CHAPTER 6

CONCLUSION AND SCOPE FOR FUTURE WORK

6.1 CONCLUSION

The focus of this work was on optimizing the software maintenance task and improving the quality of software which are considered as the major driving forces of software development process in industries. Optimization of software maintenance was achieved through a well designed software evolution process and the software quality was improved by incorporating the adaptability feature. The software evolution process was designed with an architecture level adaptability model for performing adaptive maintenance task at run-time and for making the software system possess the quality of adaptability. An adaptable middleware model was defined using the proposed dynamic aspect weaving approach, which was designed with the novel features of dynamic change aspect, dynamic structural introspection and dynamic function invocation.

The software evolution process was targeted towards the accommodation of changes in the requirements of a system. The dynamic change aspect unit was designed to separate the representation of changes in the requirements, which enabled the implementation of the changes without moving the system into non-productive state. Enabling a software system to incorporate the changes without affecting its structure was achieved through dynamic structural introspection and dynamic function invocation techniques. These novel techniques were used in the design of adaptable middleware
model (AMM) to incorporate the dynamic adaptability feature at architectural level of a software system. The components of AMM, which implemented the proposed novel techniques, were Aspect Generator, Component Meta Model and Component Function Wrapper. Based on the requirements change specification Aspect Generator created a Change Request Aspect class to implement the change in the advice part and specified the wrapper method as the location into which the advice has to be weaved. Component Meta Model, based on the dynamic change request, extracted the structure details of the components associated with the changes. Component Function Wrapper generated the wrapper to invoke the component functions associated with the change.

The proposed adaptable middleware allowed the system to incorporate the changes by expending an effort for defining an advice to implement the changes. The middleware offers the services for the extraction of component functions related to the changes, generation of the function wrapper and binding the change implementation with the wrapper. The middleware also reduces the effort expended towards the composition of classes/components/aspects and reconfiguration of the classes/components through modifying class/component functions and their parametric values either directly on the application objects or on the meta-object that will get reflected on the application (base) objects. The compositional, reconfiguration and parameterized approaches allowed handling anticipated changes in a static way. To enhance the dynamic adaptability feature, the dynamic weaving was achieved through binding aspect with wrapper generated at runtime that reduces the complexity involved in performing dynamic weaving through run-time byte code binding. The proposed adaptability solution was proved as simple and optimal solution through the evaluation done using conceptual binding between aspect and class (CBAC) and functional change adaptation time (FCAT) metrics.
The proposed middleware model was described both by using the schema constituted with the declaration of data and function of components and also by the set of rules to define them. The formal description provided the set of specification for implementing the components of the proposed middleware model in any platform. In the same way, the description of the static and dynamic structures of the proposed middleware model using unified modeling language allowed to easily map these platform independent specifications into platform specific way.

Modularization was considered as a unique way of making a software system flexible to accommodate the changes. Since adaptability was identified as directly associated with modularization, this work used software metrics to measure the modularization as the base for measuring the adaptability efficiency of a system. In general, the change impact analysis on a software system was performed based on the dependency existing between the functions and data involved in a system. Coupling between the objects (CBO) was identified as one of the software metrics used to measure the dependency between objects, which was considered as the basic metric used to define the metrics for measuring the adaptability of a system. Since the complexity of incorporating the dynamic changes into the methods of the classes is based on the association of that method with other classes in a system. The proposed metric was the conceptual coupling between classes (CBC) using coupling principle to measure the adaptability efficiency of an object-oriented system. This metric implied that adaptability is reduced with increasing the association between classes.

The adaptability of a software system that implements adaptability using aspect-oriented approach was measured using the conceptual binding between aspect and class (CBAC) metric proposed in this work. This metric was used to specify the dependency that exists between the aspects that
implement dynamic changes and the methods of the classes associated with the changes. This metric was designed using the appropriate coupling measure and the metric properties were evaluated to confirm its suitability to the adaptability context.

The adaptability feature was incorporated into a banking transaction system using compositional approach and the proposed middleware; and its adaptability efficiency was evaluated using CBC and CBAC metrics. The observed results showed that the proposed middleware model was most appropriate for adapting the changes associated with multiple functions instead of using compositional approach for adapting changes in an object-oriented system.

6.2 SCOPE FOR FUTURE WORK

Nowadays, software developers are involved in creating the software solutions as services that are to be deployed in web or in cloud environment, which will allow the clients to avail of them with less investment and for easy customization. Hence, the proposed adaptability solutions can be defined as web or cloud services. There is a need to enhance the existing distributed software services so as to include business, communication, transaction, security, interoperability and discovery and location awareness with the capability of accommodating these variations in an operating environment or in user needs. Hence, the proposed adaptability middleware solution can also be demonstrated on these services to make them adapt dynamic changes.

The mobile applications have to manage the challenges such as mobility of the user and device and their locations in which they are running, resource-constraints devices in the mobile environment, heterogeneous platforms, distributed infrastructures, communication protocols and resource
discovery protocols. Applying the proposed middleware solution to create dynamically adaptable mobile application requires coping with the above mentioned challenges. Hence, the proposed adaptable middleware has to provide the resource awareness, location awareness and environment specific protocol selection services for enabling mobile applications to accommodate the variations in the user needs, resource constraints, infrastructures and protocols.

Dynamic adaptability has to accommodate the unanticipated changes at run-time. The proposed adaptable middleware automated the process of adapting the unanticipated changes except the automation of the generation of implementation code to represent the changes. This can be performed by creating an intelligent tool to generate the code according to the change specification. This can be accomplished by creating a knowledge base with the meta-information of the requirements realized in a software system and also a set of rules to show the association between the requirements. Using this knowledge base, an intelligent implementation engine can be defined with the functions of generating design for the change implementation with the solution rules, and mapping it to appropriate language according to the underlying platform.