A Study of Architectural Design Patterns for Software Architecture

Chethana. S
Computer Science Department
Nmkrv Pu College For Women
Jayanagar 3rd Block
Bangalore-560011

Dr. G. N. Srinivasan
Professor
Dept.Of Information Science
And Engineering R.V.Vidyaneketan
Post,Mysore Road
Bangalore-560059

***Abstract*: Software architecture plays an important role in software development. Architecture brings about dissimilar approach in software development. In this paper software architecture design patterns are analysed. This paper describes the software architecture from reusable architecture patterns.**

Keywords - Software Architecture, Design, Design Patterns, Reusable Patterns.

INTRODUCTION:

The software systems life cycle is continuously altered. Hence software systems need to be modified after their initial development. According to the studies show that 50% to 70% of the total life cycle cost is spent later at the initial development[1,2]. The main reasons for software maintenance cost are- to be high, is due to the productivity of the changing existing code. Therefore it is an order of magnitude lower than developing new software or relatively independent extensions to an existing software system. Preparing a software system to have changes in the activity takes place at the development of a software system. In precise, the software architecture of the system plays an important role in buffing this. On understanding the future evolution of the software system, the software architect has various design software to prepare for the future development of new and changed requirements. One reason is that software architect has few ways or techniques for deciding the goal of high modifiability has been achieved by the set of employed design solutions. This is with respect to the maintenance cost or to avoid inflexibility in the present design. Similarly, few techniques are available for selecting the number of alternative architecture to be used for a system. One area of research would be the software architecture analysis. In SA analysis, the SA of a system is analysed by its one or more quality attributes. There are number of continued methods for software architecture analysis that exist[3]. Which includes SAAM architecture level prediction of maintenance and inflexibility assessment. Although they share few common method and similarities there are fundamental difference also.

Modifiability: In case of software system life cycle maintenance cost plays a major role. When developing software system, customers would be interested where the system is designed in the way that any future changes can be implemented, where it would decrease the maintenance cost. Maintainability is the effort required to locate and fix an error in an operational program[4].

Flexibility is the effort done to modify an operational program.

The modifiability of software system is the necessicity with which it can be modified to the change the environment, requirements or functional specification.

Software architecture: The software architecture design has received increasing amounts of attention from the last decade. There are relatively three arguments defined in software architecture. Firstly, it provides n artefact where it allows the customers to decide the design process. Secondly, it authorizes for early analysis of quality attributes. Lastly, supports communication between software architects and software engineers, since it captures the earliest and most important design decisions.

“The software architecture of a program or computer system is the structure or structures of the system, which comprises software components, the externally. Visible properties of those components, and the relationship among them. The software architecture is the highest level conception of the system in its environment.”[5].

In case of various engineering disciplines provide techniques and methods which allow one to predict quality attributes of the system before it is been built for design. It is been the operation. The availability of such technique is its high importance since, without these engineers may construct systems that foil to fulfil their quality requirements.

Prediction can be done in every stage of the development process. The drawback of this approach is that it can be applied when the code is developed.

The first steps for achieving qualities are put in the software architecture. The analysis of the software architecture is important which realize the required quality. There are different techniques an methods which allow one to predict a system’s quality Based on the software architecture analysis as shown in fig[6].

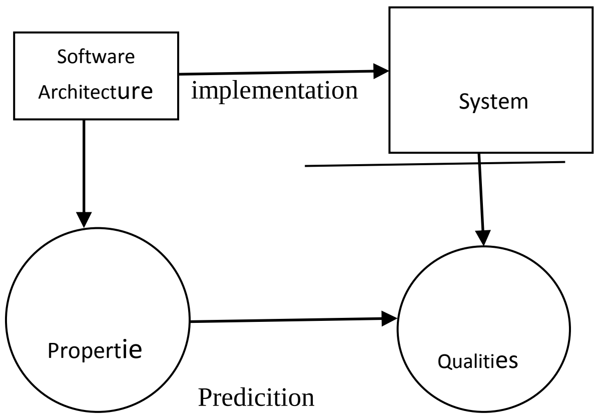

fig 6.

Software architecture analysis of modifiability in five steps.

1. Set goal determine the aim of the analysis.
2. Describe software architecture –give a description of the relevant parts of the software architecture.
3. Elicit scenarios: find the set of relevant scenarios.
4. Evaluate scenario determine the effect of the set of scenarios.
5. Interpret the result – draw conclusion from the analysis result[4].

Description of the software architecture:

Software architecture analysis needs information about the architecture. At the time of architecture designs. The software architecture evolves: that it is extended gradually with architectural decisions made overtime. As a result the detailed and amount of information available about the software architecture is dependent at point to point at which analysis can be performed. In case of software architecture analysis we use the information that is a variable. Therefore at the later stage analysis is performed in detail, which is available to the analyst.

For instance the 4+1 view model (8) consists of following design.

The logical view – which is t he object model of the design[7,8].

The process view, which describes the mapping of the software out to hardware and reflects its distributed aspect.

The development view- which describes the static organization of the software in its development environment.

The physical view – it describes the mapping of the software out to hardware and reflects its distributed aspect.

The scenario view – or +1 view, it illustrates the way the architecture satisfied the requirements and defines the relations among the other view.

Design patterns: In software engineering design pattern is a commonly occurring problem in software design. A design pattern is not finished, but it can be directly transformed to code. It is a description or template for how to solve a problem that can be used in many different situations. Design patterns are used to speed the development process. Effective software design requires certain issues which may not be visible until the implementation[9].

Reusable design patterns help to prevent some of the issues which can cause major problems. By this it improves code readability for the coders and architects familiar with the patterns.

Collection patterns: Deals with groups or collection of objects. The details of how to compose classes and objects to form larger structures. To look for a more efficient way of designing a class therefore for instance it will not carry any duplicate data.

Behavioural patterns: It provides details of assigning responsibilities between different objects. Explains the communication mechanism between objects. Classify different mechanism for choosing various algorithms by different objects at runtime.

Structural patterns: Describes the responsibilities to other objects. It results in a layered architecture of components with low degree of coupling.

Due to the incompatible interface, the object is not accessible to the other by normal means.

Provides ways to structure an aggregate object so that it is created in full and to reclaim system resources in a better manner.

Creational patterns: The most common task in an object oriented application, where the objects are created.

Support a uniform simple and controlled mechanism to create objects.

It permits encapsulation of the details about the classes that are instantiated and how the instances are created. It supports these interfaces, which reduces coupling[10].

Conclusion: This paper has described an approach of software architecture, design, and design patterns have been discussed. This research can be extended further by comparison of the different deign patterns.

ACKNOWLEDGEMENTS:

I am grateful to my guide Dr. Srinivasan ,Sri Rashmi for their support in bringing out this paper successfully. I would especially thank Dr. Suchithra madam also for giving her time and input.

REFERENCES:

1. R.Harrison, ‘Maintenance Giant Sleeps Undistributed in Federal Data Centers,’ Computer World, March 9,1987.
2. B.P.Lientz and E.B.Swanson. Software Maintenence Management, Reading, MA:
AddisonWesley 1980.
3. L.Bass, P.Clements and R.Kazman, Software Architecture in Practice,Reading, MA:
4. Addison Wesley 1998.
5. ISO/IEC, Information technology- Software product quality- part 1: Quality model,
ISO/IEC FDIS 9126-2000(E)
6. Nico Lassing ,Jan Bosch. ‘Analyzing Software Architecture for Modifiability’
Dept. Computer Science Engineering,Sweden.
7. S.A.Bohner, ‘Software Change Impact Analysis for Design Evolution’, In Proceedings of 8th International Conference on Maintenance and Re-engineering, Los Alamitos.
8. L.Briand,E.Arisholm,S.Counsell,F.Houdek and P.Thevenod-Foss,’Emprical studies of Object Oriented Artifacts,Methods and Processess:Emprical Software Engineering,1999.
9. www.Design.html.
10. Partha Kuchana, ‘Software Architecture Design Patterns in Java’.