Analyzing Design Patterns for Extensibility

Annappa B., Rabna Rajendran, Chandrasekaran K., and Shet K.C.

Department of Computer Science and Engineering,
National Institute of Technology Karnataka, Surathkal, India
annappa@ieee.org,rabna.rajendran@gmail.com

Abstract. A system is said to be extensible, if any changes can be made to any of the existing system functionalities and/or addition of new functionalities with minimum impact. To achieve extensibility, it has to be planned properly starting from the initial stage of the application development. Keeping in mind all the possible future changes to be made, the designer should select the proper design patterns and finish the design for the application. Once the application design is finished, it should be analyzed to make sure that the application is extensible.

Keywords: Design Analysis, Design Patterns, Extensible Application, Extensibility, Software Development.

1 Role of Design Patterns in Software Development

In software engineering, the functional and non functional requirements are taken into consideration during the design phase. During designing of the application, some unforeseen problems might arise. As the designer solves these problems, he might come across more problems. When the solutions for these problems are closely analyzed, lot of similarities can be found and these existing solutions can be adopted to satisfy new requirements with or without minor changes to the existing solutions. In such a situation, the designer can use a solution that is already proved to be a good solution, which can foresee the possible problems and take actions to avoid such a situation. That solution which is used again and again forms a particular pattern and the solution for these recurring problems are called as design pattern.

A design pattern can be described 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. Patterns help designers in better communication using the known and understood terms in software engineering. Knowingly or unknowingly programmers are following some patterns while writing code for a similar problem. Patterns are reusable as it can be applied for similar problems whenever necessary and it can avoid most of the issues that can happen at the time of implementation [1].

A pattern is not a finished code which can be directly used in the implementation of similar problems, but it gives a hint to solve a problem effectively and thereby speedup the development process. A Pattern is not a method or a framework. In object oriented programming, they show the relations and interactions

K.R. Venugopal and L.M. Patnaik (Eds.): ICIP 2011, CCIS 157, pp. 269–278, 2011.

between the classes and their objects. Using patterns makes it easy for the architect or programmer to understand it later for extension or modification. All the existing patterns can be mainly grouped under 3 categories based on how they are used as: (i) Creational, (ii) Structural, (iii) Behavioral.

Creational patterns deals with mechanisms for instantiating objects. The structural patterns deal with the composition of objects and their organization to obtain new and varied functionality. Behavioral pattern explains the interaction between different objects. In this paper, more importance is given to patterns related to extensibility. There are mainly three patterns which come under extension patterns. They are decorator, visitor and iterator [2].

2 Software Extensibility

With the emerging technologies, requirements are also changing and increasing day-by-day. The innovations in the software field forces the language designers and tool builders to enrich their products to be compatible with these innovations. Making changes to an already deployed code might not be easy all the time. It may be easy for a small application to be recompiled and redeployed. But for large software with many users, recompilation and redeployment may take a reasonable time and results in wastage of resources. Modern software need to be expanded by other developers or users to fit in the customer requirements. Software teams do not want to touch the code base for each and every change because it is error prone.

It is at this situation that, the designers and developers start thinking about extensibility and extensible architectures or designs come to the help of software developers. The important feature of extensibility is to make any change in existing system functions with minimum impact. By extensions, it means either the addition of new functionality or modification of existing functionality. In Software Engineering context it appears as a set of techniques in Software Architecture and Software Design. In Programming Languages it appears as a set of mechanisms and concepts that make it easy to extend the software. When the software is extended there will be some added features along with the functionalities that were available previously.

Most of the time, extensibility is misunderstood for reusability. Code reusability is copy/paste of the code that already exists for a similar application and it is just modifying that code to add more features or to correct some problems in the existing version. So the resultant application will be a newer or more efficient version of the existing version. But in case of extensibility, it is not reuse of the available code. Extensible design supports the iterative development principles. It allows functionality to be implemented in small steps as required.

To achieve extensibility, it has to be planned properly starting from the initial stage of the application development. The designer should have an idea of possible future requirements and how the application will have to be modified in future. For example, if it is an application for a restaurant, provision should be there to add more variety items in the menu and calculate the bill according