectively. By using axiomatic design to analyze the design tasks, the design process can be arranged reasonably and the lead-time can be shortened. The module-junction structure diagram shows the sequence of the software development.

Keywords. Concurrent design, axiomatic design, software development.

1 Introduction

With the rapid development of computer technologies, the application of computer is becoming more and more complicated. Large-scale and highly complicated software projects emerged continually. Software development becomes a system engineering which needs many people to participate. To shorten the developing time to the greatest extend, the tasks should be arranged reasonably. Object-oriented technology, such as Object Modeling Technique [1] and Object-Oriented Software Engineering [2], ensure the development of large-scale software in the perspective of technologies and management. But there are still many problems at the stage of development and maintenance. Such as, no method can optimize the software system [3].

[1] Postdoctoral fellow with School of mechanical, Electronic and Control Engineering, Beijing Jiaotong University, Beijing 100044, PR China; Tel: +86 10 51688175; Email: ruihong0613@yahoo.com.cn.

Concurrent design is a comparative design methodology, which enhances productivity and leads to better overall designs. The core of concurrent design is the process integration which includes two parts: multidiscipline team method and engineering of product life-cycle [4]. Concurrent means that the development activities occur in the same time and multidiscipline developing teams collaborate to carry out the design. The development process reconfiguration alters the sequential development process into concurrent development process to reduce the lead-time of product and consider all the factors of the design and development. But there is lack of the effective methods to guide the reconfiguration.

Concurrent design can be applied to the software development to optimize the software system and shorten the lead-time. Development process reconfiguration is the key to the concurrent design. Function decomposition is the basic of process reconfiguration, which shows the relationship between the design tasks. It is even as Axiomatic Design (AD) can do. AD provides a framework and criteria of function decomposition. AD can ensure the combination of software modules in a reasonable sequence and manner.

2 Concurrent Design in Software Development Based on AD

2.1 Background of AD

Suh proposed AD in 1990. AD has been considered as a theoretical foundation of designs because it can provide efficient tools and logical analyzing processes to obtain good design. AD makes it easier to integrate and analyze design requirements, solutions, and design processes.

Design in AD contains the following four domains: customer domain, functional domain, physical domain and process domain. Each domain has its corresponding design elements, namely, customer attributes (CAs), functional requirements (FRs), design parameters (DPs), and process variables (PVs). The zigzag mapping among domains is the process of product design (see Figure 1) [5]. The independence axiom, which demands maximizing the independence of the functional requirements, can be used to judge the rationality of design. The information axiom, which demands minimizing the information contents of the design, can be used to select the optimum design.

The mapping process can be expressed mathematically in terms of the characteristic vectors that define the design goals and design solutions. At a given level of the design hierarchy, the set of functional requirements that defines the specific design goals constitutes the FRs vector in functional domain. And the set of design parameters in the physical domain, which has been chosen to satisfy the FRs, constitutes the DPs vector. The relationship between the two vectors can be expressed as

$$\{FRs\} = [A]\{DPs\} \quad (1)$$

Where [A] is a matrix defined as the design matrix that characterizes the product design. Equation (1) is a design equation for product design. The design matrix is of the following form for a square matrix (i.e., the number of FRs is equal to the number of DPs):

$$[A] = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} \tag{2}$$

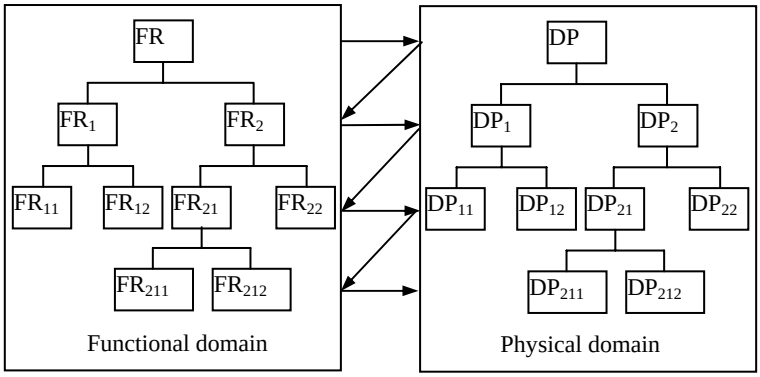

Figure 1. Zigzag mapping between FRs and DPs

When design matrix is either diagonal or triangular, the corresponding design satisfies the Independence Axiom. The former is called an uncoupled design while the latter is called a decoupled design. Any other form of design matrix is called full matrix and will result in a coupled design.

The functional-structure model of products is built by zigzag mapping among domains. The independence axiom ensures the functions' independence. The relationships between tasks are established by analyzing the design matrix. The uncoupled design shows that the design tasks are mutually independent, and can be concurrently processed so that the overall developing time can be greatly shortened. The decoupled design shows that the design tasks should be processed by sequence so that the whole process can be managed effectively.

In the same way, AD can be used to the software development. According to object-oriented technology, a system can be divided into many objects. Corresponding to AD, these objects are FRs. An object encapsulates attribute (corresponding to DPs) and method (corresponding to the relationship between FRs and DPs) (see Figure 2) [6].

In addition, "Module" has special meaning in AD. Such as,

$$\begin{Bmatrix} FR_1 \\ FR_2 \end{Bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{Bmatrix} DP_1 \\ DP_2 \end{Bmatrix} \tag{3}$$

$FR_1 = aDP_1+bDP_2 = M_1DP_1$ where, $M_1 = b(DP_2/ DP_1)+ a$

$FR_2 = cDP_1+dDP_2 = M_2DP_2$ where, $M_2 = c(DP_1/ DP_2)+ d$

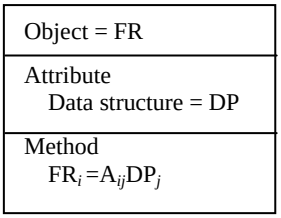

Figure 2. Graphic representation of an object

Suh [6] defined a kind of module-junction structure diagram to represent the relationships between modules in the system (see Figure 3).

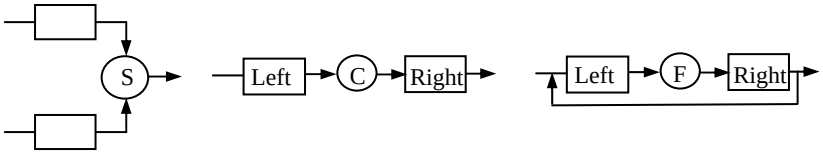

Figure 3. Graphic representations of the relationships between modules

2.2 Steps of Concurrent Design in Software Development Based on AD

The steps of concurrent design in software development based on AD are shown as follows.

Step 1: Analyzing the software using AD

In this step, designers apply AD to decompose the fundamental functions requirements and design parameters, which are generalized from the customer attributes into levels of sub-function requirements and sub-design parameters. The result is the functional-structure model and a full design matrix.

Step 2: Defining modules of the software

Software module development is a powerful method to the complicated large-scale software. It is an effective measure to control the design complexity. Function decomposing is a course that a software system is decomposed into sub-systems objectively and steadily according to the demand of the module development. The functional-structure model is a reasonable decomposing of both functions and components. In this step, designer can define the design or organize a multidiscipline development team to implement different modules and realize the software's functions.

Step 3: Reconfiguring the sequence of modules

The relationships between modules are uncoupled or decoupled on the basis of AD. The uncoupled relationships mean that these modules can be carried out simultaneously and the decoupled relationships mean that these modules must be

performed in sequence so that the effect of former modules can be considered and the iterations of design can be reduced. In this step, we can get the module-junction structure diagram that indicates the design sequence. The design tasks can be assigned according to the diagram.

3 Case Study: Software Development of AD

We are planning to develop the axiomatic design software [7]. To arrange the design process reasonably and shorten the lead-time, we use the method proposed in the previous section to analyze the design tasks.

3.1 Analyzing and Design

The software of AD should meet with the following customer attributes:

(1) Recording the design process and building documents;
(2) Providing friendly graphical user interface (GUI);
(3) Helping users make reasonable decision;
(4) Managing the whole design activity.

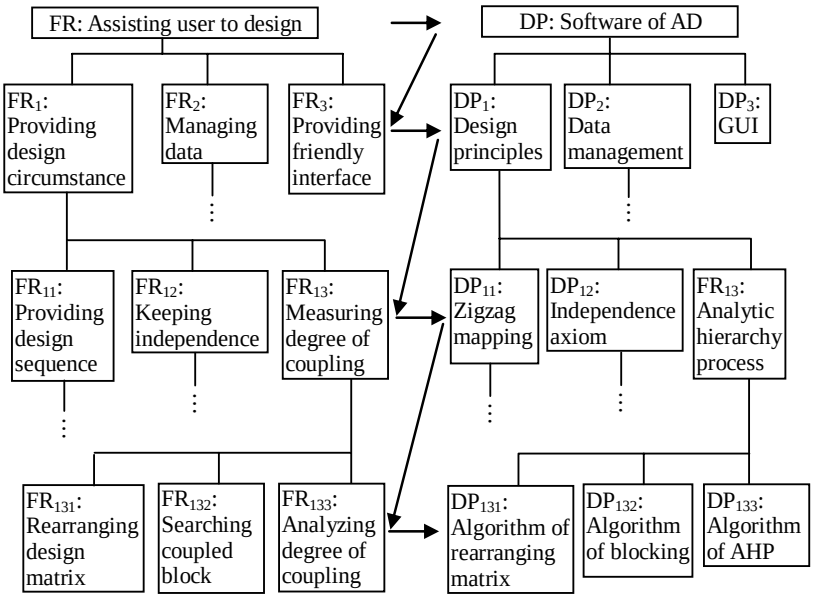

Figure 4. Functional-structure model

In this section, we just show part of the software. Figure 4 is the functional-structure model obtained by zigzag mapping and Table 1 lists the full design matrix. In full design matrix, the element "1" indicates that its corresponding FR is influenced by its corresponding DP and element "0" means no interaction. The

shading parts in Table 1 show the interaction of FRs in the same decomposing branches and the white parts show interaction of FRs in different branches.

Table 1. Full design matrix

				DP						
				1					2	3
				1	2	3				
						1	2	3		
FR	1	1		1	0	0	0	0	0	0
		2		1	1	0	0	0	0	0
		3	1	1	0	1	0	0	0	0
			2	1	0	1	1	0	0	0
			3	1	0	1	1	1	0	0
	2			0	0	1	1	1	1	0
	3			0	0	0	0	0	0	1

3.2 Defining Modules of the Software

Figure 4 illustrates the objects and attributes of the software. So there are 7 design modules of the software, that is,

(1) Zigzag mapping
(2) Independence axiom
(3) Algorithm of rearranging matrix
(4) Algorithm of blocking
(5) Algorithm of AHP
(6) Data management
(7) GUI

3.3 Reconfiguring the Sequence of Modules

Table 1 shows the design is decoupled. That is to say, these modules must be performed in some sequence so that the effect of former modules can be considered and the iterations of design can be reduced. Figure 5 is the module-junction structure diagram. “S” indicates these modules can be designed concurrently and “C” indicates these modules should be designed according to the sequence that the arrows show.

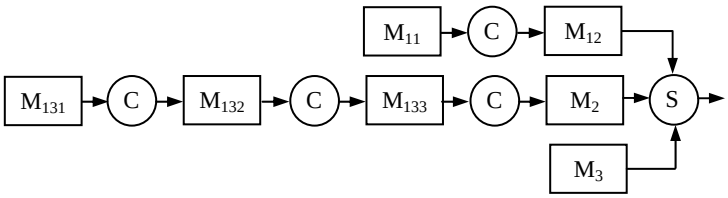

Figure 5. Module-junction structure diagram

4 Conclusion

Large-scale software projects are system engineering, which need many people to participate in. To shorten the lead-time, the design tasks should be arranged reasonably. Development process reconfiguration is the key to the concurrent design. Function decomposition is the basic of process reconfiguration, which shows the relationship between the design tasks. This is what AD can carry out. AD provides a framework and criteria of function decomposition. Axiomatic design builds the functional-structure model of products by zigzag mapping among different domains. The relationship between software modules is established by analyzing the design matrix. The uncoupled design shows that software modules are mutually independent, and can be concurrently processed so that the overall developing time can be greatly shortened. The decoupled design shows that software modules should be processed in sequence so that the whole process can be managed effectively. The module-junction structure diagram shows the sequence of the software development. The diagram can be used to guide the corresponding work.

References

[1] Rumbaugh J, Blaha M. Object-oriented modeling and design. Prentice-hall, New York, 1991.
[2] McGregor J D, Sykes D A. Object-oriented software development: engineering software for reuse. Van Nostrand Reinhold, 1992.
[3] Clapis P, Hintersteiner J D. Enhancing Object-oriented software development through axiomatic design. Proceedings of ICAD2000 First International Conference on Axiomatic Design, Cambridge, MA-June 21-23, 2000: 272-277.
[4] Xiong GL. Theory and practice of concurrent engineering. TsingHua Press, Bei Jing, 2000.
[5] Suh N P. Axiomatic design-advances and applications. Oxford University Press, New York, 2001.
[6] Suh N P, Sung-Hee Do. Axiomatic design of software systems. Annals of the CIRP, 2000; 49(1): 95-100.
[7] Zhang RH. Study on enabling technologies of axiomatic design. Ph.D. thesis, Hebei University of Technology, 2004.