International Journal of Information Processing, 6(2), 22-30, 2012
ISSN : 0973-8215
IK International Publishing House Pvt. Ltd., New Delhi, India

Design of Extensible Software Applications with Design Patterns

Annappa B[a], Rabna Rajendran[a], K Chandrasekaran[a], K C Shet[a]

[a]Department of Computer Science and Engineering, National Institute of Technology Karnataka, Surathkal, India, Contact: annappa@ieee.org

A software application is said to be extensible if the existing version of that application can be modified with minimum impact. Modification can be either addition of new features or improving existing features without changing the current working of application. For a software application to be extensible, it has to be planned starting from the initial stages of application development. The application developer should have an idea about the possible changes expected in the software application in future. While designing the application, some provision should be made for modification in future, without affecting its functionalities or features. This paper discusses the use of suitable design patterns in software application design and how design patterns helps the developers to design extensible software applications. It also presents a case study to show the use of design patterns in the design process of extensible software applications.

Keywords : Design Pattern, Extensibility, Software Development.

1. ROLE OF DESIGN PATTERNS IN APPLICATION DEVELOPMENT

In software engineering, functional and non-functional requirements of an application are considered during the design phase of that application. The developer might come across many design issues and flaws in the design. By the time the developer finds solutions for these issues, there is a possibility that new problems might arise. This continues until a correct solution is obtained. Later, when these problems are closely analyzed, it can be found that solutions for these problems have lot of similarities. These solutions can be adopted to satisfy new requirements with or without minor changes to the existing solutions or to situations in which a similar problem occurs.
In most of the cases, developers use a solution, which is already proved to be efficient and can foresee the possible problems and take actions to avoid such a situation. That solution which is used again and again has a particular logic. These solutions for such recurring problems is called as design patterns.
A design pattern is as a solution that is proved for a software design problem. It is an object model that serves as an abstraction of the implementation model and its source code. Use of design patterns help designers in better communication, as it uses the known and understood terms in software engineering. Knowingly or unknowingly, developers follow some patterns while writing code for similar problems. Patterns are reusable as it can be applied for similar problems wherever necessary and it can avoid most of the issues that can happen at the time of implementation [1]-[3]. A pattern is not the finished code, which can be directly used in the implementation of similar problems. Use of design patterns in designing an application speed up the development process by giving hints to solve the problems quickly.
A pattern is not a method or a framework. In object-oriented programming, they show the relations and interactions between the classes and their objects. Each and every design pattern has a special purpose. Design patterns make it easy for the designer to analyze the design of application more efficiently before starting the implementation. Using patterns makes it easy for the architect or programmer

ment options. Now the application has two modules; one is for calculating the price and another one is for the payment purpose. These modules are implemented separately. The module for calculating the price remains the same where as one more module has been added to the existing version for the payment purpose. Each of these modules is made extensible using the proper design pattern. The design for payment module is given in Figure 7. This module is designed using strategy pattern. It can be seen from the design of this module that more methods can be added without affecting the existing methods. Similarly, if more functionality has to be added to this application, that can be added to it without affecting the working of the application. If the developer wants to modify or delete a module, it can also be performed without affecting the working of other modules.

During the design phase, the application can be broken down into number of modules. Each module will perform a specific task. Designer can choose proper design pattern for each and every module. If the chosen design patterns for each module are loosely coupled, then it is easy to make the entire application as extensible. But choosing an extensible pattern alone does not make the application extensible. So extensibility of an application depends on the extensibility of the design pattern as well as how much that particular pattern is suitable for the application [10], [11], [12], [13].
Consider a large application which has many functionalities; it will have different modules and each module will be using different design patterns. If these modules are extensible separately, then the whole application will be extensible provided that these separated modules are joined together properly in a loosely coupled fashion [10].

7. CONCLUSIONS

Current trends in software engineering consider extensibility as an essential feature of an application. This paper discussed about extensibility of software applications and use of design patterns for design of extensible software applications. It is clear from the illustrations that, the extensibility issue need to be addressed from the very beginning of the software development life cycle. To incorporate changes into an extensible software application, the only thing that changes is the interface but not the design of this application. By proper design of the application and by using appropriate design pattern at the initial stage, the application can be made extensible to cater to the future requirements without redoing the whole application.

REFERENCES

1. Allan Kelly. The Philosophy of Extensible Software, *http://accu.org/index.php /journals/391.*
2. Design Patterns, *http://www.dofactory.com /Patterns/Patterns .aspx.*
3. Design Patterns, *http://www.oodesign.com/.*
4. James W Cooper. The Design Patterns Java Companion, *Addison-Wesley Design Patterns Series*, 1998.
5. James Sugrue. Design Patterns Uncovered, *http://java.dzone.com/articles/design-patterns.*
6. Matthias Zenger. Evolving Software with Extensible Modules, *In Journal of Software Maintenance*, 17(5):333-362, 2005.
7. Raju Pandey, J C Browne. Support for Extensiblity and Reusability in a Concurrent Object Oriented Programming Language, *IEEE, In Proceedings of IPPS*, 1996.
8. Shriram Krishnamurthi, Matthias Felleisen. Toward a Formal Theory of Extensible Software, *In Proceedings of the 6th ACM SIGSOFT international symposium on Foundations of software engineering*, pages 88-98, 1998.
9. Steven John Metsker, William C. Wake. The Design Patterns in Java, *Addison-Wesley Professional*, 2006.
10. Ellen Agerbo, Aino Cornils. How to Preserve the Benefits of Design Patterns, *In Proceedings of Conference on Object-Oriented Programming, Systems, Languages, and Applications OOPSLA1998, Vancouver, Canada*, pages 134-143, 1998.

11. Elisabeth Freeman, Eric Freeman, Bert Bates, Kathy Sierra. Head First Design Patterns, *OReilly Media*, 2004.
12. Gou Masuda, Norihiro Sakamoto, Kazuo Ushijima. Evaluation and Analysis of Applying Design Pattern, *In Proceedings of ACM SIGSOFT International Symposium on the Foundations of Software Engineering*, 1998.
13. William B McNatt, James M Bieman. Coupling of Design Patterns, Common Practices and Their Benefits, *In Proceedings of Computer Software & Applications Conference-COMPSAC 2001*, pages 574-579, 2001.

Annappa B is a Research Scholar in the department of Computer Engineering, National Institute of Technology Karnataka, India. He has more than 16 years of experience in teaching. He holds a Masters degree in Computer Engineering from National Institute of Technology Karnataka and Bachelors degree from Mysore University, Karnataka. He is a member of IEEE, ACM, Computer Society of India, and Indian Society of Technical Education, Institution of Engineers (INDIA). His research interests include Distributed Computing, Object technology, Software Engineering, Service Oriented Architecture. He has published more than 25 papers.

Rabna Rajendran is currently working as Assistant Professor in the department of Computer Science, Travancore Engineering College, Oyoor, India. She completed her M.Tech. in Computer Science and Engineering from National Institute of Technology Karnataka, Surathkal and B.Tech from Mar Baselios College of Engineering and Technology, Trivandrum. Her research interests include Software Engineering and Distributed Computing.

K Chandrasekaran is currently working as Professor in the department of Computer Engineering, National Institute of Technology Karnataka, Surathkal, Mangalore, India. He has 22 years of experience in academic teaching and research. His Ph.D work was in the area of formal methods in communication protocols using object oriented approach. He has been an active researcher in the areas of Distributed Computing, Network Protocols, and Cyber Security. He was the Organizing Chair of ADCOM 2006, ISAHUC06, ADCOM 2008, and associated with many International events and Journals in various capacities.

K C Shet is a Professor in the department of Computer Engineering, National Institute of Technology Karnataka, India. He has more than 36 years of experience in teaching and research. He holds a Ph.D from IIT Bombay, India. He is a member of Computer Society of India and Indian Society of Technical Education. He is a Fellow of Institution of Engineers (INDIA). His research interests include Software Testing, Security Solution for Web Services, Cyber Laws, Anti spam solutions, Wireless Networks, Mobile Computing, Ad hoc Networks. He has published more than 200 papers in refereed Conference Proceedings and Journals.