Advanced Materials Research *Online: 2012-02-10*
ISSN: 1662-8985, Vols. 466-467, pp 1145-1149
doi:10.4028/www.scientific.net/AMR.466-467.1145

Refactoring of Mechanical Model Simulation Software based on Component Technology

Chen Xin [1,a], Zhao Jian [2,b], Fu Kai [1,c] and Chang Ying [2,d]

[1]School of Software, Dalian University of Technology, Dalian, China

[2]School of Automotive Engineering, Dalian University of Technology, Dalian, China

[a]cx.dlut@gmail.com, [b]zhaojian0403@163.com,

[c]kapenware@gmail.com, [d]yingc@dlut.edu.cn

Keywords: software refactoring; component technology; software reuse; simulation software

Abstract. In this paper, we refractor existing mechanical model simulation software based on component technology. The original software, which was programmed by procedure-oriented method, has high coupling, poor usability and maintainability. So we redesign and develop it with object-oriented method and component technology. After refactoring, the software, with friendly interaction interface, reasonable structure and better reusability, can provide better service for users.

Introduction

Software refactoring means a change made to the internal structure of software to make it easier to understand and cheaper to modify without changing its observable behavior [1]. It is one of the core technologies that support software reuse, through which we can improve the design of software, make the software easier to be understood, find bugs more easily and program faster.

Another core technology to support software reuse is component technology. Software component is a software entity which has independent function, clear identification, designed interface by appointing, obvious dependent relationship with context and can be assembled and deployed independently in a software system [2]. It separates software architecture clearly from system logic, constructs the cooperation of different program languages and platforms, helps to organize large-scale exploitation, and makes the system development cycle shorter and cost less. At present, the component-based software engineering (CBSE) has become a research focus in the field of software engineering.

Aim to design optimization and software reuse; we refractor an existing mechanical model simulation software based on component technology in this paper. After refactoring, the software, which has more improvement in its intelligibility, maintainability and human–computer interaction, can reduce maintenance cost and provide better service for users.

Refactoring development methods

According to different degree, refactoring development methods can be classified into code refactoring, design refactoring and component refactoring.

A. Code Refactoring

Code refactoring is the most common refactoring behavior. In the case that some programs have integrated and reasonable structure, but the codes of individual module is difficult to understand, test or maintain, we can refactor the code of this module.

For example, when faced with a long method, we can refactor it through composed method, replacing conditional dispatcher with command or moving accumulation to visitor, in order to make each function’s code in ten rows. If the code’s conditional logic is too complicated, we can refactor it through replacing conditional logic with strategy, moving embellishment to decorator, replacing state-altering conditionals with state or introducing null object [3], so to ensure the code is intelligible and maintainable.

Usually, code refactoring only concerns the details of module design and local data structure in individual module, and does not modify the overall program structure.

B. Design Refactoring

If the architecture of software is not very well, the program is difficult to modify or enhance. Therefore, we need to redesign the structure to ensure the long-term maintainability of the software, what we called design refactoring.

GoF's opus "design patterns" pointed out that, "design pattern is the goal of refactoring" [4]. Twenty-three kinds of classical design patterns proposed by this book provide valuable experience for us to refactor code. For example, if classes in a hierarchy implement a method similarly except for an object creation step, we can introduce polymorphic creation with factory method to reduce duplication resulting from a custom object creation step and express the position and process of object creation effectively. In another situation, if clients interact with two classes while one of which has a preferred interface, we can unify the interfaces with an adapter to simplify client code and reduce duplicated code [3].

C. Component Refactoring

For the purpose of software reuse, we can design software into several reusable components based on design refactoring. Complex software systems can be built through the assembled component, which can be inserted into the frame when needed [5].

It is a continuous and repeated process of component design. After the component designed and reused for period of time, we need to optimize and improve component according to the actual situation, which is component refactoring [6]. Through constructing component model based on their features, we can analyze the reuse system of variable point, and then refactor the components to reduce the cost of reuse [6]. For instance, through the decomposition of the complex business logic, we can refine business component to a smaller size, then reconstruct a large component which meets large business requirement to several smaller components each of which satisfies one business process, so to improve the reuse degree of smaller component [7].

Application

We take the refactoring process of existing mechanical model simulation software as example to describe the application of refactoring methods in practical project.

A. Mechanical Model Simulation Software

According to mechanical formula, this software calculates the object's force situation in the process of its movement, and then draws the time-displacement image, time-forces image and displacement-forces image. At present, it is mainly used for calculating the force of inertial switch with permanent magnet, including magnetic force, elastic force, inertial force and friction force [8]. The core calculation process is shown as Fig.1.

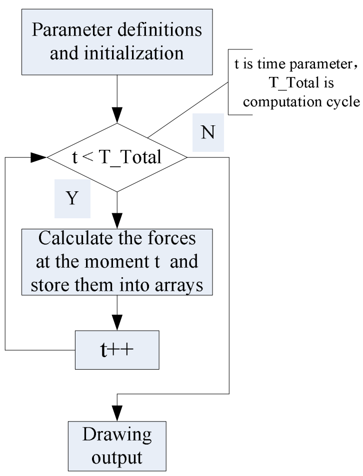

Figure 1. Core calculation process of simulation software

Though this software can still meet current requirements, there are some disadvantages.

Firstly, the system's human-computer interactivity is not strong because parameters initialization procedures were directly written into code so users can not adjust them according to actual data of object's property.

Secondly, module's coupling is high, for the calculation and drawing courses were written into the same module which resulted module's function dispersed.

Thirdly, the system was programmed by procedure-oriented method, which made the program structure tight, intelligibility poor and maintainability bad, and can not be reused effectively.

Contraposing these problems, in order to ensure the software's long-term availability and maintainability, we need refactor this software with major programming pattern.

B. Refactoring Solution

Under the precondition of original function, we redesign the software reasonably based on object-oriented method and component technology.

Build reasonable architecture to reduce system coupling

Making reference to classic MVC pattern, we divide the whole system into three parts, which are performance module, calculation module and transmission module. Among them, the performance module is used for human-computer interaction, i.e. inputting property parameters and displaying final images. Calculation module is made of several calculation components which are used for mechanical model simulation calculation. Transmission module is used for transmitting the parameters and results, which are shown out through performance module finally.

Use component technology to enhance system's reusability

The calculation module in this system is designed to meet such calculations as time-displacement calculation, time-forces calculation and displacement-forces calculation, while the forces include magnetic force, elastic force, inertial force and friction force. So we design different component for each type of calculation to ensure system's reusability. The proposed solution shown as Fig.2 illustrates the architecture for refactoring the system.

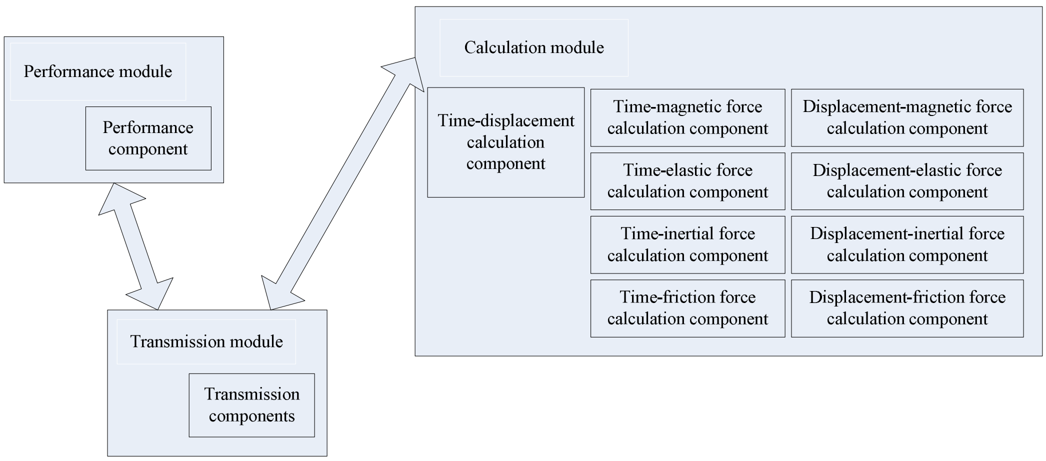

Figure 2. The architecture for refactoring the system.

Design component reasonably with design patterns

After analyzing each calculation component, we can sum up that the calculation processes are consistent though their formulas are different. That is through clock cycle, each component calculates force and displacement, and then stores the results into array, which would be outputted by transmission module when calculation ended. So we can use "Factory" pattern to design each component [4], as Fig.3 shows.

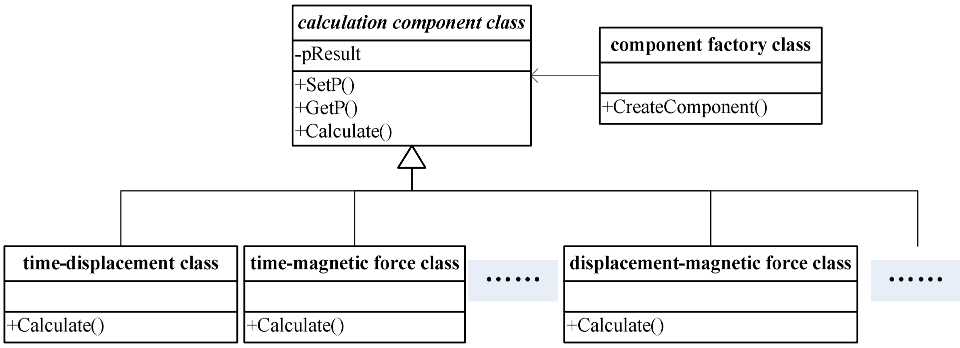

Figure 3. Design components with “Factory” pattern

C. Rrunning Effect

After refactoring, the software running effect is shown in Fig.4.

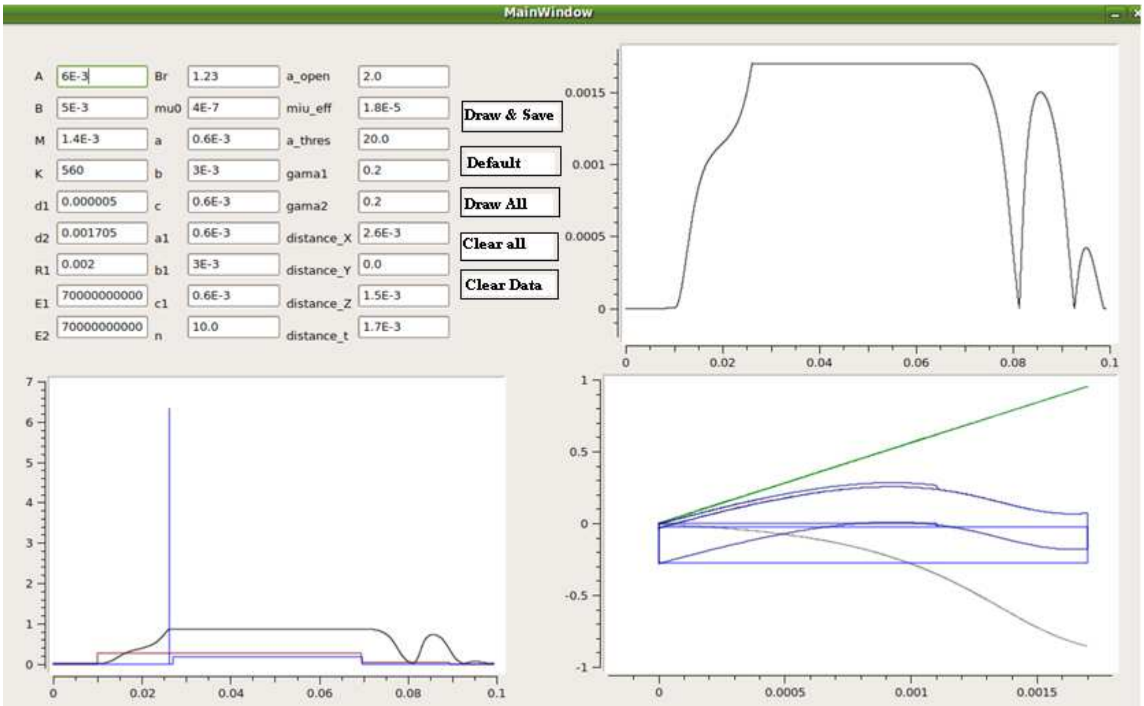

Figure 4. Running effect after refactoring

The upper-left area is for users to input the property parameters of the measuring object; upper-right one shows the time-displacement image; lower-left one shows time-forces image; and lower-right one shows displacement-forces image. Each image can be popuped separately for zooming-in, so that users can observe data information clearly.

Conclusion

Many age-old programs are still used effectively to solve the problems in the corresponding field. However, their design criterion and programming principle may be not reasonable in the development process, making their intelligibility and maintainability bad. To ensure the software can be used more long-term, it is necessary to refactor them.

Component-based software engineering has become one of research focuses in the field of software engineering at present, and software development has been changed from the original "data structure + algorithm" mode into "component development + component composition" mode. Developing software with component technology brings higher efficiency and lower cost, so it has been adopted by most software engineer.

Based on component technology, we refactor existing mechanical model simulation software. We modify its architecture to reduce the system coupling, and then improve its reusability using component technology. In component design process, we use reasonable design pattern to guarantee the system's maintainability. Result shows that the refactoring software has improvement in its human-computer interaction ability, intelligibility, maintainability and reusability etc, which shows the significance of refactoring it.

References

[1] M. Fowler, Refactoring: Improving the Design of Existing Code, Addison Wesley, 1999, pp. 46.

[2] Huang Gang, Zhang Lu, Zhou Minghui and Cao DongGang, Design and Implementation of Component-Based Software, Beijing: Tsinghua University, 2008, pp. 10(in Chinese).

[3] J. Kerievsky, Refactoring to patterns, Addison Wesley, 2004, pp. 64-65,115-116.

[4] E. Gamma, R.Helm, R. Johnson, and J. Vlissides, Design Patterns:Elements of Reusable Object-Oriented Software, Addison Wesley, 1998, pp. 2.

[5] Li Xurong, "Research on the Model of Business Component Oriented Reconfigurable Information System," Unpublished PhD Dissertation, Nanjing University of Aeronautics and Astronautics, 2002. (in Chinese)

[6] Wang ZhongJie, Xu Xiaofei and Zhan Dechen ,"Reuse Cost Optimization Oriented Component Refactoring Method"， Journal of Software, vol 16(12), Dec. 2005, pp. 2157-2165(in Chinese).

[7] C.Szyperski, G.Dominik and M.Stephan, Component Software:Beyond Object-Oriented Programming.Addison Wesley, 1998, pp. 193.

[8] Wang Hongxi and Zhao Jian, "Inertial Switch with Bistable Permanent Magnetic Structure", Nanotechnology and Precision Engineering, vol 7(2), Mar.2009, pp.137-141.