Implementation of Continuous Adaptive Requirements Engineering (CARE) Framework

By

Faiza Gul
2011-NUST-MS-PhD IT-003

Advisor: Dr. Nauman Qureshi

A thesis submitted in partial fulfillment of the requirements for the degree

of Masters of Science in Information Technology (MS IT)

In

Department of Computing,
School of Electrical Engineering and Computer Sciences (NUST),
Islamabad, Pakistan
(July 2014)

Co-Advisors: Dr. Hamid Mukhtar, Dr. Awais Shibli, Dr. Kashif Sharif

Approval

It is certified that the contents and form of the thesis entitled “Implementation of Continuous Adaptive Requirements Engineering (CARE) Framework” submitted by Faiza Gul have been found satisfactory for the requirement of the degree.

Advisor: __Dr. Nauman Ahmed Qureshi__

Signature:_____

Date: _____

Committee Member1: _Dr. Hamid Mukhtar_

Signature_____

Date:_____

Committee Member2: _Dr. Awais Shibli____

Signature _____

Date: _____

Committee Member3: __Dr. Kashif Sharif__

Signature _____

Date: _____

Abstract

The importance of self-adaptability has emerged as an exciting research area among the software engineering community in recent years. The future software systems are envisioned as self-adaptive. They will respond in a satisfactory manner to the changes in the system's requirements or context.

Service-Based Applications (SBA) are self-adaptive applications which have to respond in a satisfactory manner at run time whenever there is a change in user preferences, needs or context by selecting suitable third party web services available on internet. Requirements Engineering (RE) for such applications is different from conventional RE for non-adaptive systems. Design time requirements soon get out-of-date and continuous addition of new requirements or modification of existing requirements is needed which a real challenge for requirements engineering. There is a dire need to introduce new architectures and frameworks in this area.

In [12] and [13], a novel conceptual architecture for SBA is proposed under the title of 'Continuous Adaptive Requirements Engineering (CARE) framework' for capturing user requirements and system adaptation at runtime by incorporating these new user requirements. However, implementation of this framework is yet to be done. This research thesis is about the implementation of CARE framework by figuring out the efficient methods and techniques which will eventually lead to a seamless adaptation of user and system requirements at runtime.

My research has two main objectives:

-First, to implement 'Continuous Adaptive Requirements Engineering (CARE)' framework by using efficient methods and techniques so that a general purpose middleware is developed that supports runtime adaptation for any self-adaptive service based application in a user-oriented and goal-driven manner.

- Second, to provide better assessment of CARE framework by instantiating it through prototype service based application

Keywords*-Service-based applications, framework, self-adaptive requirements, requirements engineering*

CERTIFICATE OF ORIGINALITY

I hereby declare that the research paper titled "***Implementation of Continuous Adaptive Requirements Engineering(CARE) Framework***" is my own work and to the best of my knowledge. It contains no materials previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any degree or diploma at SEECS or any other education institute, except where due acknowledgment, is made in the thesis. Any contribution made to the research by others, with whom I have worked at SEECS or elsewhere, is explicitly acknowledged in the thesis.

I also declare that the intellectual content of this thesis is the product of my own work, except to the extent that assistance from others in the project's design and conception or in style, presentation and linguistic is acknowledged. I also verified the originality of contents through plagiarism software.

Author Name: Faiza Gul

Signature: _____

Acknowledgements

All praises to Almighty Allah, the most beneficent, the most merciful Who gives me the power of thinking. His blessings and mercy are the real source of all human accomplishments.

I would like to express my gratitude to Dr. Nauman Qureshi (my advisor) and Dr. Hamid Mukhtar, Dr. Awais Shibli, Dr. Kashif Sharif(my co-advisors) for their support, guidance and constant supervision in completing the thesis.

I would also like to express my gratitude towards my family for their kind co-operation, support and encouragement which helped me in completion of this work.

My thanks and appreciations also go to my colleagues who have eagerly supported and helped me in weak moments.

[To my Parents]

TABLE OF CONTENTS

List of Tables

List of Figures

-

1. INTRODUCTION

1.1 Problem Definition

The importance of self-adaptability has emerged as an exciting research area among the software engineering community in recent years. The future software systems are envisioned as self-adaptive which will respond in a satisfactory manner to the changes in the system's requirements or context. The main challenge is that how we can achieve it with least human intervention.

Self- adaptability is not a new concept. In real life, change is inevitable. Humans and other living species adapt themselves to the changes in environment. In robotics, sense, plan and act architecture model based on control loops is used for self-adaptability of these autonomous systems. A robot senses any changes in its environment, and then it creates a plan and acts on the plan to adapt itself to the changes occurred in its environment.

Service-Based Applications **(**SBA) are self-adaptive applications which are supposed to respond in a satisfactory manner at run time whenever there is a change in user preferences, needs or context by selecting suitable third party web services available on internet. For such applications, end-user requirements may get changed with respect to the dynamic environment in which these applications run and executed.

Requirement engineering activities' aim is to identify the purpose of the system and to specify system's properties and behaviors in order to satisfy the purpose in a precise and complete way. For traditional software systems, requirements are characterized as functional and non-functional requirements. Through interviews or domain documents using natural language, the analyst first tries to classify the stakeholders' needs into the above mentioned categories. Goal-oriented requirements engineering approach enables an analyst in providing a rationale behind functional requirements. Non-functional requirements are further categorized as soft goals (without a precise criteria for their satisfaction such as user friendliness) and Quality constraints (with a precise criteria e.g. less than 5 minutes). Preferences, another concept, are used to express stakeholder's attitude towards a particular requirement.

Requirements engineering for Self adaptive and self-configured software systems are very different from the traditional software systems. In traditional software systems, requirements are finalized mostly at design time. For SBA, it is required to be performed only at design-time (where more explicit constructs are needed in order to specify requirements) but also required at run-time.

A self-adaptive system's properties cannot be categorized just as functional or non-functional requirements but they also includes requirements for monitoring the variability in the operational context, evaluation criteria for checking where the system is not giving the expected values and behaviors to be adopted at run-time(alternative solutions) for achievement of end-users' goals. Such requirements are called adaptive requirements. For this purpose, in SBA, a feedback control loop is there which enables them to monitor any changes in the environment, then evaluate them and finally adapt its behavior accordingly.

For self-adaptive systems, we can categorize requirements according to the phase in software life cycle where they are specified. First, requirements can be specified at design time. These requirements describe the functional and non-functional requirements which are known at design time. They also include those techniques which are required for run time addition or modification of requirements. Requirements can also be specified at run time. The need for runtime requirements addition is triggered by the two reasons. It is either by change in user's needs and preferences or by the change in user's context and environment. In fact, for SBAs, design time requirements soon get out of date and continuous addition of new requirements or modification of existing requirements is needed which a real challenge for requirements engineering. Developers try to select Web services closest to design time requirements at design time from the particular application domain. At run time, as the user's preference and context changes, so a need arises to search for new web services.

1.2 Current Challenges for Self-Adaptive Systems

- Evolution of the requirements at run-time
- Quick responsiveness for adaptation
- Autonomic service discovery, invocation, composition and monitoring
- Incorporating several QoS parameters
- New requirements engineering language to cope with the uncertainty in requirements

1.3 Objective of the research

Future software systems are envisioned as self-adaptive. Frameworks are required to achieve self-adaptability at runtime. The main objective of this research is to implement a framework(CARE) that can support self-adaptability at runtime for service based applications by using efficient methods and techniques so that a general purpose middleware is developed that supports runtime adaptation for any self-adaptive service based application in a user-oriented and goal-driven manner. The main functionality which will be provided by this framework includes:

1. Capturing change in user preferences and context at runtime
2. Finding suitable web services accordingly
3. Selecting the best candidate web service
4. Updating the requirements

Also, the framework will capture user requirements at runtime by a wizard like easy to use interface through which end users can easily specify their needs by following simple and easy steps. And then convert these needs into a machine understandable format for further processing. In short, the objective of this research is:

- To Implement CARE framework that can support self-adaptability at runtime for service based applications

- instantiating CARE framework through prototype service based application to provide better assessment

1.4 Problem Statement

The problem statement of this research is*: 'Implementation of Continuous Adaptive Requirements Engineering (CARE) framework that provides runtime adaptability by responding in a satisfactory way to the changes in end-user preferences and context.'*

1.5 Reason/Justification for the Selection of the Topic

For self-adaptive service based applications, change is inevitable. Capturing and satisfying user requirements at run time have appeared a new challenge in the field of Requirements Engineering. Not much research is done in this direction. Requirements engineering for self-adaptive applications is still at its early stages. Novel methods and techniques are required in order to support run-time adaptation. The existing approaches do not support the accommodation of new or changed requirements. A novel conceptual architecture for SBA is proposed under the title of 'Continuous Adaptive Requirements Engineering (CARE) framework' for capturing user requirements/preferences and system adaptation at runtime but still there are many challenges from implementation point of view which needs to be addressed.

CARE architecture is finalized however implementation is yet to be done. CARE architecture consists of multiple agents. Implementing a middleware that will help mobile users in requirements specification for support of constant requirements reappraisal at runtime by selecting suitable web services with least human intervention is a challenge from research point of view.

The advantage of this research will be the implementation of a middleware which will support:

- Service based applications to achieve self-adaptability at runtime.
- Automatic monitoring of changes in context or user's goals at run time and will have the capability to find the web services which best satisfy user's preferences and goals.
- At runtime, there will be no need of an analyst; rather the users and the system will be responsible for reappraisal of requirements.
- An API of common functions will be developed for service based applications so that there will be no need to re-create them for each new service based application.

1.6 Areas of Applications

Following are the areas where CARE can find its applications:

- CARE framework can be used as a step towards the development of future software systems to realize the vision of Mark Weiser about ubiquitous computing where computing devices will penetrate in our everyday lives to the extent that we will hardly notice their presence. Self-adaptability based on context information is one of the key concepts to realize ubiquitous or pervasive computing.

- The CARE framework can be used as a middleware for Self-Adaptive Software systems (SAS) and service based applications (SBA) which requires adaptation at run-time as a result of change occurrence in user's preferences and operating contexts with minimal involvement of system administrators.

- Artificial Intelligence (AI) is another area, where CARE frame work can find its application.

1.7 Structure

This thesis is organized in following chapters:

Chapter 2 presents literature survey of state of the art approaches for development of self-adaptive systems.

Chapter 3 discusses the architecture of CARE framework. It also presents scenarios and sequence diagrams along with implementation details of CARE framework.

Chapter 4 discusses the goal model and implementation of an adaptive mobile application called iAssistant that instantiates CARE framework.

Chapter 5 illustrates the evaluation of CARE framework.

Chapter 6 presents concluding remarks and identifies some future research directions.

2. LITERATURE REVIEW

2.1 Self-adaptive Systems

Self-adaptive systems need to show self-optimizing, self-protecting and self-healing capabilities and adapt themselves in automated manner. They adjust their behavior in response to their perception of the environment. System's high level goals and their priorities may change at run-time. The term autonomic computing was first coined by IBM in 2001[3] for self-adaptive systems which exhibit self-management, self-configuration, self- optimization, self-protection and self-healing capabilities. Given the high level administrator's goals and objectives, these systems will be able to manage themselves and their operations in a seamless manner in the face of external conditions and demands.

2.2 Current Challenges

The journey towards a fully adaptive system will take many years as there are several valuable and important milestones along the path. Innovative ways need to be investigated to develop such systems. Only recently the first attempts have been made in developing self-adaptive systems within specific application domains. Proper realization of self-adaptation is still a significant intellectual challenge as discussed in [1] and [2] which provide a research road map to identify key technical and scientific challenges for self-adaptive systems to realize true implementation of self-adaptability in future. In [1], the following areas are identified where research is required to realize true implementation of self-adaptability in future:

1. A self-adaptive system has to perform a trade-off analysis between several conflicting goals. Practical techniques are needed to specify and generate utility functions. For example use of preferences to compare the situations under Pareto optimal conditions.

2. More research is required on lightweight monitoring techniques when there are several QoS parameters need to be considered.

3. Centralized control loop pattern used for engineering self-adaptive systems has scalability problem for a large-scale software system. Research is needed for a decentralized approach for constructing self-adaptive systems.

4. Quick responsiveness is a current challenge for adaptation.

5. More advance models for predictability of exact behavior of the system in response to a change is required.

6. The requirements specification of an adaptive system is not complete. Research is needed in the domain of the evolution of requirements at run-time.

7. New requirements engineering language is required for self-adaptive systems to cope with the uncertainty in their requirements.

8. New techniques are required to map the requirements into architecture. Right now component-based, product-line based and aspect-oriented approaches are used.

9. Managing uncertainty in requirements remains a key challenge and requires future work.

10. Research is required for refinement of goals at runtime.

11. Traceability from requirements to implementation and vice versa is a continuous challenge for self-adaptive systems.

12. Research is required in the area where there are several control loops and we need to make them explicit so that designer can reason about them. Also new reference architecture models to address issues related to them are required. Verification and validation techniques for the testing of control loop are required.

In [1], challenges are discussed for software engineering of self-adaptive systems for the following four views of self-adaptation:

- ✓ Modeling dimensions
- ✓ Requirements
- ✓ Engineering
- ✓ Assurances

The detail review against each view is given in the following:

- ✓ **Modeling Dimension**

  The first view 'modeling dimensions' is used to obtain precise models which will support run-time decision making and reasoning for attaining self-adaptability. The conception of self-adaptation depends on various aspects such as user needs, environment characteristics, and other system properties. Each modeling dimension for self-adaptive systems specifies a particular aspect of self-adaptive systems. Modeling dimensions can be classified in four groups w.r.t:

  1. System goals
  2. Change (Causes of the self-adaptation)
  3. Mechanisms to achieve self-adaptation
  4. Effects of self-adaptability on a system

- ✓ **Requirements**

  A self-adaptive system needs to monitor the changes in its environment and react accordingly. But the problem is that the system cannot monitor all aspects of the environment. Moreover, it is possible that the system detects less than optimal conditions regarding an aspect of the environment. In any case, it is required that high-level goals must be met. Non-critical goals can be relaxed allowing some degree of flexibility. Requirements engineering needs to address what are the possible adaptations and what are constrains in order to realize these adaptations. For that matter, aspects of the environment needs to be identified which are

relevant to adaptation. Also the requirements must be categorized whether they must always be maintained or allowed to be changed at run-time.

- ✓ **Engineering**

  Engineering of the self-adaptive systems remain a major challenge. They are required to focus on feedback and control loop principles. In self-adaptive systems design decisions are made partially at run-time. The system reasons about the state and environment. It involves feedback process with four key activities: collect, analyze, decide, and act. This feedback behavior exhibited by a self-adaptive system is realized with the help of control loops and it is an important feature. It should be considered the most important entity in modeling, designing and implementation of the system.

- ✓ **Assurances**

  The main goal of system assurance is to provide evidence whether a system satisfies functional and non-functional properties. It is quite difficult to achieve this goal and novel verification and validation methods are required. The challenge is to provide runtime assurances. For self-adaptive systems, the continuous changes occur as the context changes hence a continuous need for runtime verification and validation is required. Also a framework for assurances is proposed in the paper.

In [2], overview, future directions and associated challenges for self-adaptation is given for the following four areas:

- Design space decisions about the adaptive systems which need to be taken by the system developer by observing the system and its environment. On the basis of such decisions, proper adaptation mechanisms are selected.

- Defining and investigating new innovative process to cover complete life cycle of an adaptive system as the evolution activities of such systems are shifted to runtime.

- The four activities of adaptation (monitor, analyze, plan, and execute) forms a feedback control system. Systematic engineering approaches need to be defined in order to design centralized and decentralized control schemes for system adaptation.

- Innovative runtime efficient methods and techniques needs to be investigated for verification and validation at runtime of adaptive systems which is an urgent necessity to establish a trust in adaptation mechanism.

2.3 State-of-the-art approaches

Any adaptation mechanism can be decomposed into four major processes: monitoring, analyzing, planning and executing (MAPE) as discussed in [4]. Several approaches are used to achieve self- adaptivity in software systems. These state-of-the-art approaches are summarized in [5]. Main approaches identified are:

- External control mechanisms

- Model-driven approaches
- Component based software engineering
- Feedback systems
- Multiagent systems
- Nature-inspired engineering

The detail of these approaches is given below:

- **External control mechanisms** utilize separate modules to detect and resolve system adaptivity problems. These modules can be modified and reused across different systems. These solutions realize the adaptation processes in a separate adaptation manager which is external to the application logic. An adaptation manager is used to perform the four processes of adaptation (monitoring, analyzing, planning and executing) in order to control the adaptable software behavior. The application logic which is separate from adaptation manager is part of the adaptable software. It has the proper sensors and effectors required for adaptation.

  In [6], a framework is presented which uses architecture based self-adaptivity approach by using external control mechanisms. The framework utilizes the architectural model of the system as runtime artifact which is used to monitor the system to achieve self adaptivity at runtime.

- **Model-driven** approaches usually use a middleware that selects one of the abstract models and then generates corresponding model-to-code transformations. In [7], an external application independent middleware based adaptation approach (MADAM) has been adopted. An architecture centric approach has been used in order to realize the middleware. This middleware performs three main functions related to adaptation:

  - **Context management**: middleware monitors the context of the system, detects any change in the context and then reasons about these changes.

  - **Adaptation management:** middleware makes a decision about the application of proper adaptation strategy

  - **Configuration management:** Reconfiguration of the application

  At design time, the application developer specifies adaptation capabilities and code for the runtime environment by using tools and modeling language of MADAM framework. He needs to develop architecture models of application which will be used at runtime by the adaptation middleware. These models are converted into java code by using transformation tools. The java code is accessed by the middleware at runtime for adaptation.

- **Component based approaches**[8],[9],[10],[11] tries to get the best candidate component from a repository in case a component fails at runtime to achieve self-adaptivity. In [8], a self-organizable framework is presented that manages and reorganizes modular components of a system in a changing environment at runtime while the system is executing. In [10], a self-managed component based architecture is presented where at runtime different components

configure their interaction in order to achieve the overall system goals. A three layers (Component Control, Change Management and Goal management) reference model is also presented for self-management.

- **Control engineering approaches** promise integration of self-adaptive capabilities in the software systems. They introduce feedback control loops to achieve system adaptivity in dynamic and unpredictable environments in an effective and timely way. Control theory identifies control loops as crucial elements in order to incorporate adaptation in the software systems and considers self-adaptive software as a closed-loop system because it constantly monitors its environment, context and itself and then adapts properly. Two approaches can be used; either a single centralized control loop may realize the adaptation in a software system or several control loops may be utilized to realize the adaptation in composite software systems in decentralized manner.

  A detailed analysis of existing solutions in this area is discussed in [14]. In [12] and [13], an architecture of 'Continuous Adaptive Requirements Engineering (CARE)' framework is proposed that is based on feedback control loops and performs continuous requirements engineering (RE) at run-time for SBA by involving the end-user in a goal driven manner.

- **Multi-Agent** systems (MAS) [15],[16],[17] comprise of agents which are independent entities. Agents can autonomously perform their tasks and work together to achieve system's overall goal. The Multi-Agent System paradigm can be considered as an alternative approach in order to design adaptive, intelligent and robust systems. They realize adaption by using decentralized control through distributed and autonomous agents and introduce robustness, flexibility and modularity in the systems.

  In [16], a multi-agent fuzzy-neural system is used in a wafer fabrication factory to increase the overall performance of scheduling jobs. Every agent in the system creates and changes its own scheduling algorithm in order to adapt to its environment. This approach eventually increases the scheduling jobs performance in terms of average cycle time and cycle time standard deviation.

- **Nature-inspired engineering** is inspired by biological systems. Research in this area focuses to achieve self adaptivity in software systems in same way as the natural and biological systems. Some efforts have been done in this direction [17], [18], [19]. The rationale behind this approach is that if we are able to understand how complex things are performed in a simple way in nature and biological systems, then we will be able to apply this knowledge in order to design powerful adaptive systems. In [17], some mechanisms from the nature have been studied in order to understand and solve complex engineering problems by using multi- agents.

In [20], a detailed discussion is given to show how the evolution in software methodologies and technologies eventually led to the development of self-adaptive service based applications. It also discusses how requirements evolution played an important role to the journey of the self-adaptive and highly dynamic applications. **Service-based applications** are self-adaptive in nature. Self-adaptivity is required as the applications must autonomously and automatically adapt to the failures of component services, changes in context and evolving requirements. The key factors that trigger the

need for adaptation for SBA are: new or emerging needs and preferences specified by the end user, change in contextual information of the system and/or the end user, relevant service availability or resource availability. SBAs make a promising solution in order to realize highly dynamic adaptable distributed systems. There is a constant need for RE activities at runtime which lead to a seamless evolution of service based applications and their specification.

The concept of requirements monitoring is first presented in [21] and [22]. To perform continuous requirements engineering (RE) at run-time by involving the end-user for service based applications, a framework (CARE) is presented in [12] and [13]. CARE is a framework that performs continuous requirements engineering (RE) at run-time by involving the end-user for service based applications. It is based on service-oriented architecture in which system itself supports run-time analysis and refinement of requirements. It is goal oriented and user oriented which aims to provide third party services available on the internet when the user goals change. It adapts itself in order to meet the changes in goals or user preferences in a satisfactory manner. For this purpose, CARE depends on two types of knowledge. First of all, it should know its own goals and requirements and second it must know the requirements of the environment in which it operates. Whenever a change occurs in the environment or the user preference changes then it selects another service that best suits the changed environment and user preference. Continuous reappraisal of requirements (addition of new requirements or modification of existing ones) is required to handle the openness and dynamicity of the environment. CARE helps to achieve continuous reappraisal of requirements. Further detail about CARE framework is discussed in Section 3.

In [29], a Modeling language called Adaptive Requirement Modeling Language (ARML) is used to systematically model adaptive requirements. It enables the analyst during early RE to formulate and analyze requirements problem before going for detailed specification. Concept of context and resource is part of ARML. Similarly relegation and influence relations are also included. Event-Condition-Action (ECA) pattern was introduced to operationalize adaptive requirements and to derive monitoring specification, evaluation criteria and action specification.

Research is done in the areas to explore new user friendly ways to capture user requirements at runtime by involving end users directly. In [23], end users have been involved in requirements elicitation process. In this paper, a prototype of a mobile requirements elicitation tool called iRequire is given which can be used by the end users to blog their needs. The end users can easily specify their needs. IRequire also supports context sensing technologies like GPS to get up-to date information about users context in which these needs arises. Other mobile technologies like Camera and audio can be used to capture a picture of the environment or record the requirements in audio form. This tool has a wizard like easy to use interface which guides the end users. A user can document his needs by following four simple and easy steps: First step is to take a picture of the environment through mobile camera which helps to determine the context in which a user need arises. Second step is to document the needs either through text-based requirements descriptions or through audio recording. The third step is to document the rationale to clarify that why a need is important to the end user. Again, it can describe either through text-based requirements descriptions or through audio recording. The last step is the display of summary on documented needs for final confirmation from the user. After this step, the data is stored in the database along with current context and time stamp.

This data can be used later on for further analysis so that these needs can be transformed into more structured requirements. IRequire was built by using .NET Compact Framework 3.5, a subset of the .NET Framework which has libraries to support mobile application development like to access mobile Camera and GPS receiver etc. Microsoft's Visual Studio 2008 IDE and the Windows Mobile 6.5 Toolkit were used for development. SQL Compact Edition Database was used to store users' needs on mobile device. SQL Server Compact 3.5 was used which is a database engine used for building a database that can run on Windows Mobile smartphone. The database of iRequire has 5 tables: Requirements, User, Picture, Audio and Log.

In [24], a version of iRequire tool 'bada iRequire' is given which is a mobile RE tool developed with Samsung for bada platform. It removes some of the limitations of iRequire prototype of [23]. With this tool, Samsung will be able to provide new applications and features according to the user needs which will eventually help in the development of an end-user driven platform.

In [25], an idea of a tool based solution is proposed to combine CARE framework with iRequire so that end user are directly involved in eliciting requirements at runtime for service based applications.

2.4 Development Approaches

For the development and implementation of software systems with adaptation abilities, existing frameworks and middleware used for the development of distributed systems played a vital role as discussed in [5]. They are given in the following:

- **Java 2 Enterprise Edition (J2EE)**

  Java 2 Enterprise Edition (J2EE) is extensively used among research community for the development of adaptive software despite the fact its API provides only partial support for adaptation. The selection of a Java-based execution environment is usually used to achieve high degree of cross-platform portability.

- **.NET**
  Another approach which is widely used for development of adaptive software is .NET by Microsoft.

- **Open Services Gateway initiative (OSGi) framework**

  OSGi (Open Service Gateway Initiative) is a Java based service framework. It is used for modular software programs development and deployment. It helps developers to define Modular components called bundles. These bundles can be managed remotely. For example, they can be installed, updated, uninstalled, started or stopped and how they will interact. To facilitate adaptation for service-oriented architecture (SOA), it helps in specifying a service registry which can be used by the bundles for service publishing, discovery and binding.

- **Adaptive Communication Environment (ACE)**
  The Adaptive Communication Environment (ACE) is an object-oriented (OO) framework which is open-source and freely available. It provides a set of reusable patterns and

framework components to perform communication tasks on different OS platforms. These tasks include event handling inter-process communication, synchronization, concurrent execution, shared memory management and event demultiplexing etc. It simplifies the system automation as it helps the developers to link services dynamically at runtime into applications and then execute them in different threads or processes.

- **JADE (Java Agent DEvelopment Framework)**
  It's the java based solution that is widely used to develop adaptive Multi-Agent system. Using JADE, a developer can easily specify goals and corresponding plans required to achieve these goals.

3. CARE FRAMEWORK

SBAs are distributed applications as they have to depend on third party web services. They have to adapt themselves quickly to cope with dynamism because all the time, new web services appear and existing ones disappear. User goals and preferences for SBA may change at runtime. Similarly, the context conditions are also changed so there is a constant need for RE activities at runtime. This leads to a seamless evolution of service based applications and their specification.

The key factors that trigger the need for adaptation for SBA are: new or emerging needs and preferences specified by the end user, change in contextual information of the system and/or the end user, relevant service availability and resource availability.

CARE is a framework that performs continuous requirements engineering (RE) at run-time for SBA by involving the end-user. It is based on service-oriented architecture in which system itself supports run-time analysis and refinement of requirements. It is goal oriented and user oriented which aims to provide third party services available on the internet when the user goal changes. It adapts itself in order to meet the changes in goals or user preferences in a satisfactory manner. For this purpose, CARE depends on two types of knowledge. First of all, it should know its own goals and requirements and second it must know the requirements of the environment in which it operates. Whenever a change occurs in the environment or the user preferences changes then it selects another service that best suits the changed environment and user preferences. Continuous reappraisal of requirements(addition of new requirements or modification of existing ones) are required to handle the openness and dynamicity of the environment in which SBA operates and CARE helps to achieve continuous reappraisal of requirements.

In [13], following two types of RE activities for SBA are identified:

- **RE activities at design-time**

  Design time RE involves activities which are performed by a human analyst by consulting system stakeholders to find goals and domain knowledge. It includes Goal elicitation, analysis, and elaboration to generate goal oriented requirements specifications. For that purpose, goal oriented methodologies are adopted. These activities also involve determining the monitoring requirements and adaptation actions which can be used by SBA at runtime for achieving self-adaptability. Therefore the requirements resulted from these activities are called adaptive requirements. These design time adaptive requirements are described in terms of different alternatives. A modeling language, called Adaptive Requirements Modeling Language (ARML) based on Techne modeling language is used for representation and specification requirements at this stage.

- **RE activities at runtime**

  Run-time RE activities are performed by system itself. The adaptive requirements specification acts as an input to CARE framework which performs requirements engineering at run-time. At runtime the system itself acts as an analyst. It performs requirements engineering activities by using information obtained through monitoring or specified by the

end user. It then decides on better alternative solutions in order to best satisfy end user goals. End user is actively involved in RE process at runtime. The new requirements specified by them are called service requests. Runtime RE activities help in refining requirements artifact.

Following four types of adaptations are supported by CARE framework as discussed in [12] and [13]:

- **Type 1:** anticipated changes for which alternatives are given in the specification at design time.
- **Type 2:** changes in the context or preferences of the user at runtime which are monitored by system and then the system adapts to the most suitable alternative solution using different services after evaluating them. The alternative solution may not exist in the design time specification.
- **Type 3:** unanticipated changes where end user is prompted to provide input or possible solution
- **Type 4:** unanticipated changes where there is no possible solution or refinements exists

3.1 Architecture of CARE Framework

The simplified architecture of CARE framework is given in the following figure:

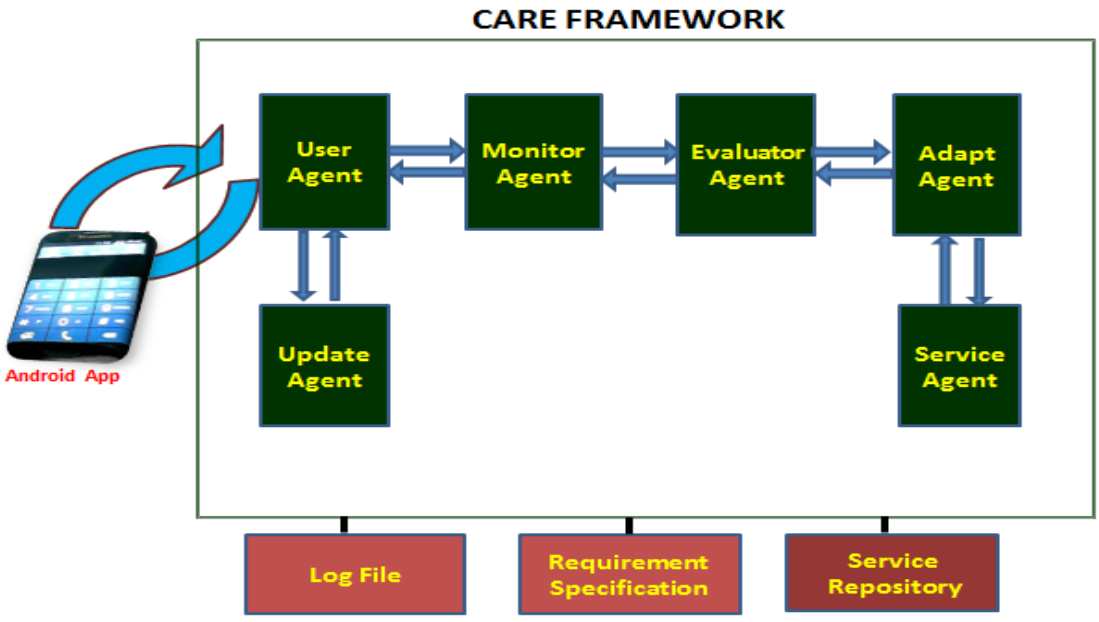


Figure 3.1: Architecture of CARE framework

The architecture of CARE is self-adaptable where different components automatically configure their interaction at runtime in such a way that it achieves the overall system goal. It consists of a following set of agents:

- User Agent
- Monitor Agent
- Evaluator agent
- Adaptation Agent
- Service Agent
- Update Agent

.

It performs following four RE activities at runtime:

1) Request Acquisition
2) Evaluation, Planning and Adaptation
3) Service Lookup and Selection
4) Update Specification

To perform the above activities, CARE uses a set of available services from service repository. Service can be added, dropped or replaced with new ones as required by the system at runtime. Detail of these activities is given below:

1) Service Request acquisition

During this activity, new requests are captured in the form of Runtime Requirements Artifact (RRA). Requests are generated either by the user or by the system itself which triggers the need for adaptation. These two types of RRA are given as:

- *User generated RRA*

Using a user friendly interface of the application running on top of CARE framework, a user can specify her new request. User preferences are handed to user Agent which generates RRA. The user generated RRA has the following format:

```
<?xml version="1.0" encoding="UTF-8" standalone="true"?>
- <RRA>
    <Source>User</Source>
    <FuncGoal>Find Plane Options</FuncGoal>
    <RRAStatus>NEW</RRAStatus>
    <Time>00:10:03</Time>
    <Date>2014-05-16</Date>
    <Context Longitude="400" Latitude="200"/>
    <Service endpoint="GetWebServiceStatus"
        uri="http://10.0.2.2:81/AirTravelAgent/Service1.asmx"
        namespace="http://tempuri.org/" type="soap"
        category="Flights" name="Air Travel Agent"/>
    <Parameter Source="Islamabad"/>
    <Parameter DepartureDate="12/06/2014"/>
    <Parameter DepartureTime="12:30pm"/>
    <Parameter Destination="Karachi"/>
    <Preference TicketCost="1000"/>
</RRA>
```

- *System generated RRA*

The system constantly monitors the contextual changes (i.e. location, resources, services etc.). Any violation of a goal, change in user preferences or some unexpected behavior of a service triggers a need for adaptation. For example, once the web service is selected and SBA is bound to it then constant monitoring of web service is required whether it is delivering the required functionality over a given period of time or not. Otherwise a need to discover a new web service is generated and the whole process for web service discovery is repeated.

Monitor Agent of the CARE framework is mainly responsible for this activity. It monitors the changes in

- User context (latitude, longitude etc.)
- Available resources (internet etc.)
- Web service failure
- Status changes (Booking cancelled, meeting delayed etc.)

Any change triggers a need for adaptation.

2) Evaluation, Planning and Adaptation

User Agent hands over the generated RRA to the evaluator agent which in turn gives it to adaptation agent for suitable operationalization. Adapt Agent takes RRA and Requirement specification as input and then applies one of the four adaptation strategies. It calls the service agent when required. The format for Requirement Specification is given as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
- <Requirements>
    <FuncGoal parentFuncGoal="" name="Manage Travel" id="0"/>
    <FuncGoal parentFuncGoal="0" name="Find Means of Transportation" id="1"/>
    <FuncGoal parentFuncGoal="1" name="Find Plane Options" id="2"/>
    <Task parentFuncGoal="2" name="Get Plane Itenary Detail" id="0" preference="1"/>
    <Service name="Air Travel Agent" parentTask="0" endpoint="GetWebServiceStatus"
        uri="http://10.0.2.2:81/AirTravelAgent/Service1.asmx" namespace="http://tempuri.org/"
        type="soap" category="Flights"/>
    <FuncGoal parentFuncGoal="1" name="Find Train Options" id="3"/>
    <Task parentFuncGoal="3" name="Get Train Itenary Detail" id="1" preference="1"/>
    <Service name="Train Service" parentTask="1" endpoint="TestService"
        uri="http://10.0.2.2:86/TrainService/Service1.asmx" namespace="http://tempuri.org/"
        type="soap" category="Trains"/>
    <FuncGoal parentFuncGoal="0" name="Display Options" id="5"/>
    <FuncGoal parentFuncGoal="0" name="Book" id="6"/>
    <FuncGoal parentFuncGoal="0" name="Send Confirmation" id="7"/>
    <Task parentFuncGoal="7" name="Send Sms" id="3" preference="1"/>
    <Task parentFuncGoal="7" name="Make Phone Call" id="4" preference="2"/>
    <Task parentFuncGoal="7" name="Send Email" id="5" preference="3"/>
</Requirements>
```

3) Service Look up

Service Agent is the main agent that performs Service Look up and selection activity. Adapt agent requests for a service lookup and selection process which contains all necessary information. By utilizing this information, Service Agent tries to check the services stored in Service repository by performing look up operation. Note that Web services from the particular domain are already stored in Service repository at design time. As a result, new services are selected in case of requirements addition or existing ones are replaced with related ones in case of requirements refinement. The end result of this activity is a list of services.

Based on different QoS parameters, ranking of web services is done. The highest rank service is selected and the application is bound to it. Currently, QoS parameters used for the selection in implementation of CARE are:

- Availability
- Minimum response time

4) Update Specification

This activity is performed by Update Agent. As a result of this operation, existing repositories (Requirements specification and service repository) are updated accordingly and a log is created of each change made.

3.2 Goal Modeling

A Modeling language called Adaptive Requirement Modeling Language (ARML) presented in [29] is used to systematically model adaptive requirements. It enables the analyst during early RE to formulate and analyze requirements problem before going for detailed specification. Concept of context and resource is part of ARML. Similarly relegation and influence relations are also included.

In the following, implementation of different agents is explained with the help of diagrams which are developed using ARML.

The user agent of CARE framework is responsible to perform activities related to communication with user. It gets functional goal and preferences for requests from users and generates new RRA or modifies existing one after reading requirements specification document and invokes monitor. It also gets the user feedback. . Detail of the responsibilities of this agent is explained using following diagram:

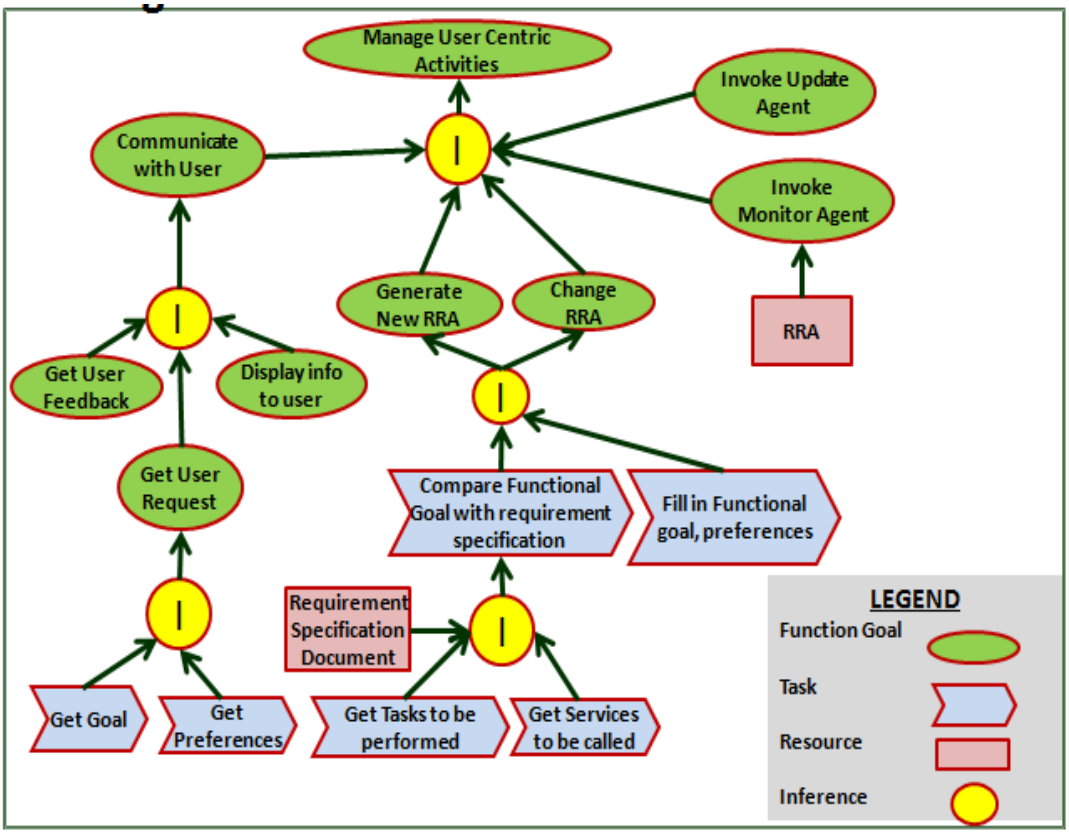


Figure 3.2: User Agent

The monitor agent is responsible to monitor the status of web services and booking status etc. and invokes evaluator agent accordingly. Detail of the responsibilities of this agent is explained using following diagram:

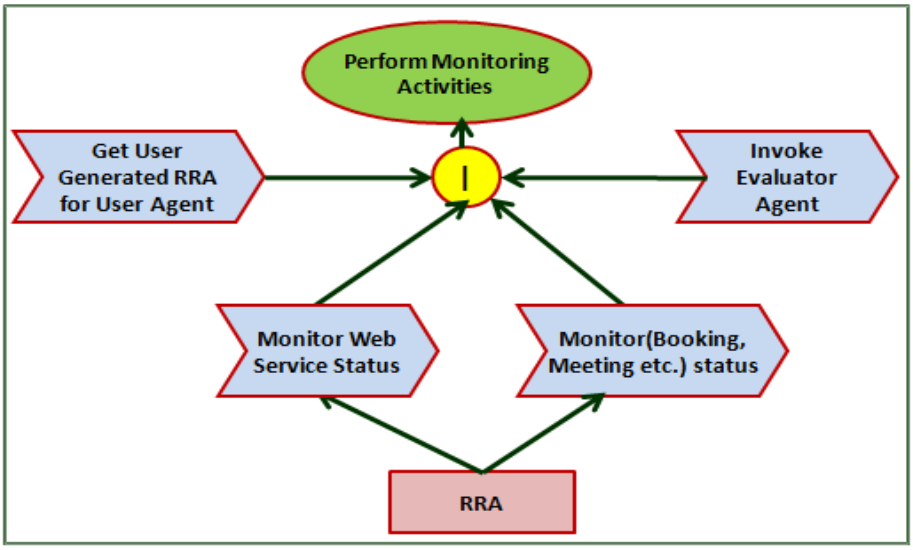


Figure 3.3: Monitor Agent

Evaluator agent evaluates the request coming from monitor agent and then calls adapt agent if required. Similarly, it also evaluates the results coming from adapt agent. Detail of the responsibilities of this agent is explained using following diagram:

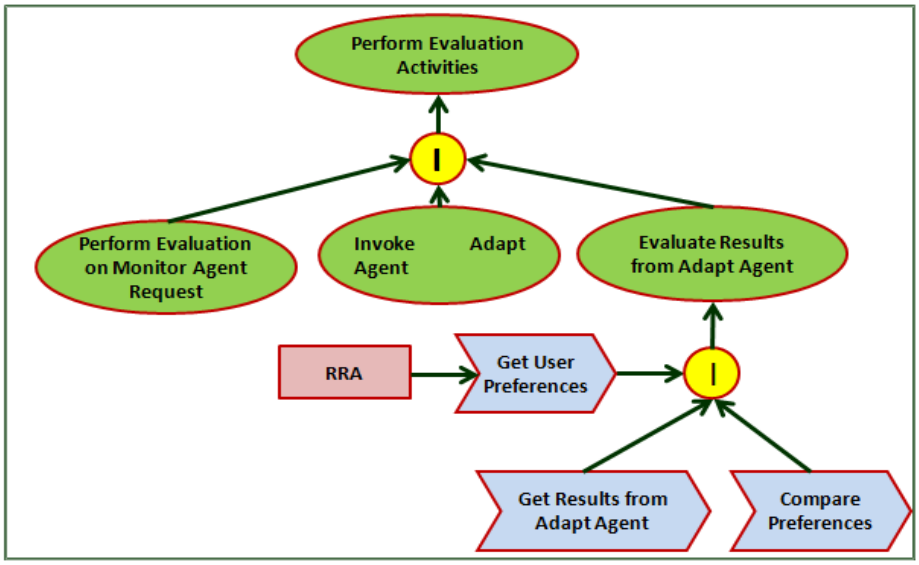


Figure 3.4: Evaluator Agent

Adapt agent is mainly responsible to perform one of the four adaptation strategies. Detail of the responsibilities of this agent is explained using following diagram:

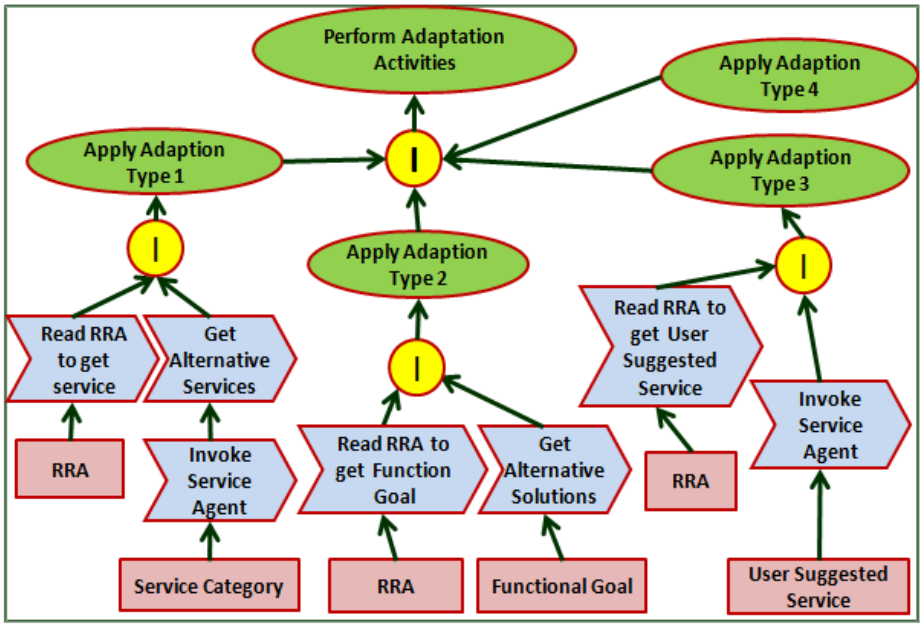


Figure 3.5: Adapt Agent

Service agent performs service lookup and invocation activities. It also searches repository to discover new services on request. Detail of the responsibilities of this agent is explained using following diagram:

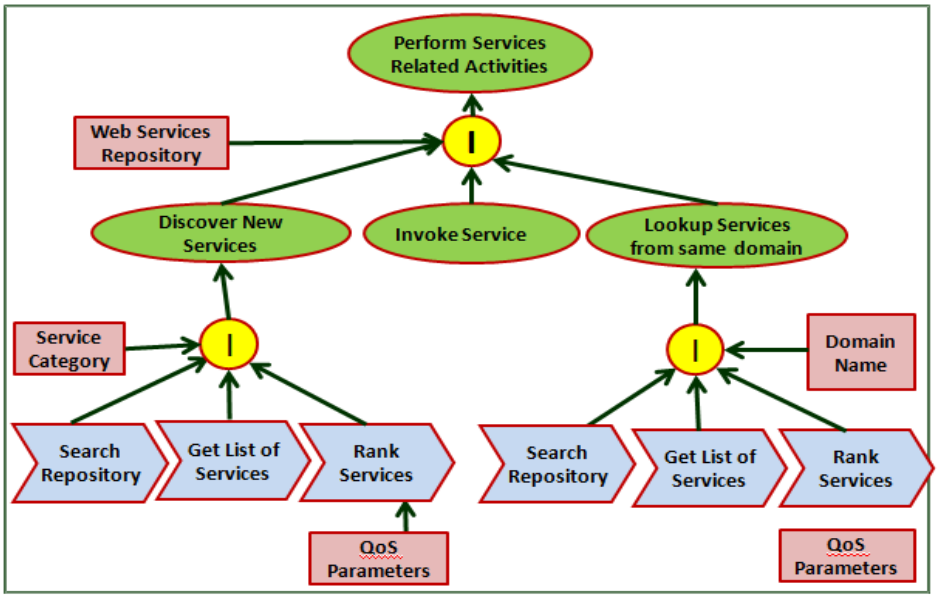


Figure 3.6: Service Agent

Update agent updates the requirement specification document in case of adaptation type 3. It also creates a log in case of adaptation type 4. Detail of the responsibilities of this agent is explained using following diagram:

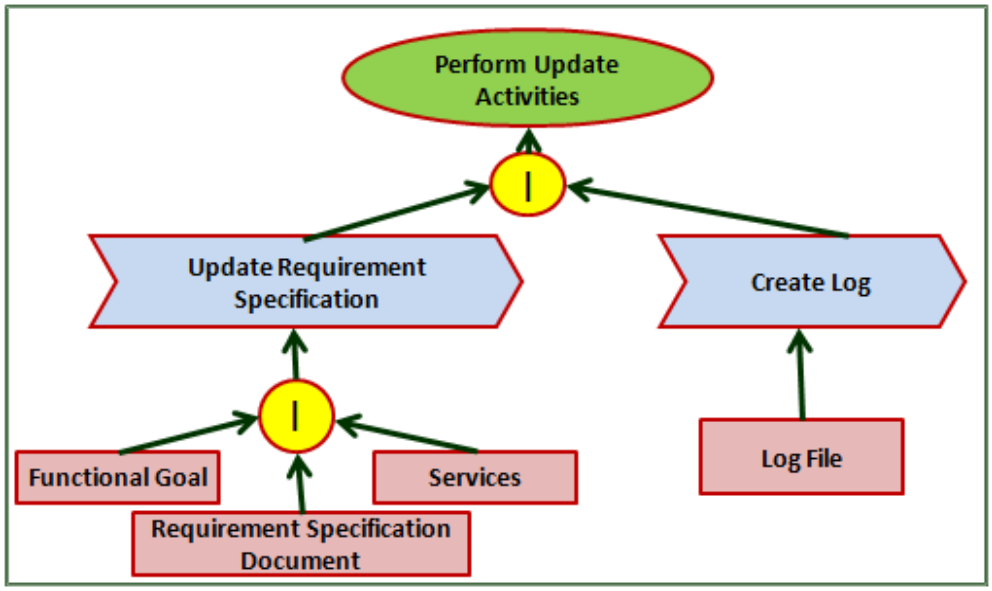


Figure 3.7: Update Agent

3.3 Scenarios and Sequence Diagram of CARE to support adaptation

As already discussed CARE framework supports four types of adaptations. In the following, scenarios are given along with sequence diagrams to explain how CARE framework achieves each one of these adaptation scenarios:

Type 1 Adaptation	Anticipated changes for which alternatives are given in the specification at design time.
Scenario	1. During monitoring, Monitor agent gets current status of the selected service in the RRA and informs evaluator agent. 2. Evaluator agent evaluates the information about the status of the web service. In case of service unavailability, it informs adapt agent for further action. 3. Adapt agent first apply adaptation type 1 and asks service Agent to get alternative services. 4. Service agent looks for suitable alternative services, rank them and then returns the candidate services to adapt agent.

	5. Adapt agent passes results to evaluator. 6. Evaluator agent returns results to monitor agent. 7. Through user interface agent, monitor agent informs the user about service unavailability and asks to select one of the available alternative services. 8. User selects one of the services. 9. User Interface agent informs User agent of CARE. 10. User Agent updates RRA with new service. 11. Monitor Agent again starts monitoring the web service status.

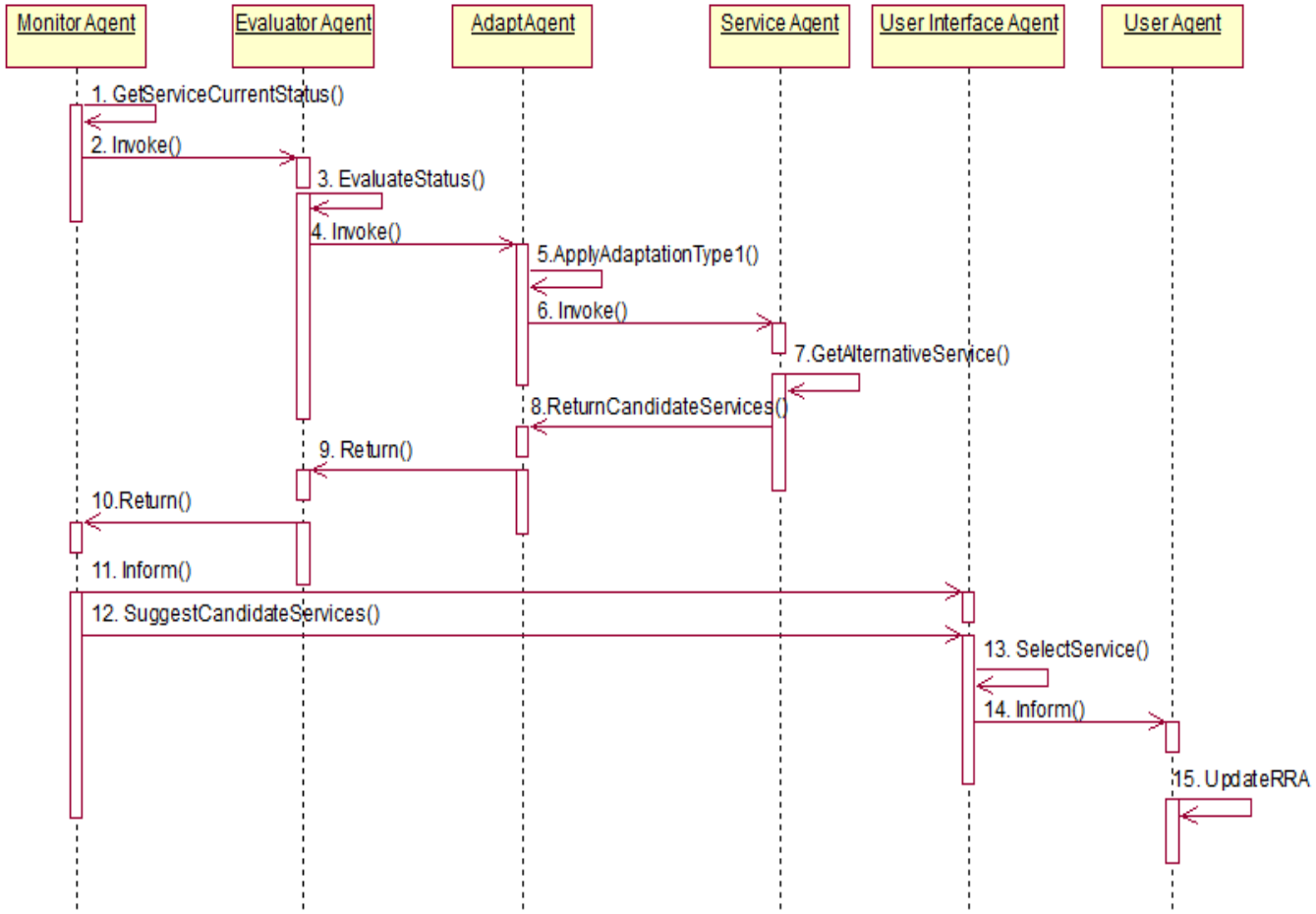


Figure 3.8: Sequence diagram for Type 1 Adaptation

Type 2 Adaptation	Changes in the context or preferences of the user at runtime which are monitored by system and then the system adapts to the most suitable alternative solution using different services after evaluating them. The alternative solution may not exist in the design time specification.
Scenario	1. During monitoring, Monitor agent gets current status of the selected service in the RRA and informs evaluator agent. 2. Evaluator agent evaluates the information about the status of the web service. In case of service unavailability, it informs adapt agent for further action. 3. Adapt agent first applies adaptation type 1 and asks service Agent to get alternative services. 4. Service agent informs adapt agent that no alternative service is available. 5. Adapt agent then applies adaptation type 2 and reads the requirement specification document to get alternative solutions against the functional goal. 6. Adapt agent passes results to evaluator. 7. Evaluator agent returns results to monitor agent. 8. Through user interface agent, monitor agent informs the user about service unavailability and asks to select one of the available alternative solutions. 9. User selects one of the alternative solutions. 10. User Interface agent informs User agent of CARE. 11. User Agent updates RRA and operationalize the alternative solution.

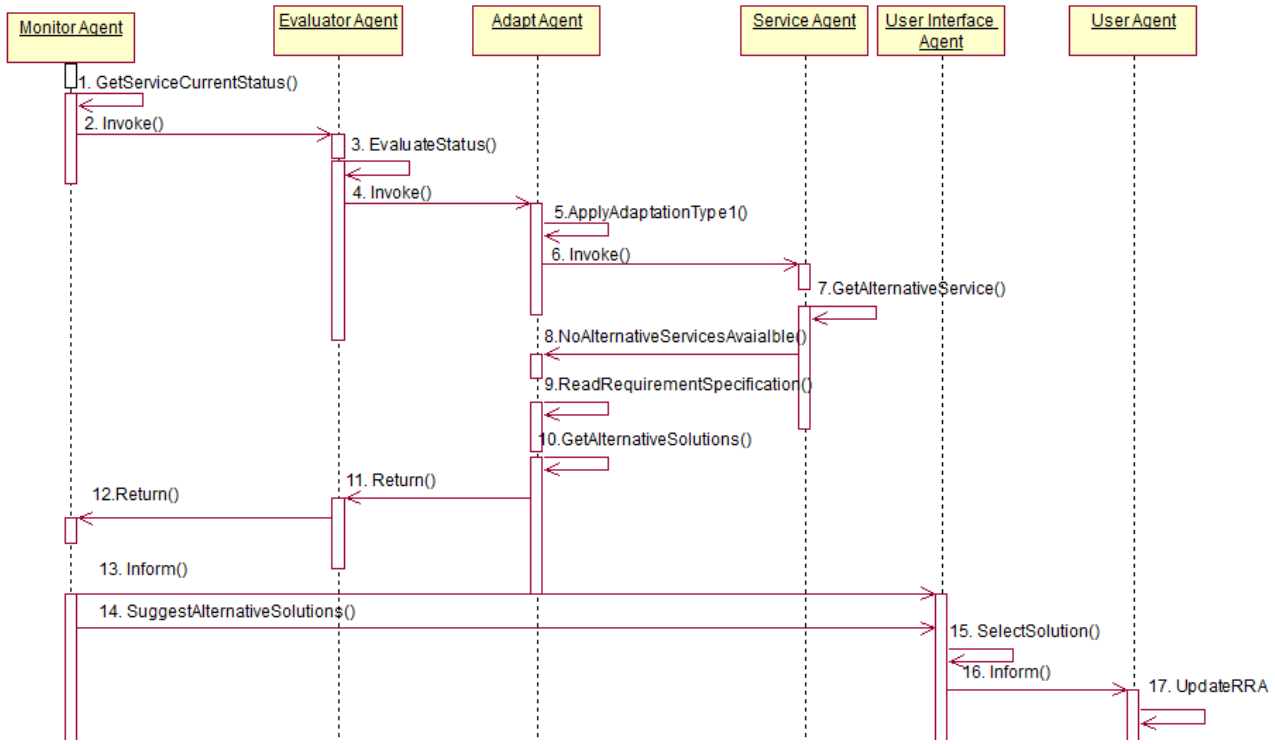


Figure 3.9: Sequence diagram for Type 2 Adaptation

Type 3 Adaptation	unanticipated changes- end user is prompted to provide input or possible solution
Scenario	1. During monitoring, Monitor agent gets current status of the selected service in the RRA and informs evaluator agent. 2. Evaluator agent evaluates the information about the status of the web service. In case of service unavailability, it informs adapt agent for further action. 3. Adapt agent first applies adaptation type 1 and asks service Agent to get alternative services. 4. Service agent informs adapt agent that no alternative service is available. 5. Adapt agent then applies adaptation type 2 and reads the requirement specification document to get alternative solutions against the functional goal. 6. Adapt agent passes results to evaluator.

	7. Evaluator agent returns results to monitor agent. 8. Through user interface agent, monitor agent informs the user about service unavailability and asks to select one of the available alternative solutions or suggest her own solution. 9. User opts to suggest a new solution with a new plan and a service to operationalize the suggested plan. 10. User Interface agent informs User agent of CARE. 11. User Agent updates RRA and informs monitor agent. 12. Monitor agent calls evaluator who calls adapt agent agent. 13. Adapt agent asks service Agent for service invocation. 14. Service agent checks about the service availability. In case service is available, it informs the adapt agent. 15. Adapt agent passes results to evaluator. 16. Evaluator agent returns results to monitor agent who informs the user agent. 17. User agent informs the user interface agent. It also calls the update agent. 18. Update agent updates the requirement specification document with new functional goal and the service that can be used to operationalize that goal.

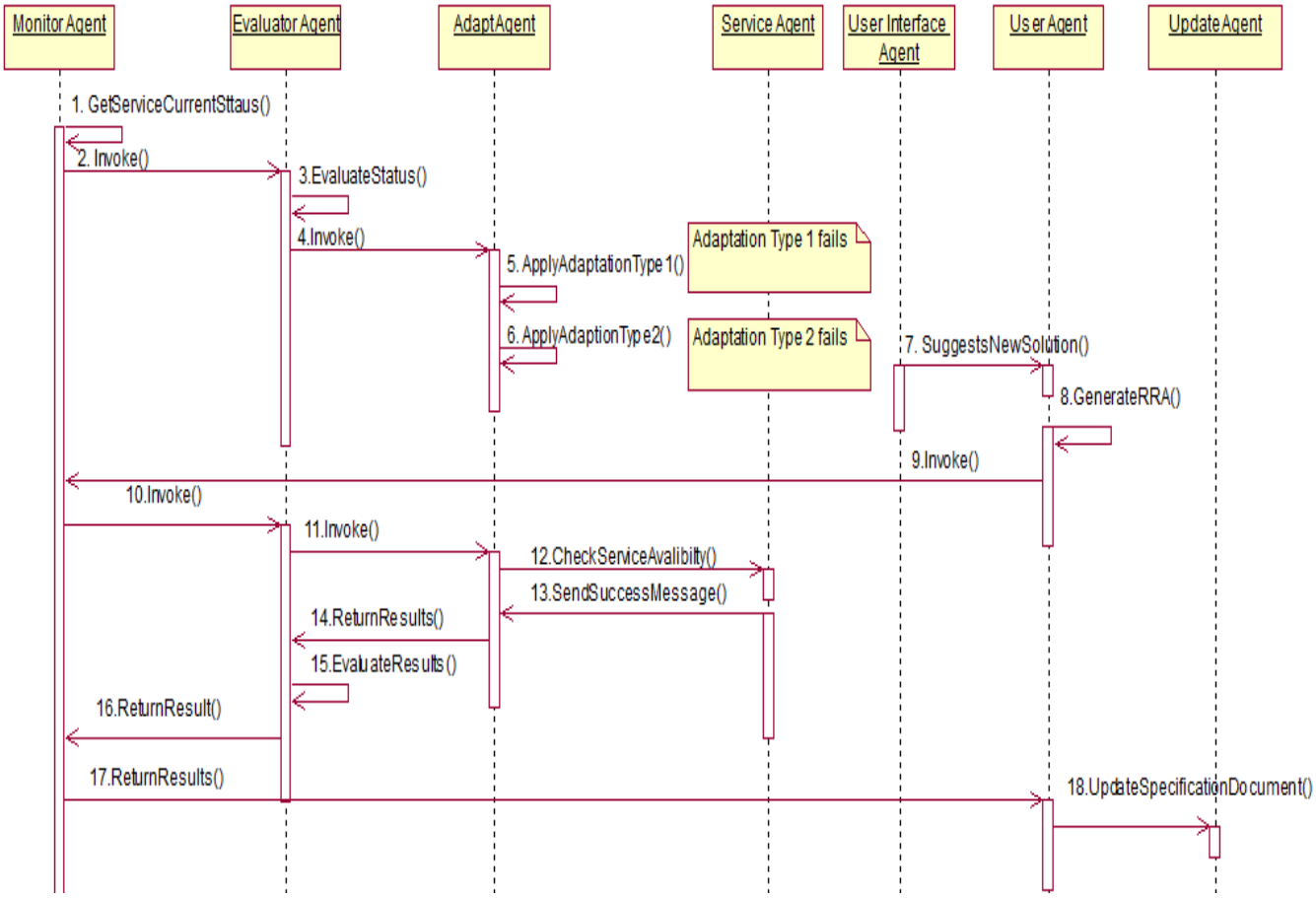


Figure 3.10: Sequence diagram for Type 3 Adaptation

Type 4 Adaptation	unanticipated changes where there is no possible solution or refinements exists
Scenario	1. During monitoring, Monitor agent gets current status of the selected service in the RRA and informs evaluator agent. 2. Evaluator agent evaluates the information about the status of the web service. In case of service unavailability, it informs adapt agent for further action. 3. Adapt agent first applies adaptation type 1 and asks service Agent to get alternative services. 4. Service agent informs adapt agent that no alternative service is available. 5. Adapt agent then applies adaptation type 2 and reads the requirement specification document to get alternative solutions against the

	functional goal. 6. Adapt agent passes results to evaluator. 7. Evaluator agent returns results to monitor agent. 8. Through user interface agent, monitor agent informs the user about service unavailability and asks to select one of the available alternative solutions or suggest her own solution. 9. User opts to suggest a new solution with a new plan and a service to operationalize the suggested plan. 10. User Interface agent informs User agent of CARE. 11. User Agent updates RRA and informs monitor agent. 12. Monitor agent calls evaluator who calls adapt agent. 13. Adapt agent asks service Agent for service invocation. 14. Service agent checks about the service availability. In case service is not available, it informs the adapt agent. 15. Adapt agent passes results to evaluator. 16. Evaluator agent returns results to monitor agent who informs the user agent. 17. User agent informs the user interface agent about unavailability of the service. It also calls the update agent. 18. Update Agent creates a log for off line evolution.

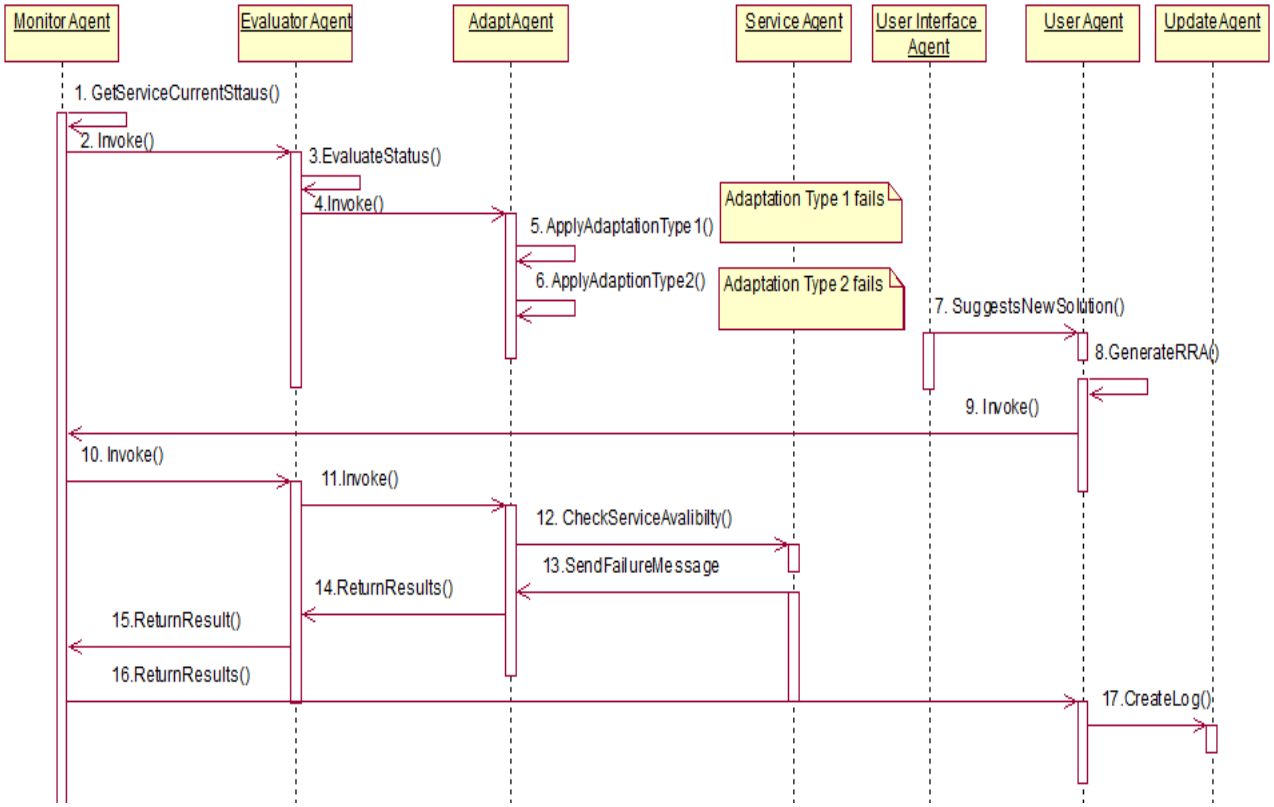


Figure 3.11: Sequence diagram for Type 4 Adaptation

3.4 Implementation

Care framework is developed using Eclipse Java EE IDE for Web Developers which is widely used to build Java applications. Version: Indigo Service Release 2 is used. It is available for free download from internet.

CARE framework is developed using java project option and then complied and exported as a JAR(Java ARchive) file. This JAR file can be added as a library in any android project.

In the following, implementation of different agents is explained with the help of algorithms which run on these agents.

Whenever there is a need to create new RRA then Invoke procedure of User agent is called. This procedure reads requirements specication documents, gets the corresponding data(service lists to operationalize user goals etc) and then write RRA. Finally it calls Monitor Agent for further actions. The algorithm for this procedure is given below:

begin procedure Invoke()

ReadRequirementSpecification ();

WriteRRA();

Result ← InvokeMonitorAgent();

Return Result;

end procedure;

The Invoke procedure of monitor Agent inserts context information (Current location), date and time etc. in the RRA and then calls Evaluator Agent for further action.

begin procedure InvokeMonitorAgent ()

Get UserCurrentLocation();

GetCurrentDateTime

UpdateRRA();

Result ← InvokeEvaluatorAgent();

Return Result;

end procedure;

Monitor agent also monitors the status of the service in an already operationalized RRA. In case of a service failure, or a change in the status of reservation, booking or meeting, it informs the user interface agent for further action. The algorithm for this procedure is given below:

begin procedure Monitor()

service ← ReadRRA();

webServiceStatus ← CallServiceAgent(service);

IF webServiceStatus=true **THEN**

Status ← checkStatus();//(it is the status of a reservation or booking or meeting etc.)

IF Status=false **THEN** informUserInterfaceAgent(AlternativeSolutions)

IF webServiceStatus=false **THEN**

alternativeServices ← CallServiceAgent(service)

IF alternativeServices.size>0 **THEN**

AlternativeServices ← SearchServiceRepository()

informUserInterfaceAgent(AlternativeServices)

Else if alternativeServices.size=0 **THEN**

AlternativeSolutions ← ReadRequirementSpecfication()

informUserInterfaceAgent(AlternativeSolutions)

END IF

end procedure;

In the following, the algorithm for Invoke() procedure of Adapt Agent is discussed. This is the procedure that is used by adapt agent to apply proper adaptation strategy at runtime.

begin procedure Invoke()

functionGoal, service ← ReadRRA();

serviceStatus ← InvokeServiceAgent(service);

IF serviceStatus= = true **THEN**

Result. returnCode=1;

Result.ReturnMessage=”Service available”;

Result.ReturnServiceList=service;

ELSE IF serviceStatus= =false **THEN**

AlternativeServiceList ← FindAlternativeServices();

IF AlternativeServiceList.size >0 **THEN**

Result. returnCode=2;

Result.ReturnMessage=”Alternative Services available”; Result.ReturnServiceList = AlternativeServiceList;

ELSE IF AlternativeServiceList.size=0 **THEN**

AlternativeSolutionsList ← ReadRequirementSpec ()

IF AlternativeServiceList().size>0 **THEN**

Result.returnCode=3;

Result .ReturnMessage=”Alternative Solutions available”

Result .ReturnServiceList = AlternativeSolutions;

ELSE IF AlternativeServiceList().size=0 **THEN**

Result. returnCode=4;

Result .ReturnMessage=”Adaptation type 3 required”;

Result .ReturnServiceList =null

END IF

CallEvaluatorAgent(Result);

end procedure;

Algorithms for two procedures CallServiceAgent() and FindCandidateServices() for Service agent is given below. CallServiceAgent() procedure a service as its paramet and simply checks whether the service is available or not It also returns the response time of the service.

Procedure CallServiceAgent(service**);**

GetServiceDetailsFromServiceRepository();

Long t1 ← GetSystemCurrentTime();

serviceStatus ←CallService();

ServiceResponseTime ← GetSystemCurrentTime-t1;

Return serviceStatus,ServiceResponseTime;

end procedure;

FindCandidateServices() takes a service as parameter, gets the alternative services available from the service repository in the same category, then calls these services to check for their availability. This procedure finally returns list of all the services from the repository which are available. The response time of each service is also returned.

Procedure FindCandidateServices(service**)**

```
serviceCategory ← GetServiceCategory(service);
serviceList ← serachRepository(serviceCategory)
        FOR(serviceList.Iterateror)
        BEGIN
        if(serviceList. Iterateror=service)
        removeServiceFromList();
        else
        serviceStatus ← callService();
        if(serviceStatus=”false”)
        removeServiceFromList();
        END FOR
        return serviceList;
end procedure;
```

4. CASE STUDY

4.1 Example Self Adaptive Application- iAssistant

To provide better assessment of CARE framework, it is instantiated through a service based application called Personal Assistant (iAssistant). iAssistant is a self-adaptive software application which assists the end user in managing their travelling related tasks(booking of flight, monitoring the status of flight etc.). It is deployed on the end user's smart phone.

.

iAssistant has a user friendly interface that guides end user in order to specify her preferences in easy steps. The main purpose of iAssistant is to support the end user by adapting to the changes in the end-user needs and environment through new candidate solutions. It is aware of end-users' goals and preferences. It achieves these goals by monitoring. Information about current context is automatically obtained by using the location sensing facility of the smartphone. It depends on third party services available over the Internet. In case where the end user's goals and preferences are not met, it looks for alternative solutions (e.g. booking is canceled, look for another booking). It involves the end-user by asking for her feedback. In short, at run-time, iAssistant acts as an analyst. It acquires user input about changed or new requirements. To operationalize these requirements, it then looks for available services as alternative ways. If iAssistant is unable to meet a goal or preferences then it creates a log which can be used later on for the off line evolution.

It also informs the end-users time to time through notification messages (about the status of the flight i.e. confirmed, canceled or in progress). If any change is found through monitoring, then it takes appropriate action accordingly. For example, if the currently selected web service is not available, then it searches for alternative services in the same domain and ask user's feedback to choose one of them. Similarly, in case a booking (of flight etc.) is cancelled or delayed then again it informs the user and suggests some alternative means of travelling.

4.2 Goal Model of IAssistant

The goal model for IAssistant is developed with the help of ARML[29]. Following figure shows the mandatory root level goal 'Manage Travel' which is further decomposed into other goals 'Find means of transportation', 'Display Options', 'Book' and 'Send Confirmation'. All these goals are mandatory which means all four must be met in order to achieve system overall goal "Manage Travel". This is represented with AND inference relationship.

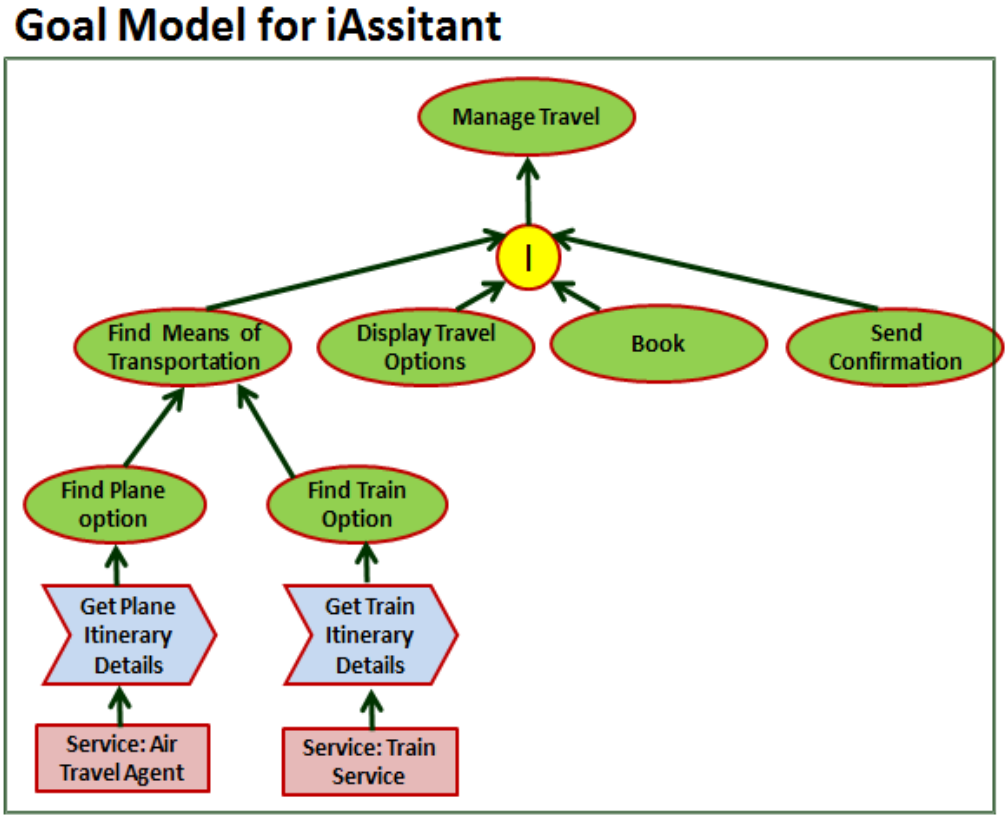


Figure 4.1: Goal Model for IAssistant

'Find means of transportation' is further decomposed into other goals 'Find Plane Options' and 'Find Train Options'. Either of the two can be used to achieve goal 'Find means of transportation'. This is represented with OR inference relationship.

'Find Plane Options' is operationalized using Task 'Get Plane Itinerary Details' which requires a flight service (resource).

'Find Plane Options' is operationalized using Task 'Get Train Itinerary Details' which requires a Train service (resource).

Based on this goal model, at design time a requirements specification document was developed at design time in xml which is given as an input to CARE framework.

4.3 Implementation of iAssistant

To build Android applications, Android Development Tools (ADT) is a plugin that is used for the Eclipse IDE. It provides an integrated environment where powerful android apps can be developed

quickly. Compatible versions of both the Eclipse IDE and the Android SDK must be installed on machine before installing or using ADT. iAssistant application is developed using Android Development Tools (ADT). The CareFrameworkAPI.jar file is included so that iAssistant Application can utilize CARE agents to support runtime.

The web services required for these two applications were developed in Microsoft Visual Studio 2010 using .Net technology and the database accessed by the web services was developed using SQL server Management Studio 2008. Following six web services in the travel domain have been developed:

TABLE 4.1: WEB SERVICES

Category	***Service name***
Flight	*Air Travel Agent*
	Flight Companion
Train	*Train Companion*
	Train Service
Bus	*Road Liner Agency*
	Bus Service Agency

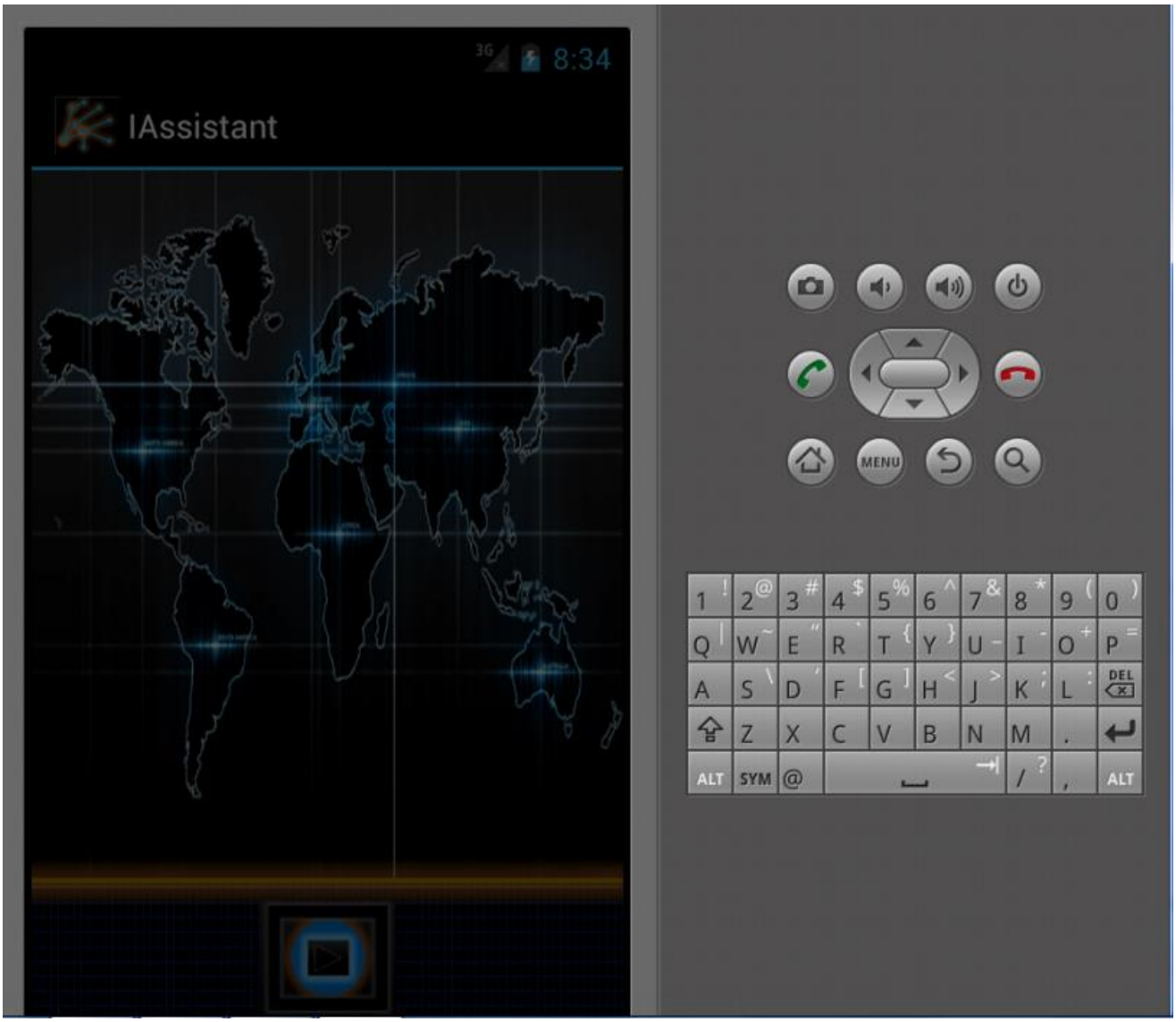
These services are configured in Internet Information Services (IIS) Manager. The information about how to access these services is saved category-wise in Service Repository in the form of xml file which format is given below:

```
<?xml version="1.0" encoding="UTF-8"?>
- <services>
    <Service endpoint="GetWebServiceStatus" uri="http://10.0.2.2:81/AirTravelAgent/Service1.asmx"
      namespace="http://tempuri.org/" type="soap" category="Flights" name="Air Travel Agent"/>
    <Service endpoint="GetWebServiceStatus" uri="http://10.0.2.2:83/FlightCompanion/Service1.asmx"
      namespace="http://tempuri.org/" type="soap" category="Flights" name="Flight Companion"/>
    <Service endpoint="TestService" uri="http://10.0.2.2:86/TrainService/Service1.asmx"
      namespace="http://tempuri.org/" type="soap" category="Trains" name="Train Service"/>
    <Service endpoint="TestService" uri="http://10.0.2.2:85/TrainCompanion/Service1.asmx"
      namespace="http://tempuri.org/" type="soap" category="Trains" name="Train Companion"/>
    <Service endpoint="GetWebServiceStatus" uri="http://10.0.2.2:84/RoadLinerAgency/Service1.asmx"
      namespace="http://tempuri.org/" type="soap" category="Buses" name="Road Liner Agency"/>
    <Service endpoint="GetWebServiceStatus" uri="http://10.0.2.2:82/BusServiceAgency/Service1.asmx"
      namespace="http://tempuri.org/" type="soap" category="Buses" name="Bus Service Agency"/>
  </services>
```

Services are saved category-wise (Flights, Trains, and Buses etc.) in this repository. Information about services can be added, deleted or updated.

4.4 Graphical User Interface of iAssistant

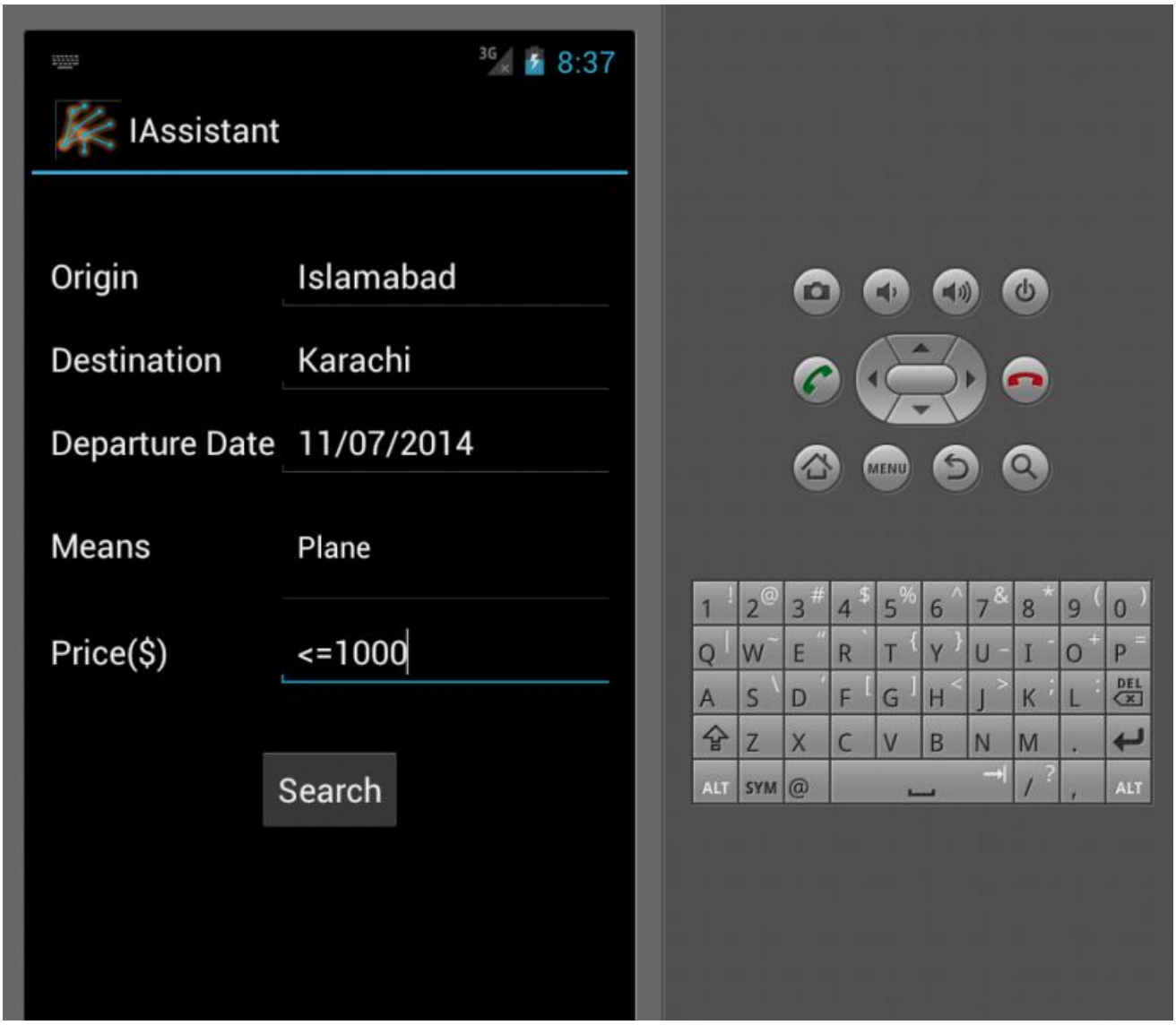
iAssistant is a service based android app which can be installed on end user smart phone. Themain user interface of App is given below:



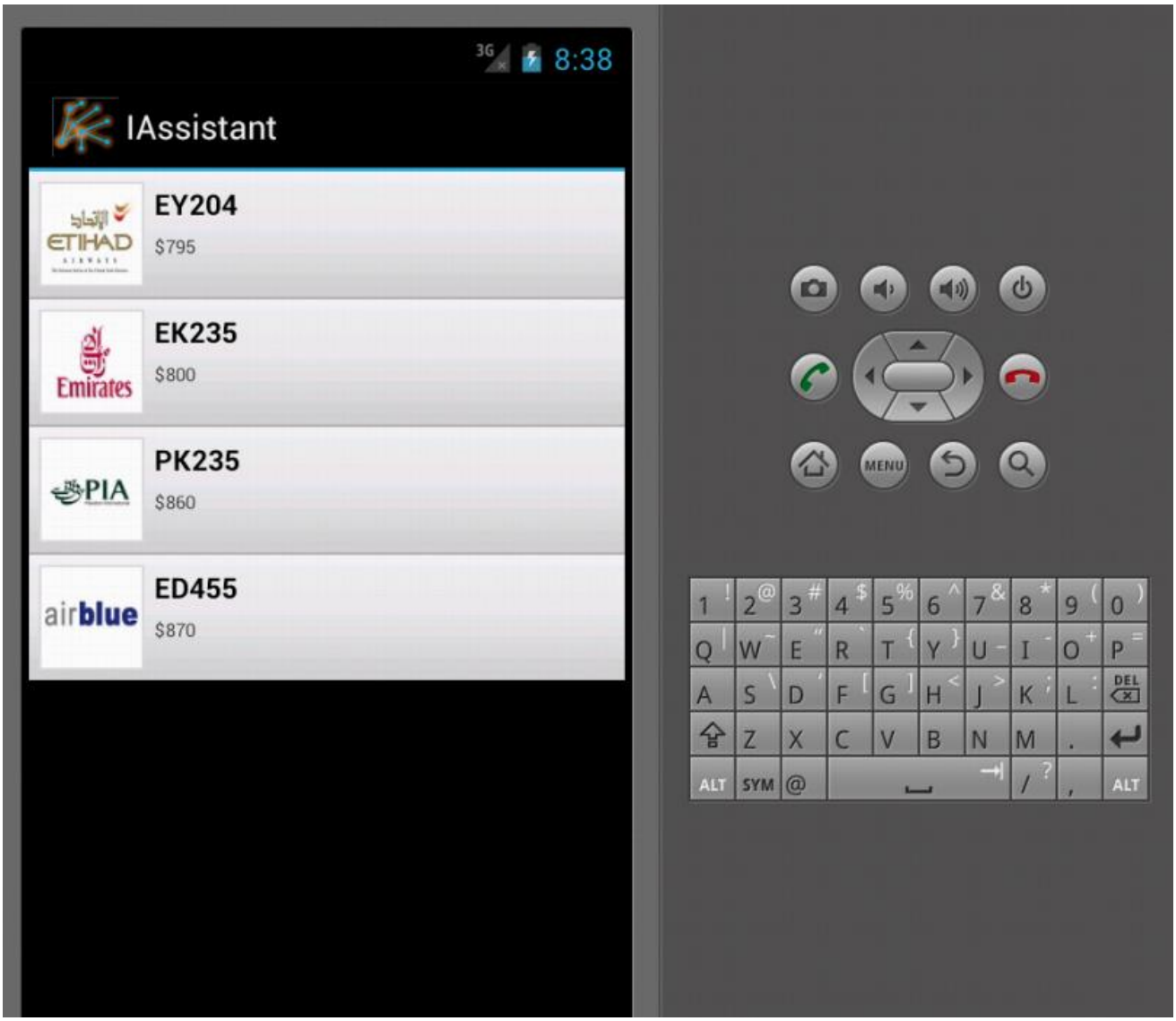
This App will help the end user to manage her travel related activities. For example, End User can search for the flight options by entering:

- Travel Origin
- Travel Destination
- Departure Date and Time

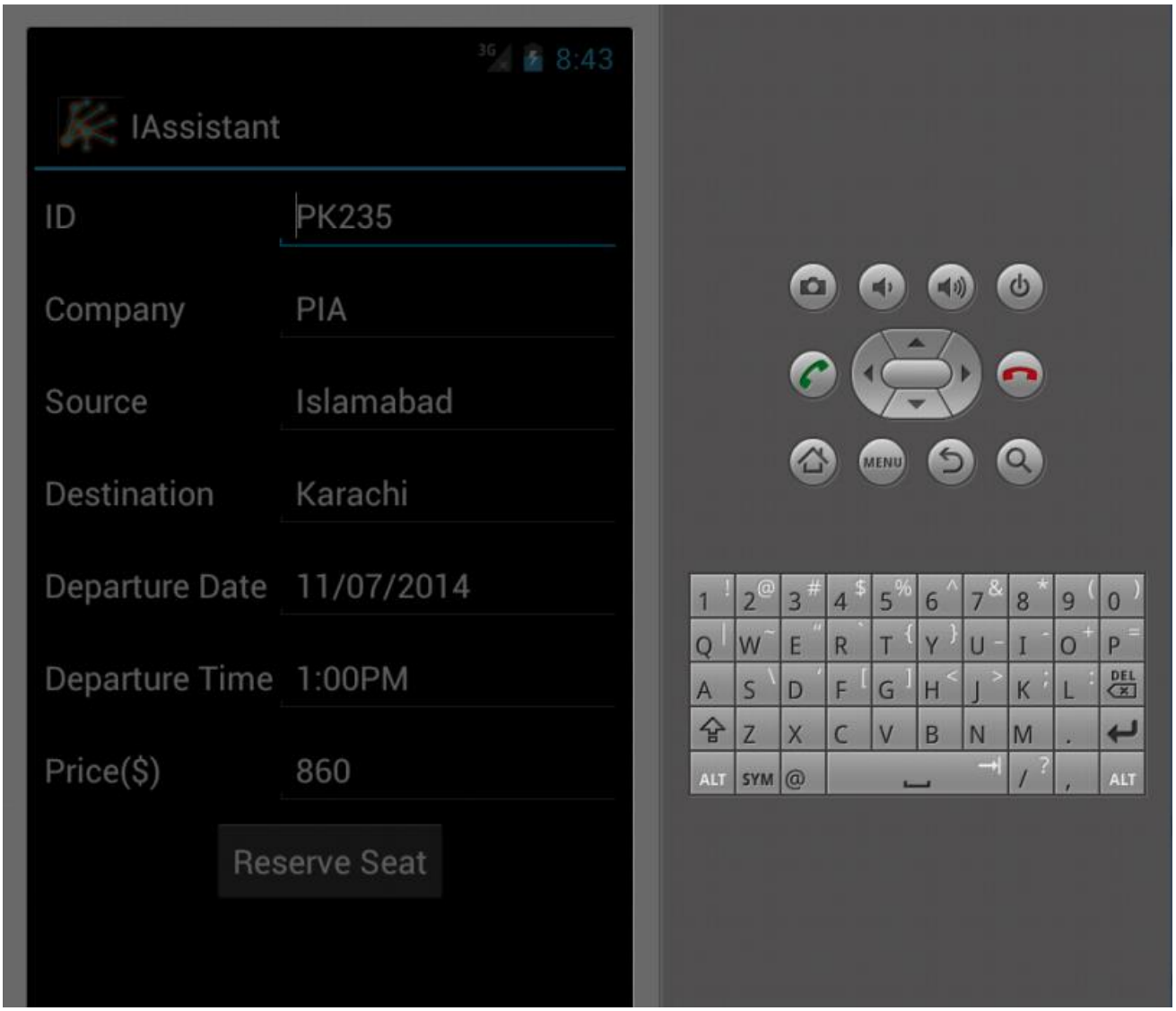
End user can select one of the available means of transportation and can give her preferences (cost).



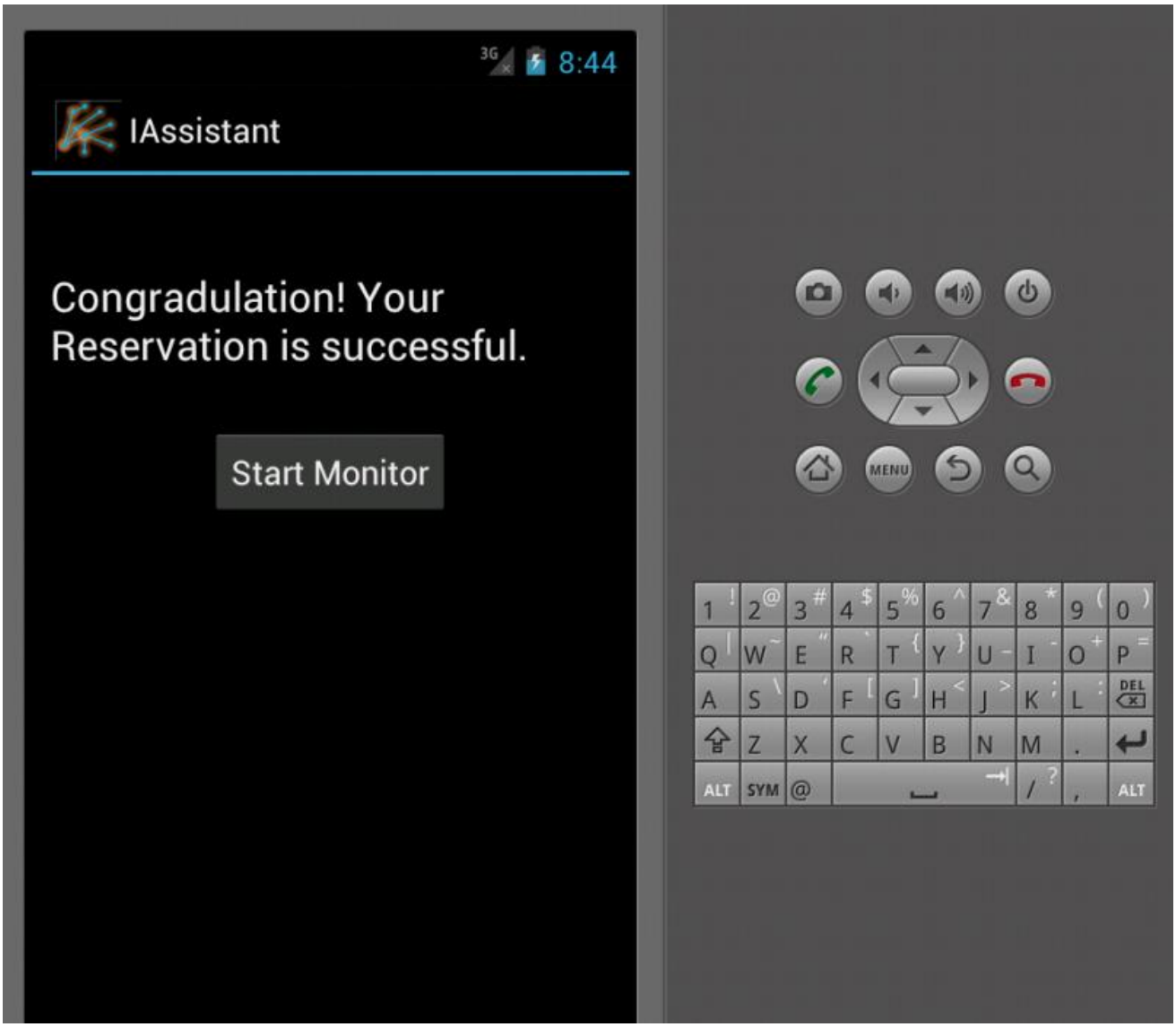
Once the Search button is pressed, different flight options will be shown to the user based on her search criteria.



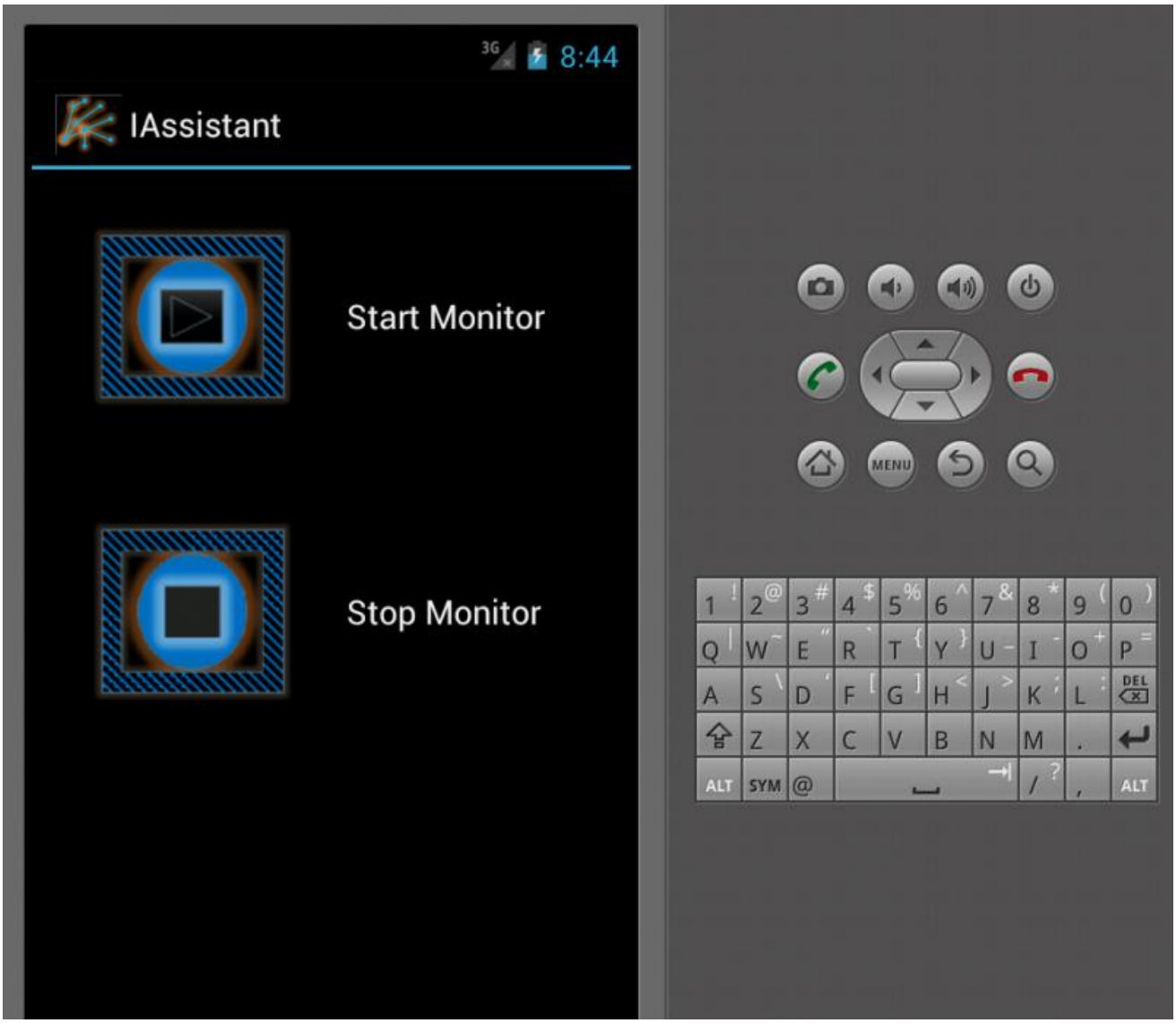
User can select one of the available options to view further details and reserve a seat.


8:43
IAssistant
ID
PK235
Company
PIA
Source
Islamabad
Destination
Karachi
Departure Date
11/07/2014
Departure Time
1:00PM
Price($)
860
Reserve Seat
MENU

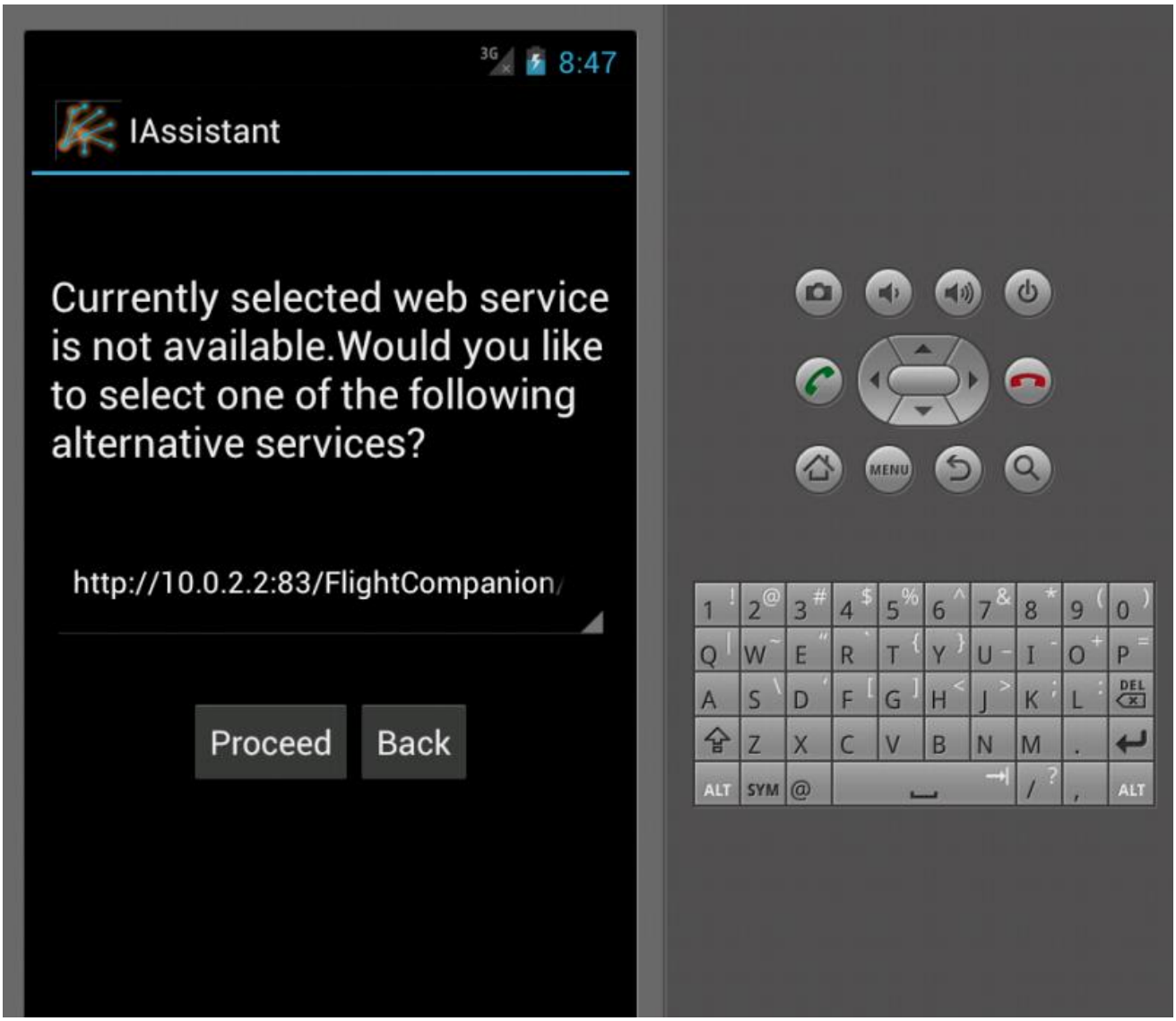
If the reservation is done successfully, then a success message is displayed.



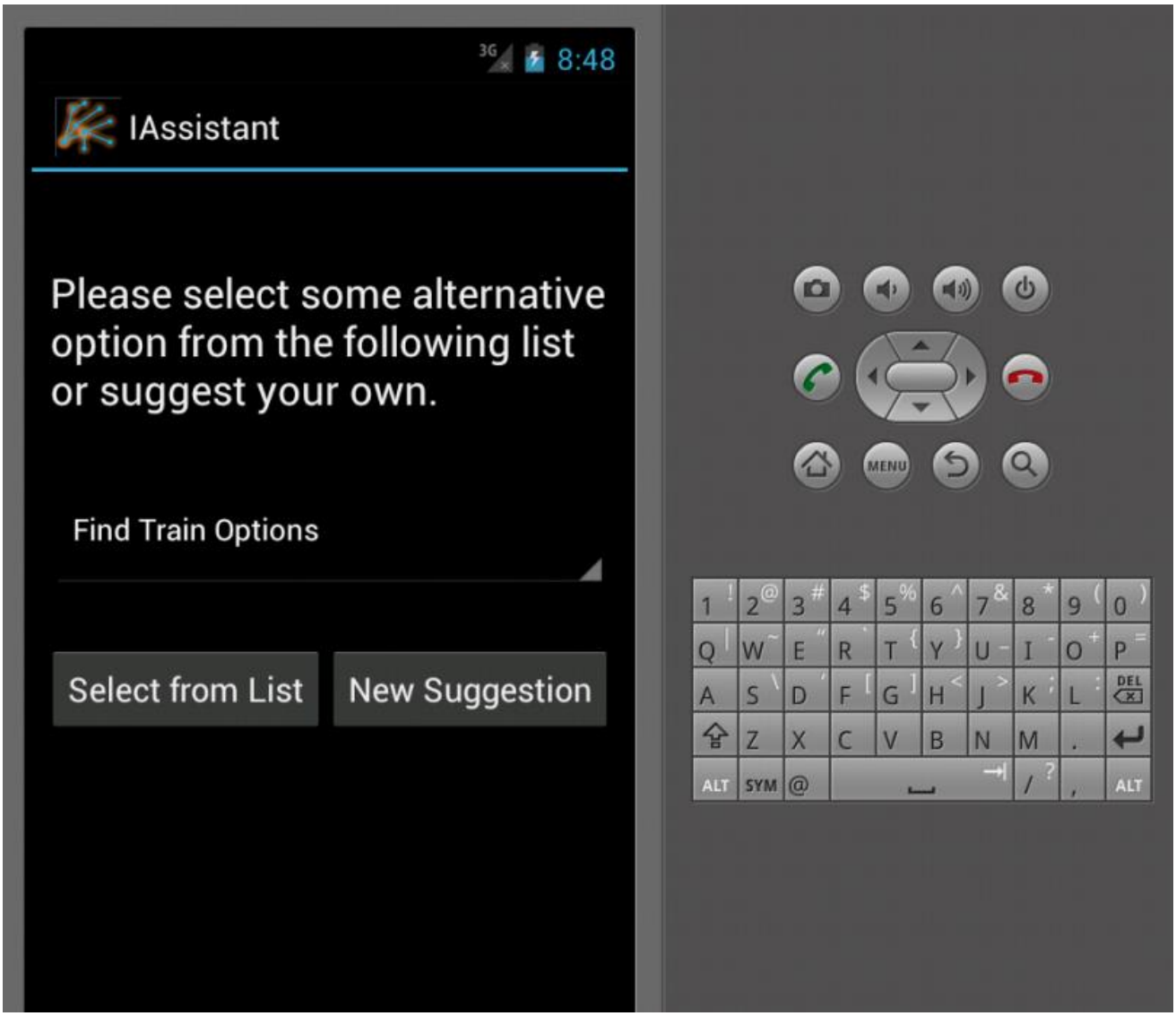
At this stage iAssistant will ask user whether she would like iAssistant to monitor the status of the booking. If user selects Yes, then iAssistant will start monitoring.



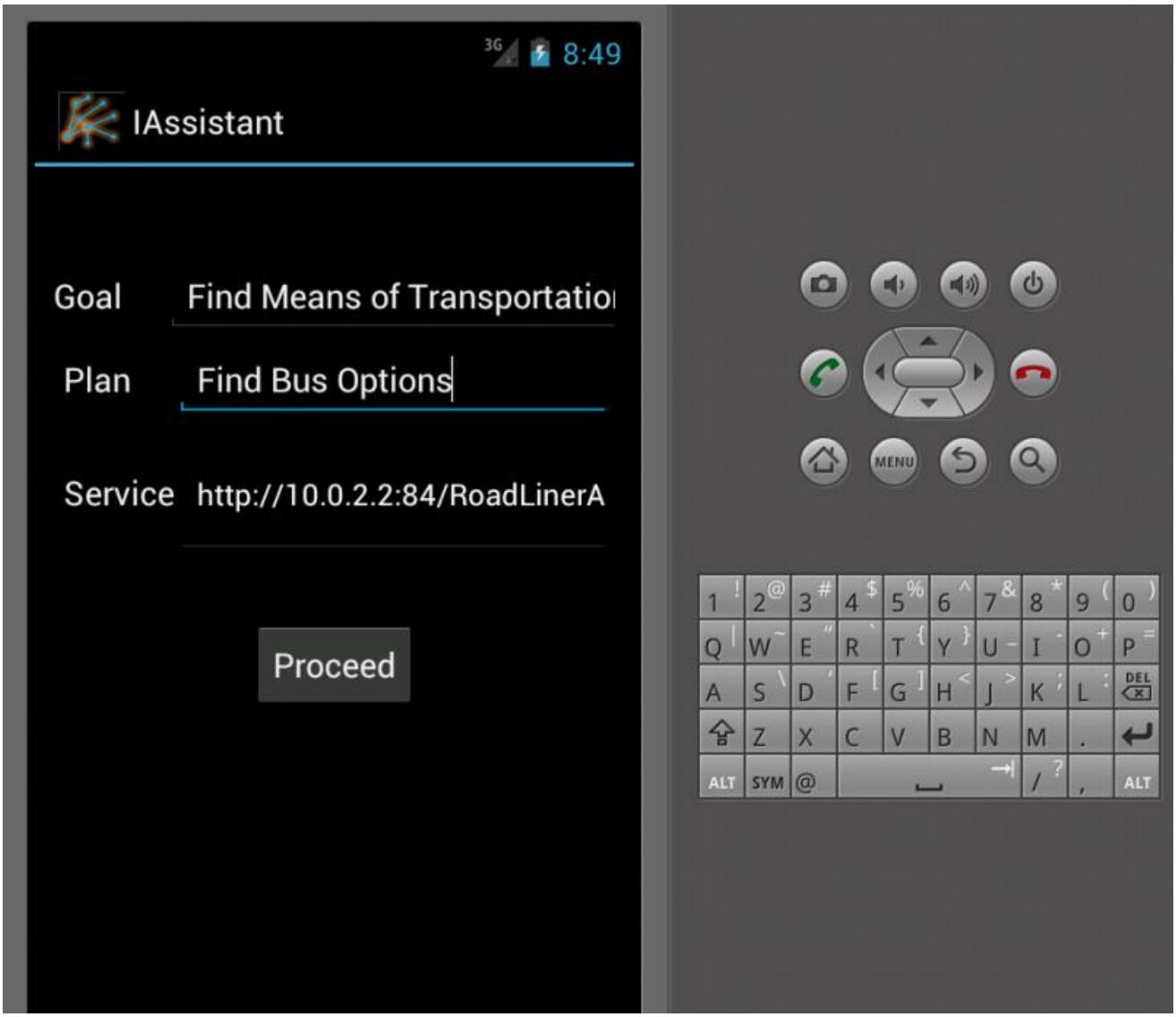
In case of currently selected services failure, iAssistant will suggest alternative services available for searching.



In case of cancelation or delay of flight, iAssistant will present alternative solutions to travel. If user selects an alternative solution, then iAssistant shows the different options and their details.



If user opts to suggest a new option, then an interface will be shown where he can specify her plan achieve the functional goal. She can also select one of the available service from service repository.



In case user suggested option is feasible then the options along with details are displayed, otherwise a failure message is shown.

5. EVALUATION OF CARE FRAMEWORK

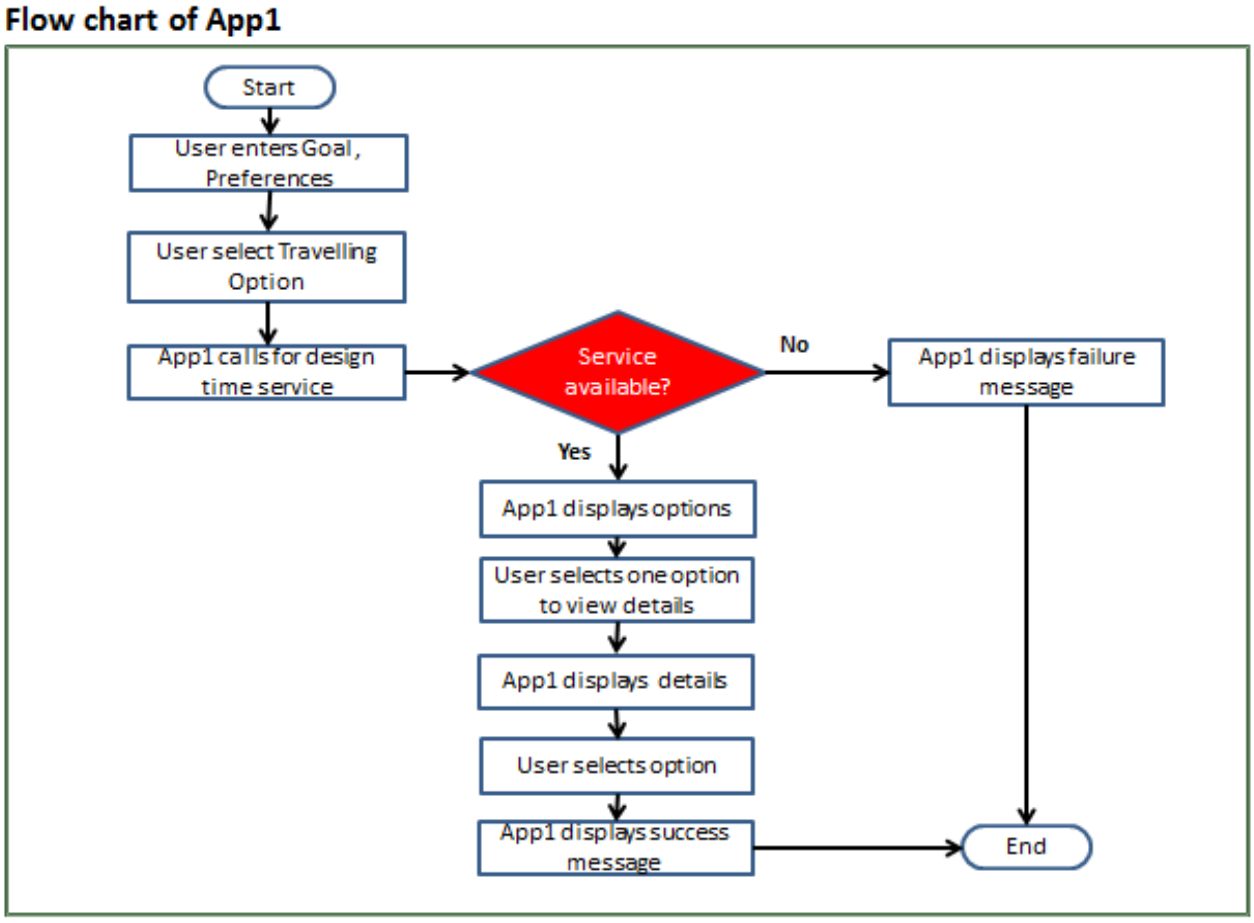
5.1 Service based android applications

To provide better assessment of CARE framework, two service based android applications have been developed using Android Development Tools (ADT).

The detail about two android applications used for evaluation is given below:

5.1.1 App1

App1 does not use CARE framework. It helps the user in booking by using design time options. It has no monitoring capability to detect whether a service is failed or a booking or meeting is cancelled. How this App works is explained using following flow chart:



.

Figure 5.1: Flow chart for App 1

5.2.2 App2(iAssistant)

App2 is the service based application(iAssistant) given in SECTION 4. The flow chart of activities performed by this app is explained with the help of following flow chart:

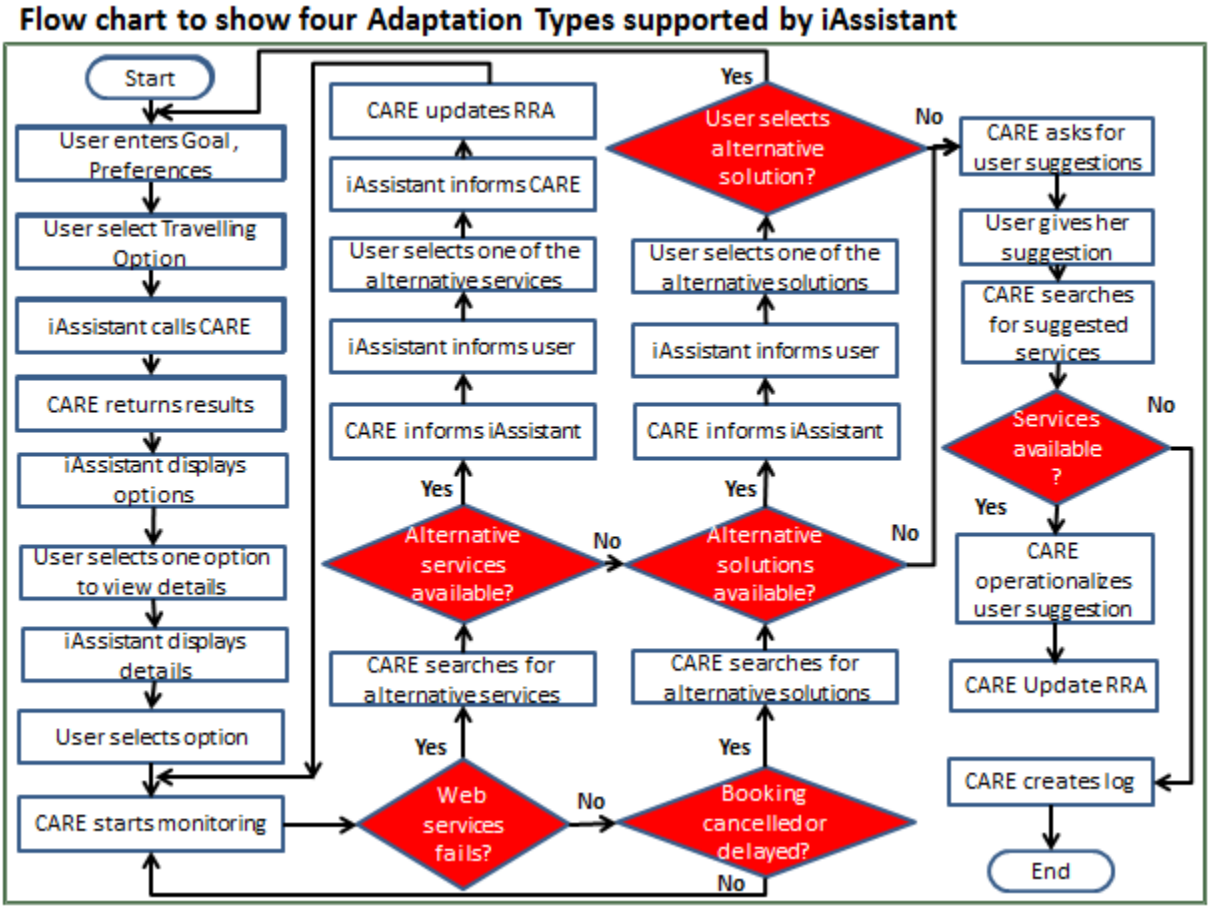


Figure 5.2: Flow chart iAssistant

In following table, a simple scenario is discussed where iAssistant with the help of CARE framework helps end user to reserve a flight seat.

TABLE 5.1: SCENARIO FOR SEAT RESERVATION

Scenario	1. Sue decides to plan a trip for her vocation. Using iAssistant user interface, she enters her preferences (Origin, Destination, Departure Date & Time Ticket Cost, means of transportation) for the trip. She opts to travel by air. 2. iAssistant calls CARE framework to operationalize user preferences and passes function goal and user preferences to it. 3. CARE searches for the service that can operationalize user preferences by using requirement specification document and returns the service *(air travel agent)* details

	to iAssistant. 4. Using this service, iAssistant shows the possible Flights options from different airline companies. 5. She clicks one of them to find further details. 6. iAssistant displays the required details. 7. She chooses to select one of the displayed options. 8. On behalf of her, iAssistant makes a booking and then a confirmation message is sent to her.' 9. IAssistant asks her whether she would like iAssistant to monitor the flight status. 10. She clicks yes. 11. iAssistant asks CARE to start monitoring. 12. Monitor Agent of CARE starts monitoring the flight status using the service.

By applying different scenarios, CARE framework is evaluated successfully thorough iAssistant Application that it supports all four types of adaptation at runtime. Detail is given below:

TABLE 5.2: SCENARIO FOR ADATATION TYPE 1

Scenario	1. During monitoring, Monitor agent of CARE detects that the web service(air travel agent) , it is using for monitoring flight status is no more available. 2. CARE informs iAssistant about service failure along with alternative web services available(in this case Flight Companion) . 3. iAssistant informs the user about service failure and asks her whether she would like to select one of the alternative services. 4. User selects an alternative service(Flight companion). 5. iAssistant passes this service to CARE. *6.* CARE updates RRA and monitor agent of CARE starts monitoring flight status using this new service.

TABLE 5.3: SCENARIO FOR ADATATION TYPE 2

Scenario	1. During monitoring, Monitor agent of CARE detects that the flight has been

	cancelled due to some reasons. 2. CARE gets alternative solutions (in this case, "Find Train options") from Requirements specification document. 3. CARE informs iAssistant about flight cancellation and suggests the alternative solutions. 4. iAssistant informs the user about it and asks her whether she would like to select some alternative solution or suggest her own solution. 5. User selects alternative solution (Find Train options). 6. iAssistant asks CARE to operationalize it. 7. CARE looks for a service corresponding to alternative solution (Find Train Options)in requirement specification and successfully returns the best candidate service(Train Companion) to iAssistant. 8. Using this service, iAssistant shows the possible Train options. 9. She selects one of the options. 10. On behalf of her, iAssistant makes a booking and then a confirmation message is sent to her.' 11. CARE updates RRA and its monitor agent starts monitoring the status of new reservation.

TABLE 5.4: SCENARIO FOR ADATATION TYPE 3

Scenario	1. During monitoring, Monitor agent of CARE detects that the flight has been cancelled due to some reasons. 2. CARE gets alternative solutions (in this case, "Find Train options") from Requirements specification document. 3. CARE informs iAssistant about flight cancellation and suggests the alternative solutions. 4. iAssistant informs the user about it and asks her whether she would like to select some alternative solution or suggest her own solution. 5. User suggests a new solution (Find Bus Options). This option was not anticipated at design time. 6. User also suggests a web service(Road Liner Agency) that can be used to

	operationalize this new solution.. 7. iAssistant asks CARE to operationalize it. 8. CARE looks for suggested service details corresponding to suggested solution (Find Bus Options) in service repository 9. If the service is available then CARE informs iAssistant. 10. Using this service, iAssistant shows the possible Bus options available to user. 11. She selects one of the options. 12. On behalf of her, iAssistant makes a booking and then a confirmation message is sent to her.' 13. CARE updates RRA and start monitoring the status of new reservation. 14. CARE also update the requirement specification file by adding a new goal" Find Bus Options" under the goal "Find means of transportation" along with the service for future use.

TABLE 5.5: SCENARIO FOR ADATATION TYPE 4

Scenario	1. During monitoring, Monitor agent of CARE detects that the flight has been cancelled due to some reasons. 2. CARE gets alternative solutions from Requirements specification file (Find Train options). 3. CARE informs iAssistant about flight cancellation and suggests the alternative solutions. 4. iAssistant informs the user about it and asks her whether she would like to select some alternative solution or suggest her own solution. 5. User suggests a new solution (Find Taxi Options). This option was not anticipated at design time. 6. iAssistant asks CARE to operationalize it. 7. CARE searches service repository to find some service against the suggested solution but it fails. 8. CARE informs iAssistant about this and creates a log for offline evolution.

5.2.3 Comparison of App1 and App2

Table 7 compares App1 and App2. It clearly depicts that App1 is a non-adaptive application whereas App2 is an adaptive application and supports all four types of adaptations.

TABLE 5.6: COMPARISON OF APP1 AND APP2

	Adaptation Type 1	***Adaptation Type 2***	***Adaptation Type 3***	***Adaptation Type 4***
App1	*No*	*No*	*No*	*No*
App2(IAssistant)	*Yes*	*Yes*	*Yes*	*Yes*

5.2.3 Performance evaluation of CARE framework

Performance of CARE framework is evaluated through App2 for adaptation activities (Monitoring, Analysis, Planning and Executing) against each of four adaptation types. The following graph shows the elapsed time for each activity.

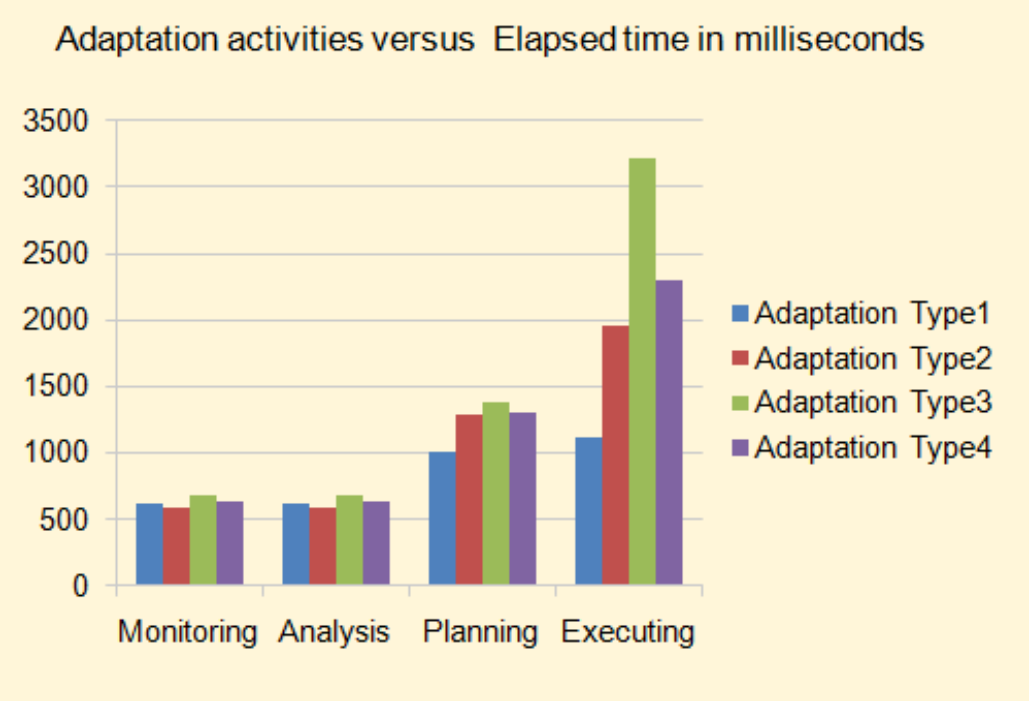


Figure 5.3: Performance Evaluation 1

During monitoring activity, the status of the currently selected web services or status of the reservation is checked, so it is almost same for all four types of adaptations.

During analysis, the status monitored by the monitor agent is analyzed by the evaluator agent and then it calls the adapt agent if required, so again the time taken by this activity is almost same for all four types of adaptation.

The time taken for planning in adaptation type 1 is least as it requires alternative services discovery whereas for adaptation type 2,3 and 4, the additional time is required for searching requirement specification document and finding alternative solutions.

Time taken by executing activity for Adaptation type 3 is longest. The reason is that it includes time to search for user suggested services and then updating the requirement specification document accordingly. For adaptation type 4, it is slightly less as it excludes the time required for updating the requirement specification document, although it includes time to create the log. The time taken by executing activity for adaptation type 1 is least as during this activity, only RRA is updated.

The following graph compares the total time taken by each adaptation type in milliseconds. For example, total time taken by all four activities of adaptation (Monitoring, Analysis, Planning and Executing) for type 1 adaptation is 1sec. This graph clearly depicts that minimum time is required for applying adaptation type 1 and maximum time for adaptation type 3.

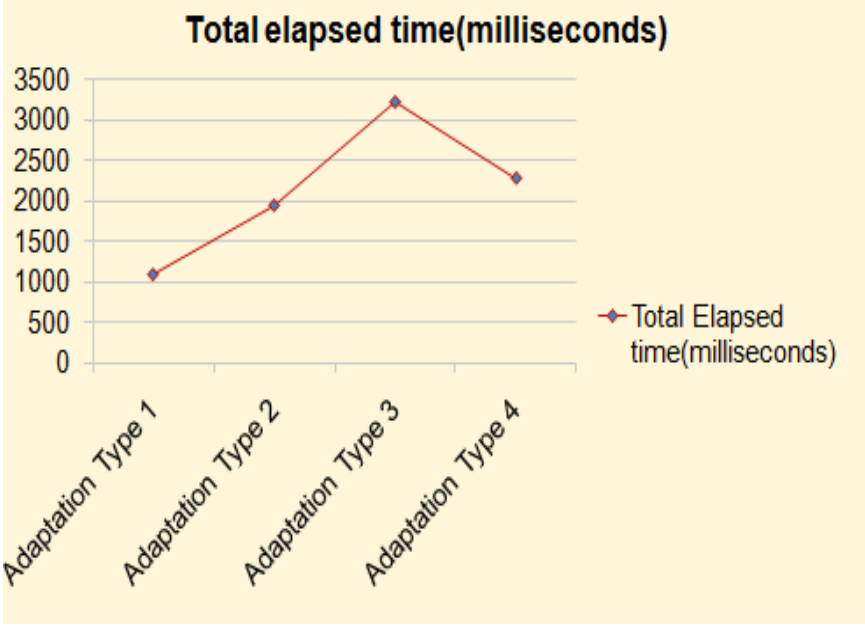


FIGURE 5.4: PERFORMANCE EVALUATION 2

6. CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

To achieve self adaptivity in software systems is an exciting emerging area among software engineering community for past decade. The future software systems are envisioned as self-adaptive software systems. There is a dire need to introduce new techniques, tools and frameworks for this purpose. A novel conceptual architecture for SBA which is already proposed under the title of 'Continuous Adaptive Requirements Engineering (CARE) framework' but some refinements and its implementation was yet to be done which is the main motivation for this research work.

The main objective of this research was to implement a framework (CARE) that can support self-adaptability at runtime for service based applications by using efficient methods and techniques. As a result, a general purpose middleware was developed that supports runtime adaptation for any self-adaptive service based application in a user-oriented and goal-driven manner. The main functionality which is provided by this framework includes:

- Capturing change in user preferences and context at runtime
- Finding suitable web services accordingly
- Selecting the best candidate web service
- Updating the requirements

Also, the framework captures user requirements at runtime by a wizard like easy to use interface through which end users can easily specify their needs by following simple and easy steps. And then convert these needs into a machine understandable format for further processing.

In summary, the main objectives achieved during this research were:

- Implementation of CARE framework that supported self-adaptability at runtime for service based applications
- Instantiation of CARE framework through prototype service based application to provide better assessment for assessment and evaluation

6.2 FUTERE WORK

For self-adaptive service based applications, change is inevitable. Capturing and satisfying user requirements at run time have appeared a new challenge in the field of Requirements Engineering. Not much research is done in this direction. Requirements engineering for self-adaptive applications is still at its early stages. In future, novel methods and techniques are required in order to support run-time adaptation as the existing approaches do not support the accommodation of new or changed requirements. CARE framework' for capturing user requirements/preferences and system adaptation at runtime has been presented during this research. This section identifies some interesting future directions which are given below:

- CARE architecture consists of multiple agents. It helps the end-users in requirements specification for support of constant requirements reappraisal at runtime by selecting suitable web services with least human intervention. Currently there is no web search engine available that can help CARE for autonomic web service discovery. CARE searches a web service repository where information for web services discovery and invocation is already stored. It will be a great idea if in future CARE framework is integrated with such a web service search engine for automatic service discovery, invocation and integration.

- CARE framework introduced an API of common functions for service based applications so that there is no need to re-create them for every new service based application. It is also instantiated through a service based mobile application called iAssistant for assessment and evaluation. In future, to provide better assessment of framework, more prototype applications needs to be developed which is the focus of further research.

7. REFERENCES

[1] Cheng, Betty HC, et al. "Software engineering for self-adaptive systems: A research roadmap. "*Software engineering for self-adaptive systems*. Springer Berlin Heidelberg, 2009. 1-26.

[2] De Lemos, Rogério, et al. "Software engineering for self-adaptive systems: A second research roadmap." *Software Engineering for Self-Adaptive Systems II*. Springer Berlin Heidelberg, 2013. 1-32.

[3] Kephart, Jeffrey O., and David M. Chess. "The vision of autonomic computing." *Computer* 36.1 (2003): 41-50.

[4] Salehie, Mazeiar, and Ladan Tahvildari. "Towards a goal-driven approach to action selection in self-adaptive software." *Software: Practice and Experience* 42.2 (2012): 211-233.

[5] Macías-Escrivá, Frank D., et al. "Self-adaptive systems: A survey of current approaches, research challenges and applications." *Expert Systems with Applications* 40.18 (2013): 7267-7279.

[6] Garlan, David, et al. "Rainbow: Architecture-based self-adaptation with reusable infrastructure." *Computer* 37.10 (2004): 46-54.

[7] Geihs, Kurt, et al. "A comprehensive solution for application-level adaptation." *Software: Practice and Experience* 39.4 (2009): 385-422.

[8] Yeom, Kiwon, and Ji-Hyung Park. "Morphological approach for autonomous and adaptive systems based on self-reconfigurable modular agents." *Future Generation Computer Systems* 28.3 (2012): 533-543.

[9] Chen, Toly. "A self-adaptive agent-based fuzzy-neural scheduling system for a wafer fabrication factory." *Expert Systems with Applications* 38.6 (2011): 7158-7168.

[10] Kramer, Jeff, and Jeff Magee. "Self-managed systems: an architectural challenge." *Future of Software Engineering, 2007. FOSE'07*. IEEE, 2007.

[11] Bencomo, Nelly, et al. "Genie: Supporting the model driven development of reflective, component-based adaptive systems." *Proceedings of the 30th international conference on Software engineering*. ACM, 2008.

[12] Qureshi, Nauman A., and Anna Perini. "Requirements engineering for adaptive service based applications." *Requirements Engineering Conference (RE), 2010 18th IEEE International*. IEEE, 2010.

[13] Qureshi, Nauman A., and Anna Perini. "Continuous adaptive requirements engineering: An architecture for self-adaptive service-based applications." *Requirements@ Run. Time (RE@ RunTime), 2010 First International Workshop on*. IEEE, 2010.

[14] Patikirikorala, Tharindu, et al. "A systematic survey on the design of self-adaptive software systems using control engineering approaches." *Software Engineering for Adaptive and Self-Managing Systems (SEAMS), 2012 ICSE Workshop on*. IEEE, 2012.

[15] Tesauro, Gerald, et al. "A multi-agent systems approach to autonomic computing." *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 1*. IEEE Computer Society, 2004.

[16] Chen, Toly. "A self-adaptive agent-based fuzzy-neural scheduling system for a wafer fabrication factory." *Expert Systems with Applications* 38.6 (2011): 7158-7168.

[17] Leitão, Paulo, José Barbosa, and Damien Trentesaux. "Bio-inspired multi-agent systems for reconfigurable manufacturing systems." *Engineering Applications of Artificial Intelligence* 25.5 (2012): 934-944.

[18] Brun, Yuriy. "Building Biologically-Inspired Self-Adapting Systems." *Software Engineering for Self-Adaptive Systems* 8031 (2008).

[19] Clement, Lauren, and Radhika Nagpal. "Self-assembly and self-repairing topologies." *Workshop on Adaptability in Multi-Agent Systems, RoboCup Australian Open*. 2003.

[20] Di Nitto, Elisabetta, et al. "A journey to highly dynamic, self-adaptive service-based applications." *Automated Software Engineering* 15.3-4 (2008): 313-341.

[21] Fickas, Stephen, and Martin S. Feather. "Requirements monitoring in dynamic environments." *Requirements Engineering, 1995., Proceedings of the Second IEEE International Symposium on*. IEEE, 1995.

[22] Cohen, Don, et al. "Automatic monitoring of software requirements." *Proceedings of the 19th international conference on Software engineering*. ACM, 1997.

[23] Seyff, Norbert, Florian Graf, and Neil Maiden. "End-user requirements blogging with iRequire." *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 2*. ACM, 2010.

[24] Ollmann, Norbert Seyff Gregor, and Manfred Bortenschlager. "iRequire: Gathering end-user requirements for new apps." (2011).

[25] Qureshi, Nauman A., Norbert Seyff, and Anna Perini. "Satisfying user needs at the right time and in the right place: a research preview." *Requirements Engineering: Foundation for Software Quality*. Springer Berlin Heidelberg, 2011. 94-99.

[26] Qureshi, Nauman A., et al. "Towards a continuous requirements engineering framework for self-adaptive systems." *Requirements@ Run. Time (RE@ RunTime), 2010 First International Workshop on*. IEEE, 2010.

[27] Qureshi, Nauman A., and Anna Perini. "Engineering adaptive requirements." *Software Engineering for Adaptive and Self-Managing Systems, 2009. SEAMS'09. ICSE Workshop on*. IEEE, 2009.

[28] Sawyer, Peter, et al. "Requirements-aware systems: A research agenda for re for self-adaptive systems." *Requirements Engineering Conference (RE), 2010 18th IEEE International*. IEEE, 2010.

[29] Qureshi, Nauman A., Ivan J. Jureta, and Anna Perini. "Towards a requirements modeling language for self-adaptive systems." *Requirements Engineering: Foundation for Software Quality*. Springer Berlin Heidelberg, 2012. 263-279.