CHAPTER 6

SUMMARY, CONCLUSIONS AND FUTURE ENHANCEMENTS

6.1 INTRODUCTION

The earlier chapters in this thesis established the research work and experimentally proved results. Based on the visions, this chapter makes conclusions and provides guidelines for future work. It also throws light into the summary of findings in the research.

6.2 SUMMARY

The aim of the thesis is to propose architecture and implement it for dynamic self-adaptive software architecture. It is needed to overcome drawbacks of existing framework with the help of data mining for QoS analysis and reusing history to make well informed decisions while making self-adaptation. Towards this end extensive literature review is made and found that the problem improving self-adaption architecture is open for further research. It is also found that Rainbow kind of frameworks needed further enhancements in terms of QoS analysis and gaining intelligence from historical data for optimization of the framework. A motivating scenario is to have self-adaptive software architecture that is dynamic in nature with provision for understanding runtime contexts and adapting to it without manual changes in the application under execution.

A dynamic self-adaptive architecture by name Enhanced Self Adaptive Dynamic Software Architecture (ESADSA) is proposed and implemented. ESADSA is inspired by Rainbow framework. It incorporates modules such as QoS analyzer and Knowledge Miner with two data mining algorithms for enhancing capabilities of the architecture. ESADSA decouples self-adaptation from target system by preserving cohesion of target system with loosely coupled interaction. Afterwards, the framework is further enhanced with another algorithm known as Heuristic Self Adaptation (HSA) which will optimize decision making with heuristics at runtime. The proposed framework ESADSA is evaluated with different real world case studies such as New.com and AWS cloud based web application to realize
self-adaption. The empirical results are compared with that of Rainbow and found that the proposed architecture has more utility in self-adaptation.

6.3 CONCLUSIONS
The study made in this thesis resulted in many insights related to dynamic self-adaptive software architecture. The following are the conclusions made.

- A “dynamic self-adaptive software architecture” named “Enhanced Self-Adaptive Dynamic Software Architecture” (ESADSA) is proposed and implemented. The architecture is inspired by Rainbow but it has two additional modules known as QoS analyzer module and History miner module. These two modules are implemented with baseline approach and the architecture is evaluated with a case study known as New.com which is a distributed system. It has multiple application servers and web servers. It operates in two modes such as textual and graphical. It switches from one mode to another as part of self-adaptation. The architecture is analyzed and found that zero false positives in self-adaptation. Other observations include quality of service throughput, latency of random requests, and adaptation time against number of requests. A model application is built to demonstrate proof of the concept. The application has admin panel that can monitor self-adaptation process based on the number of requests. HTTP request generator program is used to evaluate the application and the self-adaptive architecture showed effective response to the runtime needs of the users. When the number of requests is beyond the capacity of all servers, it switched to text model and it remains in graphical model for rendering news when the servers are capable of giving timely response to the requests provided.

- ESADSA architecture is further enhanced two algorithms corresponding to the modules such as QoS analyzer and Knowledge miner. The algorithms are known as QoS-Aware Analysis (QAA) algorithm and Knowledge Mining (KM) algorithm. The former takes the QoS parameters required by the software under test and takes the stock of resource available. After analyzing the resources and parameters, it makes set of decisions that help in further improvement in quality adaptations. The latter takes the QoS parameters required by the software under test and knowledge
base then mine the database for discovering optimal decisions. A self-adaptation case study with web based distributed computing application deployed in Amazon Web Service (AWS) cloud is considered to evaluate ESADSA. The application makes use of thin clients. The response time is to be maintained as per the service level agreements (SLAs) provided to end users. This is the key factor that needs self-adaptation. Amazon Relational Database Service (RDS) based MySQL database is preferred. Initially 10 web servers, 10 application servers and 5 database servers are used by the distributed application. Base on the response time adaptation framework determines the usage of additional servers. A threshold for response time is taken. This is maintained consistency. When response time is deteriorated, the adaptation framework takes necessary steps to utility more servers in order to ensure that the response time is complying with SLAs. Available bandwidth and server load affect response time. Based on the analysis of these parameters decisions are made dynamically for self-adaptation. The performance of the software under execution is evaluated with ESADSA and without self-adaptation. The results revealed that the ESADSA with proposed algorithms in place outperformed the one without using it.

- The ESADSA is further enhanced with another important algorithm that makes use of heuristics from time to time to have more sophisticated analysis in tandem with the QAA and KM algorithms to ensure that the runtime QoS needs and contexts are taken care of with further optimizations. It is known as Heuristic Self Adaptation (HSA) algorithm. The algorithm takes care of QoS needs and history availability to update a model containing heuristics to leverage decision making process. The SVM based heuristic contains a model that is incrementally updated to have quick decision making instead of reinventing the when again each time a decision needs to be made. This algorithm is also tested in the Amazon AWS cloud with the deployed web-based application. The results are observed with HSA and without HSA and found significant difference in the optimizations. The application with comprehensive functionality of the ESADSA proved to be more efficient in self-adaptation automatically and dynamically. The experimental results revealed the increased utility of the architecture proposed.
6.4 FUTURE ENHANCEMENTS

The research visions of thesis led to the subsequent directions for upcoming work. These directions help in bringing into the line future research work to have more productive investigations of “dynamic self-adaptive software architecture” in the light of emerging complex and highly distributed applications.

1. It is interesting to investigate on the “dynamic self-adaptive software architecture” that provides the needs of “Software Product Line” (SPL). With SPL the adaptation strategies differ as it reflects line of software products with core functionalities similar while variability is introduced with a new product in the product line.

2. With the emerging of “Internet of Things” (IoT) technology, there is another direction for future work. It is to investigate further with ESADSA to make real world domain-specific systems integrated with IoT self-adapt.

3. Yet another direction for future work is to enhance ESADSA for self-adaptive security-based systems that are part of National Cyber Safety and Security (NCSS).