Study and Application of Patterns in Software Reuse

Zhu Zemin
Department of Computer Science and Technology
Huanggang Normal University
Huanggang, Hubei, China
Email:zzm163com@163.com

***Abstract*—This paper analyzes three different granularity levels in the software patterns, which are architectural patterns, design patterns, idioms, etc. An effective method of software development based on the software patterns reuse is given. The information publication subsystem in basic education information platform is demonstrated as an example to introduce the applications of architectural patterns, and logger module is demonstrated as an example to introduce the applications of design patterns. Application practice shows that the development based on the reuse of software patterns have many advantages. With the increasing scale of software, more complex procedures, the advantages of reusable software patterns will be more significant.**

Keywords- architectural patterns, design patterns, software reuse

I. INTRODUCTION

What is Software reuse? Software reuse is the process of implementing or updating software systems using existing software assets. The purpose of reuse is to reduce cost, time, effort, and risk, and to increase productivity, quality, performance, and interoperability [1-2]. Software reuse has become one of the hottest topics in the software community due to these potential benefits, and taken as a *silver bullet*.

Patterns reuse is a main method to implement software reuse. A large amount of patterns are accumulated and the granularity of them is becoming more and more big [3]. In software development process, reuse these patterns as much as possible is crucial.

In this paper, some sorts of patterns in software reuse are discussed. Applications of architectural patterns are demonstrated by the design of information publication subsystem, and applications of design patterns are demonstrated by implementation of logger module.

II. THE PATTERNS IN SOFTWARE REUSE

A. About software patterns

A pattern, from the French patron, is a type of theme of recurring events or objects, sometimes referred to the elements of a set. These elements repeat in a predictable manner (From Wikipedia). Pattern is the abstract, successful, reuse oriented experience. Pattern is not only technology but also the thinking, which is similar to the tactics learned by strategists. To put it simply, a software pattern is a general reusable solution to a commonly occurring problem in software design. The fact shows that software development and maintenance based on software patterns can reduce costs and time and improve reusability, maintainability, scalability, and other quality attributes.

B. Categories of software patterns

There are many methods in classification for software patterns. In view of granularity and abstract level of patterns, software patterns can be classified into architectural patterns, design patterns and idioms. [4] Idiom represents the usage of code; design pattern represents the model in the design for the modules, architectural pattern focusing on the design level reuse. General speaking, reusability of the architectural patterns is the largest than others, idiom is the smallest similarly.

1) Architectural patterns: An architectural pattern expresses a fundamental structural organization schema for a software system, which consists of subsystems, their responsibilities and interrelations. In comparison to other patterns, architectural patterns are larger in scale. There are two different "schools of thought" in the literature with respect to the nature of architectural patterns: one that uses the term "architectural pattern" and another that uses the term "architectural style". The difference of them is given in cf. [5].

In recent years, the software engineering community has had many researches on various aspects like design and evaluation methods, Architecture Description Languages. These researches have made fruitful achievements. Many architectural patterns have been proposed, such as Layers, Presentation-abstraction-control, Pipes-Filters, Three-tier, Blackboard system, Peer-to-peer, Model-View-Controller (MVC).[6-7] More and more architectural patterns are used in building not only the overall system but also the subsystem.

2) Design patterns: In this article, design patterns refer to the experience in Object-Oriented Analysis and Design (OOA / D). Cf. [8] is one of the most important References. There are 23 classic patterns are give out. They are classified into creational patterns, structural patterns and behavioral patterns. With the more emphasis are paid on the design patterns, the number of the design patterns and its applications are increased soon. Moreover, the patterns oriented specific programming language has been proposed, for example, cf. [9] summarized common design patterns in JAVA.

3) Idiom: Idiom represents the low level pattern which occurs in the programming and solves specific

978-0-7695-3728-3/09 $25.00 © 2009 IEEE
DOI 10.1109/CASE.2009.136

computational or functional problems rather than design problems. Idiom is the earliest pattern in the software development, which normally implemented by "COPY+PASTE", function and class library, data structure, algorithm, etc. Today, idioms using is a natural manner in software development.

C. *Development based on the reuse of software patterns*

Software development based on the reuse of software patterns can be divided into six steps, see in figure 1.

Requirement Analysis Phase: The essential purpose of this phase is to collect, identify and document users' requirement and to define the problem that needs to be solved.

Domain Analysis Phase: Domain analysis is "the process of identifying, collecting, organizing, and representing the relevant information in a domain, based upon the study of existing systems and their development histories, knowledge captured from domain experts, underlying theory, and emerging technology within a domain".[10] With the help of domain analyst, software system can be developed based on the reuse of asset which abstracted form similar system or relevant system in the specific domain.

Architecture Design Phase: In this phase, system is divided into some independent elements according to the functions. These elements are designed based on the specific architectural patterns and the corresponding components in elements are identified. Component library is essential. Architect usually can search a mature component in the component library, or have to develop a new component and add it to the component library.

Object-Oriented Design Phase: The purpose of this phase mainly refers to the development of components and modules using object-oriented development methodology and, as far as possible, based on a mature design patterns to achieve.

Coding and Testing Phase: To built system through code-level programming. The main emphasis is to choose suitable algorithms and data structures. At the same time, in order to ensure the quality of the system, test is necessary.

Deployment and Maintenance Phase: To deploy, deliver and maintenance the system.

Generally speaking, in the early design of software, developers and architects focus on a higher abstraction layer for reuse, and search specific assets or software architectures that can be used as possible. The domain model or architectural patterns provides a structured framework, but implementation of the elements in the architecture need to achieve based on design patterns and idioms. In general, the quality of architecture directly determines the quality of software systems, and design patterns can help build high-quality architecture. Similarly, idiom provides support for design patterns.

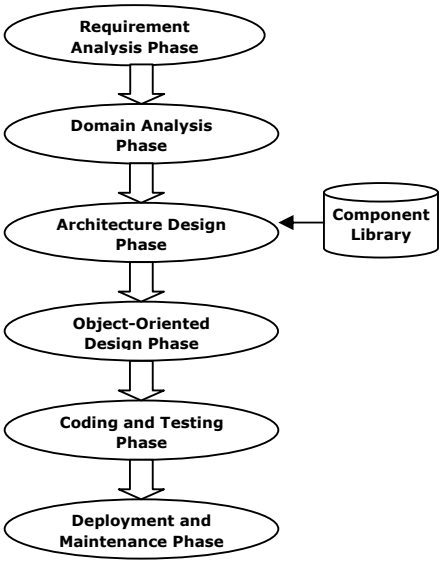

Figure 1. Steps of development based on the reuse of software patterns

III. APPLICATIONS IN BEIP

A. *Overview of basic education information platform*

The basic education information platform (BEIP) is a distributed software system designed for primary and secondary schools and administrative departments of basic education. In this section, applications of architectural patterns and design patterns in the BEIP are demonstrated through different examples.

B. *Application of architecture patterns in BEIP*

There are a lot of architectural patterns used in BEIP, for example, three-tier C/S (Client-Server) and B/S (Browser-Server) hybrid architectural patterns is adopted in the overall system and Pipes-Filters, MVC architecture etc are used in the subsystem. The information publication subsystem is designed based on Pipes-Filters. Pipes-Filters provides a solution to process the data flow system, and composed of two types of component, one is pipe component and the other is filter component. Every process is encapsulated into a filter component, and data are delivered through pipe component between the neighboring filters.

The main functions of information publication subsystem are to transform the information received from user input windows or files in different format to files in a special format which used to be saved in the database, and then to transform them to unified output format, for example HTML format with special styles. In order to integrate others' system into the BEIP system and implement SOA easily in the future, all the information is processed and delivered in XML format [11]. In the absence of Pipes-Filters when the system is implemented originally, information publication subsystem is designed as an independent module with a high

degree of coupling in the internal operation. It led to great difficulties in the development and maintenance of procedures.

The requirement document shows that the process of information publication is to transform the information from input to output and save them in the special format. The information flows in the processes. In this subsystem, three filters are proposed: input transformation, saving information in XML format, output transformation. See in figure 2.

Input transformation is to transform non-XML format information to XML format information. Saving information in XML format is to save information in XML format into database. Output transformation is to transform XML format information to other formats information, for example HTML with special CSS and logo of the company.

C. *Application of design patterns in BEIP*

Logger is a basic function module in BEIP. Its purpose is to record the important information, which helps to debug the system and monitor applications' status.[9] In logger module, an interface named Logger is defined to describe how to record information. There are two classes implemented by the Logger. One is FileLogger, which store information in log files. The other is ConsoleLogger, which display information on the screen. Logger.properties, one of the properties in the profile files, determines which class is used. Factory pattern is used in the design of two classes. Moreover, decorator pattern are used to enhance the Logger's function, such as to output the information in HTML documents, to encrypt the information.

LoggerDecorator, a default root decorator, is defined as the tools for the logger. LoggerDecorator have a reference to a Logger instance, the reference indicate a Logger object encapsulated in it. LoggerDecorator implemented the Logger interface and provide the default function of log operation. Then, the two child class of LoggerDecorator, HTMLLogger and EncryptLogger, are defined. HTMLLogger rewrite the default implementation of log operation. In new operation, the decorator transform the information to the HTML form and sent it to the Logger instance. Similarly, EncryptLogger rewrite the log operation with the specific encrypt algorithm to encrypt the information and sent it to the Logger instance.

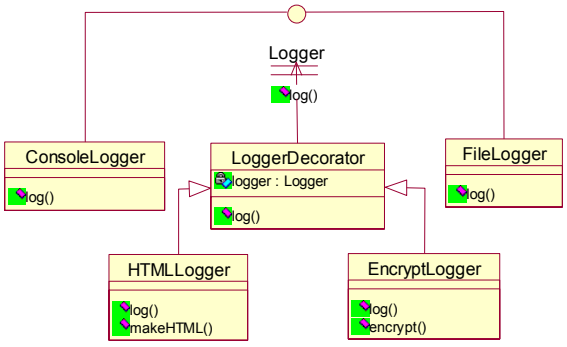

Figure 3. The class diagram of the logger function

The class diagram of the logger function is shown in Figure 3.

IV. CONCLUSION

In this paper, three different granularity levels of the software patterns, such as architectural patterns, design patterns and idioms, are analyzed. Software development method based on the reuse of software patterns is proposed. Then, several different examples demonstrate the application of architectural patterns and design patterns.

Application practice shows that the development based on the reuse of software patterns have many advantages, such as to improve software quality, reduce cost and time. With the increasing scale of software, more complex procedures, the advantages of reusable software patterns will be more significant.

ACKNOWLEDGMENT

We thank reviewers for constructive reviews and suggestions that improved the quality of this manuscript. This work was funded by the Doctor Fund of Huanggang Normal University (08cd159) and the project of Huanggang Science and Technology Bureau (08hg174).

REFERENCES

[1] Chu Wang, Qian De-Pei, "Pattern Oriented Software Development for Software Reuse", JOURNAL OF NANJING UNIVERSITY(NATURAL SCIENCES), vol.41, pp 743-748, Oct 2005.

[2] Yang Fuqing ,Mei Hong ,Li Keqin, "Software Reuse and Software Component Technology", ACTA EL ECTRONICA SINICA, Vol. 27,pp68-75,Feb 1999

[3] MARSHALL James J, DOWNS Robert R, SAMADI Shahinl, GERARD Neil S, WOLFE Robert E, "Software Reuse to Support Earth Science", Journal of Frontiers of Computer Science and Technology,Vol 03,pp 296-310, DOI:10.3778/j.issn.1673−9418.2008.03.006.

[4] Frank Buschmann, Regine Meunier, Hans Rohnert and PeterSommerlad, Pattern-Oriented Software Architecture Volume 1: A System of Patterns, Wiley, 1996.

[5] Avgeriou, Paris; Uwe Zdun (2005). "Architectural patterns revisited:a pattern language". 10th European Conference on Pattern Languages of Programs (EuroPlop 2005), Irsee, Germany, July. http://wi.wu-wien.ac.at/home/uzdun/publications/ArchPatterns.pdf.

[6] Len Bass, Paul Clements, Rick Kazman, Software Architecture in Practice, Second Edition, Addison Wesley,2003

[7] Stephen T. Albin, "The Art of Software Architecture: Design Methods and Techniques", John Wiley & Sons,2003

[8] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides. Elements of Reusable Object-Oriented Software[M]. Addison-Wesley Professional,1995.

[9] James W. Cooper, Java Design Patterns: A Tutorial, Addison-Wesley, 2003.

[10] Kang. K, Cohen. S, Novak. W, Peterson. A, "Feature-Oriented Domain Analysis (FODA) Feasibility Study", Technical Report, CMU/SEI-90-TR-021

[11] Zhu ze-min, The Information Publication Technology of Distribution Integrated Ticketing System, Computer Technology and Development,Vol.18, pp244-146, July 2008

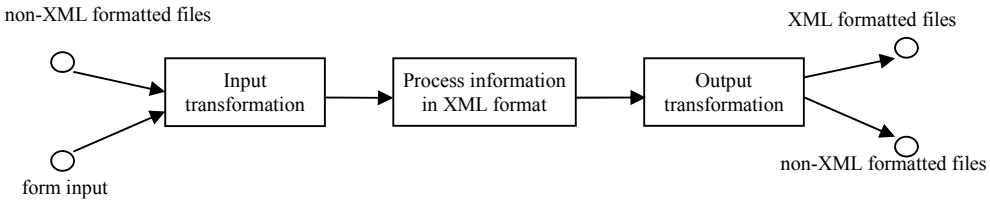

Figure 2. The architecture of the information publication sub-system