CHAPTER 1

INTRODUCTION

Software Engineering is the establishment of sound engineering principles for systematic, disciplined way of developing and maintaining quality software. Preserving software quality is considered as one of the primary activities in software development process, which is to be performed during the development and after the release of the software product. Software quality preservation includes maintenance of correct, adaptive and perfect software. The categories of maintenance process have been stated by Pigoski (2001) according to ISO/IEC standard on software maintenance as corrective, adaptive and perfective task. Corrective maintenance is the process of modifying the software to resolve the problems discovered after the release of the software. Adaptive maintenance makes the software accommodate changes in the user needs and/or in the environment.

Modifying the software to enhance its functionality and performance is referred to as software evolution or perfective maintenance process. By observing the happenings in the real world systems, it has been found that changes are inevitable and are to be considered at all times. Hence software products that automate the real world systems should have the capability of adapting to those changes, which implies that adaptive maintenance is an indispensable task in a software life cycle. Adaptability becomes the key feature of a software system and Kalareh (2012) defined it as
the ability to reconfigure its elements or change its behavior based on the changes in the user needs and/or in the environment. The demand for adaptive systems appears in many application domains ranging from crisis management such as disaster and power management to entertainment and business applications such as interactive gaming, tourist guiding and business collaborations. The software system with the adaptability feature is named as adaptable or adaptive system.

The development of the adaptable system should focus on adapting changes in the functional or non-functional requirements; managing the evolution of the requirements at the design time or runtime and managing even unanticipated requirements. Hence, the adaptive maintenance process to be defined to accomplish the above features. The operations determined to be incorporated into adaptive maintenance process are:

- specifying the level of anticipation or un-anticipation of the changes
- determining the scope of adaptation, which refers to the extent to which the adaptation of changes are spread over the system
- determining the mode of achieving the adaptation, which ranges from parametric, that is fine tuning of the parameters in the component(s) to compositional, which implies replacement or insertion of functional components
- specifying the occurrence of adaptation in software engineering process, namely in the requirements, architecture, design, implementation, deployment and after deployment level.
1.1 MOTIVATION

Nowadays, there is an increasing demand for a software that should cope with variability, get deployed on large diversity computing environment and operate in different execution environments due to growing range of domains such as: ubiquitous systems, automotive systems and web services. Hence, systems that can automatically adapt to these changes have become a current need. Constructing and executing such adaptive systems is a complex task due to several challenges. These systems are deployed on distributed heterogeneous platforms and they are composed of components with variable dependent configurations. Also, these systems are defined with the several implementation variants, which lead to exponential growth in the number of possible system configurations. Developing adaptive software that can resolve the above issues using the traditional software development techniques has become difficult due to the following limitations inherent in them.

- Difficulty in predicting the future user needs and changes in the user requirements.

- Impossibility of providing a single, best and optimal configuration of the software.

- Difficulty in predicting the environment changes in ubiquitous environment.

Hence, a promising solution is to define an adaptive maintenance process with the ability to perform adaptation at run-time, which is referred to as a dynamic adaptation. The dynamic adaptation may create different variants of the application, comprising components and services that provide
the functionalities with a changed dimension. The complexity and the challenges of providing adaptation solution depend on the time of anticipating the adaptation. The adaptations known during the design time are the easiest to achieve, whereas the adaptations that are to be handled while running an application is a very challenging task.

Dynamic adaptation challenges include the adaptation of the coarse granularity changes, medium granularity changes and fine granularity changes. In general, adaptable system is designed with internal and external adaptation approaches (Salehie and Tahvildari 2009). The structure of external adaptation approach based system is constituted with the Adaptable Component that provides to the system the adaptability feature by choosing the techniques or paradigm as stated in Adaptation Policy and by using Adaptation Engine that implements the changes based on the paradigm. Its structure is shown in Figure 1.1. There are many techniques or paradigms associated with developing the adaptable system that can accommodate the dynamic changes. The following are the core techniques used to perform dynamic adaptation:

- Adaptation of the coarse granularity changes is done using dynamic reconfiguration of the components in the architecture level (Grace et al 2008). The reconfiguration techniques aim at adjusting internal or global parameters in order to respond to the changes in the environment (Costa-Soria 2011).

- Compositional approach is used to adapt the medium granularity changes. In this approach, adaptation is defined through the insertion or replacement of components or services, and it relies on the binding or unbinding mechanism (Costa-Soria 2011).
• Parameterized approach is used to adapt the fine granularity changes. Here adaptation is achieved through the adjustment or fine tuning of predefined parametric values of software entities such as components and services (Costa-Soria, 2011).

• Dynamically linking and unlinking approach allows adding new functionality or replacing the existing components by new ones through the binding manager, which is built on the component model (Fox and Clarke 2009). It is also used to achieve the dynamic adaptation through the separation of functionality from adaptation (Dowling et al 1999).

• Generic interceptor technique intercepts the messages between components in order to add the behavior for adapting to the changes without affecting the behavior of the components (Fox and Clarke 2009).

• Aspect Oriented Programming (AOP) is described by Kiczales et al (1997) as the software engineering approach, which supports adaptability through handling the problems of tangled code. The code for implementing the features such as security, persistence, logging and monitoring are referred as tangled and are represented using separation of concerns. The code should implement each of these features as an advice that is to be woven into the basic code of the component functions. The code location named as joint-points into which the advice is to be attached is expressed in the point-cuts.

• Reflection is the capability of a system to reason about itself and adapts the changes in its behavior using this self-obtained information. This is realized through the causal connection between the base-level (functional aspect of the system to be
adapted) and meta-level (adaptation aspect of the system) (Maes 1987).

- Design patterns namely Wrapper (Adapter), Proxy and Strategy and Architectural patterns namely Microkernel, Reflection and Interception are also used to achieve adaptability to the changes in the system as stated by (Salehie and Tahvildari 2009). Strategy pattern allows the separation of functional implementation from adaptation code in achieving dynamic adaptation (Dowling et al 1999).

![Diagram of Adaptable Component Structure](image)

**Figure 1.1 Adaptable component structure**
The adaptation techniques mentioned above have the limitations of:

- Inability to manage the unanticipated changes
- Altering the structure of the system through reconfiguration that involves insertion or replacement of component, fine-tuning or adjusting the parameters and inclusion of new behavior. In this approach the adaptation process consumes more effort and time
- Enabling the adaptability of only non-functional services using cross-cutting concern policy
- Difficulty in maintaining consistent meta-information of the components
- Not providing generic adaptability solution at the architectural level

Hence, this research work focuses on deriving a solution for overcoming these limitations. Defining an adaptation process for overcoming these limitations will enable to create an adaptable system that can manage the variations in behavior and/or in operating environment of the system.

1.2 PROBLEM DEFINITION

The existing adaptability solutions allowed the system to adapt to the changes in the user needs or/and in the execution environment through modifying the internal structure of the component, reconfiguring the components, altering the compositional structure of the components and changing the behavior of the component and are implemented using
reflection, interception and aspect-oriented programming techniques. Also, these solutions are suffered from the limitations of reflection process, interception mechanism and aspect binding process. Hence, the objective of this work is defined as to provide a solution to adapt to the dynamic changes at the architecture level by addressing the major constraints of the existing dynamic adaptation techniques. In order to achieve this, the novel techniques for modifying the software system functions or components that meet the user requirements are to be considered for adapting the changes in those requirements. Such an adaptable solution should have the features of,

- adapting unanticipated dynamic changes in the user requirements without reconfiguration
- dynamic discovery of components based on the changes in the requirements
- changing the services or components without affecting the application model.

Hence, this research focuses on proposing an adaptable solution with the above mentioned features. The proposal of this research is formulated as maximizing the adaptation of dynamic changes without modifying and reconfiguring the system. Let the software system be denoted as ‘S’, structural model of a system as „Model(S)“, changes in the requirements as ‘δC’, alteration in the model of a system as „Alt(Model(S))“ and reconfiguration of a system as „Reconfig(S)“. The problem to be addressed is expressed as follows:

$$\begin{align*}
\text{Maximize} & \quad \text{Adapt(S)} = S + \delta C \\
\text{Subject to} & \quad \text{Alt (Model(S))} = \text{NULL} \\
& \quad \text{Reconfig (S)} = \text{NULL}
\end{align*}$$

(1.1) (1.2) (1.3)
Since software architecture is the backbone of constructing quality software systems and adaptability is considered as one of the quality attributes, the above stated problem has to be addressed at the architecture level (Bass et al 2003). Hence, providing the adaptability solution at architectural level is considered as an issue that is to be addressed in this research. The proposed solutions are to be validated in order to ensure its applicability. Hence, deriving the metrics to evaluate the adaptability solutions are also considered in the present work.

1.3 CONTRIBUTIONS

The major challenges of developing the adaptable system that accommodates dynamic changes in the user requirements with respect to the criteria that are stated in the problem definition are determining appropriate adaptation techniques to adapt to dynamic changes and deriving a mechanism for reducing reconfiguration to handle the explosion in the variants of technologies, development and environmental platforms. Aspect-oriented modeling techniques tackle the issue of explosion of variants through encapsulating distinct variation points into aspects. In the present work, the separation of concerns principle stated in the aspect-oriented programming is used to separate the process of handling the changes in a system, introspection principle is used to observe the structure of a system component and wrapper is used for dynamic invocation of the functions. Integration of these principles allows binding the changes with the wrapper of the functions associated with the changes without altering and reconfiguring the structure of a system. The brief discussion about the techniques proposed for defining adaptation policy and adaptation engine components of a dynamic adaptable system has been stated as follows.
Proposed Dynamic Aspect-Weaving Adaptation Approach: This approach includes the evolution of dynamic change aspect, dynamic structural introspection model and dynamic wrapping technique using the principles of aspect-oriented programming, reflection technique, and adapter pattern respectively, for adapting to the changes in requirements at run-time.

- Dynamic Change Aspect for Adaptability: The Separation of Concerns principle is restated as to separate the implementation of dynamic changes in the user requirements or functionalities of a system. Here, an aspect is defined as a unit of dynamic changes in the user requirements, which is to be integrated with the component functions associated with the changes.

- Dynamic Structural Introspection: Introspection is a type of reflection techniques to observe structural details of the components in a system. While processing the changes in the requirements, the components associated with the changes are introspected and the information about interfaces, classes, methods and properties of those components are extracted dynamically.

- Dynamic Function Invocation: The wrappers for the functions associated with the dynamic changes are generated by applying adapter pattern and meta-model information. It enables dynamic invocation of the functions and allows modifying the wrappers to incorporate the changes rather than modifying the original functions. This discards the need for modifying the existing system components and reconfiguring them.
**Proposed Adaptable Middleware:** This middleware provides the solution for the adaptability at the architectural level by providing the solution framework for the above approaches. Thus, it reduces the effort expended on the reconfiguration. The components of the proposed middleware are Aspect Adaptation Manager Component, Change Request Parser Component, Aspect Generator Component, Aspect Weaver Component, Component Function Wrapper and Component Meta Model. Aspect Adaptation Manager Component manages the processes of parsing the dynamic change request, generating the aspect with advice to implement the proposed change and dynamically weaving the changes with the corresponding function using the Change Request Parser, Aspect Generator and Aspect Weaver components along with Meta Component Model and Component Function Wrapper components.

This middleware model allows the system to reflect the dynamic changes on the associated components through aspects without altering the structure of the components. For evaluating the adaptability of this middleware, a new adaptability metric has been proposed using the principles of coupling. Here, the coupling is defined as a conceptual coupling, which represents the semantic association between the aspects that are used to represent dynamic changes and the components that are associated with the dynamic changes at the architecture level. Also, the metric named conceptual coupling between classes has been proposed to measure the adaptability of an object-oriented system at the design level. The functional adaptation time metric has been proposed to measure the adaptability of a system, which uses compositional approach as well as the proposed adaptable middleware to adapt to dynamic changes.

This proposed solution is mainly used in software industries during software evolution for incorporating changes into a software system. The
integration of a system with the proposed adaptable middleware enables to address the issues of adaptability in the architecture level. Since the architectural level solutions are more abstract, not specifying the implementation details and reusability, the proposed adaptable middleware also possesses these properties. This feature allows the developers to integrate this middleware with any software system for adapting to dynamic changes. To accomplish this, developers should have the knowledge about this middleware. This report is written with the information which will enrich the developer with the knowledge to access and enhance this middleware. The information required to envision the feature of the middleware is provided in the introduction chapter of this report. The organization of the remaining chapters of this report has been covered in the following discussion.

Chapter 2 – Background Concepts and Related Works: The concepts used in this work are aspect-oriented programming and reflection. Aspect-oriented programming allows defining cross-cutting concern using aspect, which is comprises point cut, join point and advice. Reflection is used to inspect the structure and the behavior of the application objects, to modify the objects and to reify the modifications. The need for encapsulating the adaptability solution in the middleware and the various middleware models designed using aspect-oriented approach, reflection and augmentation of reflection with aspect-oriented principles has been discussed from the perspective of understanding the proposed solution.

Chapter 3 – Proposed Dynamic Aspect Weaving Adaptation Approach: The principles suggested for the proposed dynamic aspect weaving methodology are redefined separation of concerns principle, dynamic structural introspection and dynamic function invocation. The underlying basic concepts of these principles and the refinements made on them have been explained in the separate sections of this chapter. The demonstration of
the proposed approaches on a banking transaction system has also been discussed here.

Chapter 4 – Proposed Adaptable Middleware Model: The structure of the middleware by describing the functions of each component and the realization of the component functions using classes and their structure have been specified to understand the role of middleware in adapting the dynamic changes. The formal analysis procedure used to evaluate the architecture against the objective of achieving dynamic adaptability has also been discussed in this chapter. Incorporating adaptability feature into banking transaction system using adaptable middleware model has also been demonstrated here.

Chapter 5 – Proposed Adaptability Metrics: Conceptual Binding between Classes (CBC) metric is determined using coupling principle for evaluating the adaptability efficiency of an object-oriented system that accommodates the changes through abstraction and inheritance features. Conceptual Binding between Aspect and Classes (CBAC) metric has been proposed to evaluate the adaptability of the system that adapts to changes using the proposed adaptable middleware. The formulation of these metrics and the evaluation of CBAC metric using the general metric properties have been specified to understand the applicability of these metrics. The adaptability efficiency of a banking system is measured using CBC and CBAC metrics and comparison of these values have been described in this chapter.