SITUATION AWARE MOBILE APPS FRAMEWORK

A THESIS IN
Computer Science

Presented to the Faculty of the
University of Missouri - Kansas City
in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

by

FEICHEN SHEN

B.S. NANJING UNIVERSITY, CHINA 2010

Kansas City, Missouri

2012

SAMAF: SITUATION AWARE MOBILE APPS FRAMEWORK

Feichen Shen, Candidate for the Master of Science Degree

University of Missouri – Kansas City, 2012

ABSTRACT

Mobile devices, like smart phones or tablets, have become ubiquitous, with their adoption being driven by their immediacy and sensing capabilities. Applications, or apps, that run on portable computing devices have surged in popularity, with billions of downloads taking place. However, an increasing number of mobile apps and their diverse users make it difficult to select the correct app to respond to evolving situations. To address this issue, it is vitally important to find an intelligent approach to provide situation awareness capabilities and an immediate response to the changes.

In this thesis, we have developed a semantic framework for mobile apps named the Situation Awareness Mobile Apps Framework (SAMAF) to achieve the goal of dynamic and adaptive apps for automated composition, adaptation, and evolution of software systems responding to the mobile users' context and environmental changes. SAMAF is composed of two major components: i) a cloud based service framework for mobile apps development, deployment, and adaptation using a design of dynamic patterns for Service Oriented Architecture and ii) an ontology-based context modeling and reasoning framework that is implemented based on Context Ontology modeling and Event Condition Action (ECA) rule based inference to align the adaptation with the changes.

The SAMAF framework has been evaluated by two kinds of experiments. One was conducted in real phone settings to obtain the running performance of mobile apps adapting to dynamic changes of the users' contexts. The other was performed with a large number of mobile phone users in a simulated JADE (Java Agent DEvelopment Framework), multiple agents' platform for testing the adaptability, reasoning correctness, and scalability based on the communication and reasoning capabilities among different kinds of agents. Our results show that the proposed framework supports feasible, scalable and adaptive responds to evolving contexts.

APPROVAL

The faculty listed below, appointed by the Dean of School of Computing and Engineering, have examined a thesis titled “Situation Aware Mobile Apps Framework presented by Feichen Shen, candidate for the Master of Science degree, and certify that in their opinion it is worthy of acceptance.

Supervisory Committee

Yugyung Lee, Ph.D., Chair
School of Computing and Engineering

Baek-Young Choi, Ph.D.
School of Computing and Engineering

Yongjie Zheng, Ph.D.
School of Computing and Engineering

Jungwoo Ryoo, Ph.D.
College Of Information Sciences and Technology
Pennsylvania State University Altoona

CONTENTS

ILLUSTRATIONS

Figure Page

TABLES

ACKNOWLEDGEMENTS

I would like to take this opportunity to thank the following people who have directly or indirectly helped me in academic achievements. Firstly, I would like to thank Dr. Yugyung Lee, my mentor and advisor, for her continuous support and guidance throughout my master's program in computer science. I sincerely thank Dr. Baek-Young Choi, Dr. Yongjie Zheng and Dr. Jungwoo Ryoo for accepting to be a part of my thesis committee and making time for me off their busy schedule. Finally, I would like to thank my family members and friends for all their encouragement and support.

The views and conclusions contained herein are those of the author's and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the University of Missouri - Kansas City.

CHAPTER 1

INTRODUCTION

1.1 Research Motivation

Mobile phones have been widely used in this modern world. At the beginning of the era of mobile phone development, users mainly use mobile phones as a tool to call others for the purpose of communication. With the increase of functions within a mobile phone, people can now use it for sending message, sharing multimedia contents and using some software for simple purposes. For now, today's mobile phone is not only a phone but also like a small, smart PC that has its own operating system and can run many different types of applications for various purposes.

The prerequisite of our research is the current trend of rapid development of the mobile phone. Today, mobile phones are much smarter than in previous periods. Users are more familiar with using smart phones to deal with most of their daily business and smart phones gradually become an inseparable part of human life. From this perspective, the research on smart phone application has a much greater value than ever before.

Context capture has become an increasingly more important feature in recent smart phone technology. It provides a good way to get the external or internal information for various smart phones. Different sensors are supported in the current smart phones in order to capture surrounding context in the outside environment and smart phone capability context as well. Such context capture methods inspire us a good idea about doing further context modeling and reasoning work to deal with the received context.

A cloud platform based server provides a way of doing request work from the client side in the back-end, that makes the remote communication between different devices possible through web services. It gives us a vision to use a cloud server in the back end to do various intelligent work on the received content.

Many communication methods in current technology also give us several ideas on how to do content sharing between different phones in a certain area. Our research provides a connection method in our local area for mobile phone users.

In general, the motivation of our research is to help people take a rapid response capability to the situation or event in an efficient and effective way. The method we propose is a situation aware framework for adaptive apps, called SAMAF, for evolving situations. We aim to create an adaptation pattern by the combination of an ontology rule based reasoning framework and a service oriented architecture application development and deployment framework.

The simulation in our research is particularly valuable in both disaster handling and clinical trial area. In the future, experiences of Situation Awareness Mobile Application Framework can be used in a real situation domain.

1.2 Problem Statement

In this thesis, a rule based intelligent situation awareness mobile application framework is proposed. The SAMAF framework is able to 1) adaptively generate/assign/invoke proper applications for users based on an identified situation, 2)

adaptively inactivate the unused application that is not related to the situation, and 3) share apps among different phones with synchronization of file contents.

A prototype of the SAMAF application framework has been developed as part of the thesis to illustrate the efficiency and quality of different scenarios.

1.3 Thesis Outline

In Chapter 2 we present the related systems that use the potential dynamical pattern model and also describe other framework about the smart mobile application. Chapter 3 illustrates the model of SAMAF. Chapter 4 brings the detailed implementation steps and methods for SAMAF model. Chapter 5 introduces several scenario use cases and some evaluation testing data for SAMAF model**.** Chapter 6 concludes this thesis and provides information for future work for this application framework.

CHAPTER 2

RELATED WORK

In this chapter, we will introduce some related work about an adaptation model, a reasoning model, and a design pattern model. In section 2.1, we discuss several ontology based context modeling approaches, rule based reasoning, an event condition action approach, semantic based and non-semantic based service adaptation models. In section 2.2, we illustrate some work related to service oriented architecture, mobile phone application development and an agent based approach.

2.1 Adaptation and Reasoning Model

There has been some progress in the adaptation and reasoning model development.

An OWL based formal context model to solve problems of context representation, context reasoning and knowledge sharing is proposed in (Wang, Zhang, Gu, Peng 2004). The feature of the framework is that it uses the reasoning mechanism embedded in OWL to do inference work based on captured context.

A semantic and rule based event-driven Service-Oriented Architecture for an agricultural recommendation system is proposed in (Laliwala, etal., 2006). It is an ontology and rule based semantic framework to discover, select, compose and integrate different web services and to achieve dynamical invocation, composition and execution of web services especially in the agricultural area. The TOPS (Wu and Dube 2001) proposed a declarative modeling language with an Event-Condition-Action model for clinical trial test use.

The SASSY framework (Gomaa, et al., 2010), is used as a model driven framework for the architecture work of different distributed service oriented software systems. SASSY separates operational state machine from the adaptation state machine aims at encapsulating the adaptation state machine in service connector. Instead of using semantic technology and ontology, SASSY uses state machine to implement the intelligent adaptation work.

2.2 Design Pattern Model

Perrey and Lycett (2003) defined the concept of Service Oriented Architecture and made an analysis of the criteria for the use of services. This paper also summarized some advantages of the usage of Service-Oriented Architecture especially in the development of large scale or distributed software systems.

(Lane et al., 2010) gave an overview of various sensors on the phone and the potential usage of them. Also, they combine the recent applications with the sensor paradigms. PlaSMA is a multi-agent-based simulation system introduced in (Warden et al., 2010). An ontology and JADE based simulation model is also proposed in this paper.

Table 1 Feature Comparison Table for Different Event Driven Adaptation Framework

Papers or Framework	**Semantic Based**	**Rule Based**	**Adaptive Model**	**Design Pattern**	**ECA Model**	**Multi Agent**
OWL based formal context model	**yes**	**no**	**no**	**no**	**no**	**no**
A semantic and rule based event-driven	**no**	**yes**	**no**	**no**	**no**	**no**

Service-Oriented Architecture						
ECA Mechanism in Healthcare	**no**	**no**	**no**	**no**	**yes**	**no**
SASSY	**no**	**no**	**yes**	**no**	**no**	
A Survey of Mobile Phone Sensing	**no**	**no**	**no**	**yes**	**no**	
PlaSMA	**no**	**no**	**no**	**no**	**no**	**yes**
SOA	**no**	**no**	**no**	**yes**	**no**	**no**

CHAPTER 3

SAMAF MODEL

3.1 Background

The Situation Aware Mobile Apps Framework (SAMAF) provides an intelligent approach for dynamic activation, generation compilation, installation, uninstallation and inactivation. Meanwhile, the SAMAF model supports continuous and sustainable file communications among phones and their neighbors in a local environment.

We will give a brief introduction of the features of our model.

Context: *"Localized information could be used to characterize the situation of an entity such as location, identities of nearby people and objects, and changes to those objects"*(Schilit & Theimer, 1994; Dey 2001)

Situation Awareness: "*K*nowing what is going on so you can figure out what to do"(Adam, 1993)

Mobile: Mobile is the carrier in our model for receiving situations and doing corresponding actions based on rules. All the demos and scenarios for showing our model are in a mobile phone application format. In the future, we could implement this idea in other platforms like low-end devices and PDA.

Application Framework: In this model, a main control application plays a role as input manager used to receive and deal with the situation. The output of our model is a generated

application in which functions are arranged in a proper order according to the event situation and rule condition. Overall, we use mobile applications to implement our ideas and show the concept in an intuitive method. So the model is more like a framework of the application.

Activation: This component belongs to the dynamic design pattern model. After getting the reasoning result from the reasoning model, this component is responsible for assigning corresponding applications to users. If the applications exist in the cloud server, our framework will just use them based on the reasoning result. If the applications do not exist in cloud server, our framework will dynamically generate them based on the reasoning result. If the applications exist in the users' phones without activation, our framework will invoke the local application based on the reasoning result. Therefore, through this component, the SAMAF framework not only can infer what kind of applications users need, but also do related work based on the reasoning result. The activation component includes the generation state, compilation state, download state, and installation state.

- **Generation:** In this model, a program can be dynamically generated according to the requirements of the users. The requirements include a function requirement and a hardware requirement.

- **Compilation:** In this model, a command can be conveyed to a cloud platform through a web service that compiles the program users require.

- **Download:** In this model, an installation package can be downloaded to mobile phone.

- **Installation:** In this model, an installation package can be installed on mobile phone.

Inactivation: This component is responsible for inactivating the application when situation could not be identified any more. This component, include an uninstalled model and an inactivated model.

- **Uninstallation:** In this model, an expired application can be uninstalled.

- **Deletion:** In this model, the application installation file can be deleted.

Communication: This component provides the function among users that makes sharing files and data synchronization possible.

- **Connection:** In this model, mobile phones can make connections and be ready for transfer.

- **Transfer:** In this model, files will be transferred between different phones.

- **Synchronization:** In this model, the changed content within a file will be synchronized when it is passed to the other users.

3.2 Reasoning Model Overview

This paper focuses on design an adaptation framework based on the scenario of *N smart phones*, *A apps*, *C contexts*, *S states*. We describe the relationship between those participants and summarize them in a predicate logic format.

$$\forall N{:}Phone \bullet \exists C{:}Context \Rightarrow State(S)$$
$$\forall S{:}State \bullet \exists A{:}Apps \bullet (SB(S,A)) \Rightarrow Apps\ (A)$$

To achieve this goal, we used ontology to design the back end semantic domain. We updated the existing ontology with several state classes and context classes. We also built the relationship with classes by using several specified properties. For each class, we added several distinguished individuals under them. Thus, by getting the context individual, we easily found the state individual, and then by applying reasoning rules on state individuals to get the reasoning result, we can easily get an application individual. Figure 1 describes the basic idea of our approach. We will discuss this approach in detail in Chapter 4.

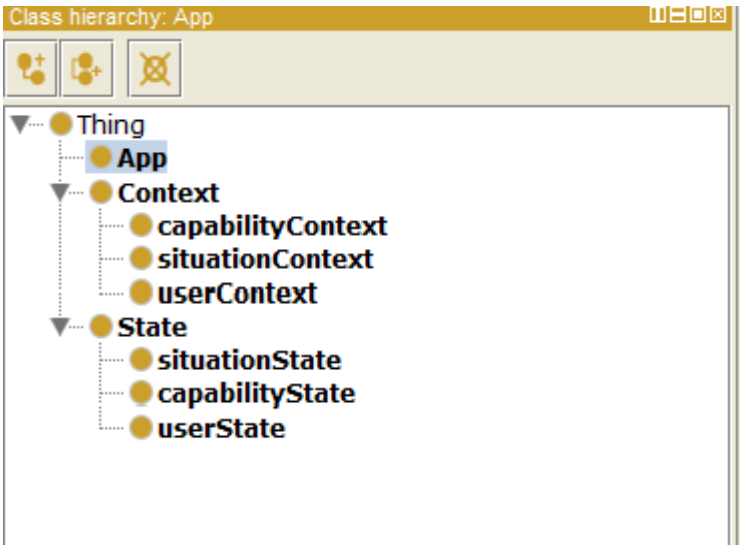

Figure 1 Ontology Design of System

Reasoning work is done by applying SWRL rules on existing elements. Based on different states, different results will be given. After identifying all situations from contexts, the number of rules are needed could be concluded by the combination of all possible situations. Therefore, we used the Cartesian product to finish this work:

$$\{R\} = \{C1\} \times \{C2\} \times \{C3\} \times \{C4\} \ldots$$

Figure 2 shows some general rules we design for our idea.

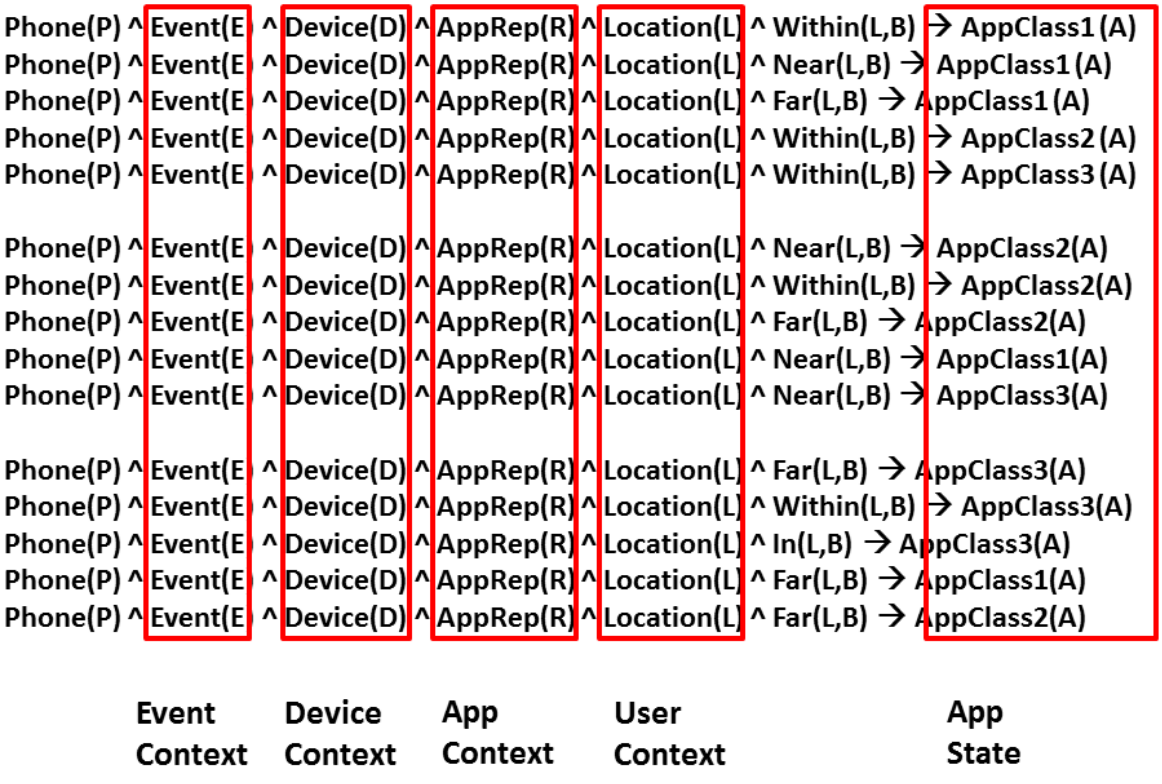

Figure 2 General Rules of Choosing an App

Figure 3 is an example to show the selection of applications chosen after applying rules on the ontology based contexts. With more restrictions in the rule, less applications will be available compared to the total number of applications in our cloud server.

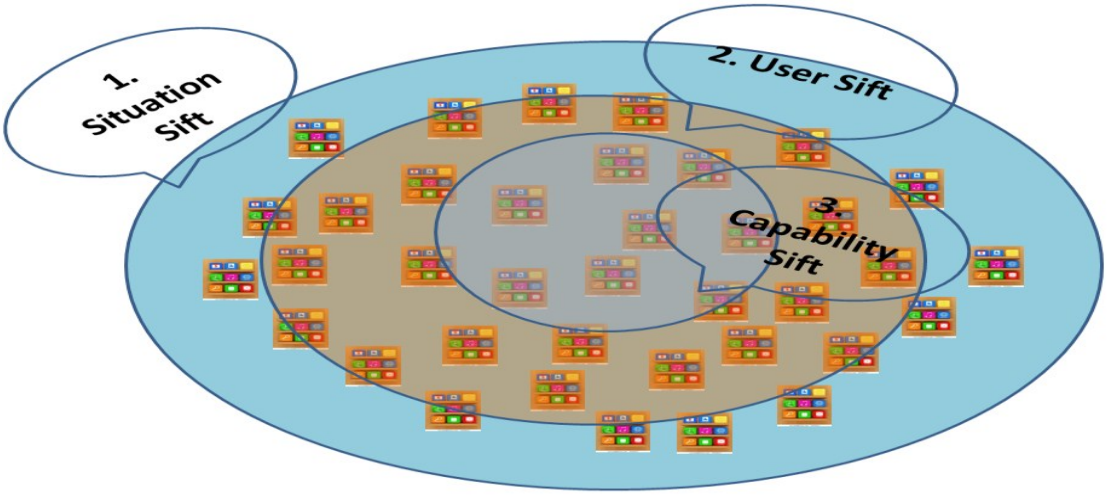

Figure 3 Example of Application Selection

Figure 4 also shows the states and the relationship among them in this work:

Context change → State change,

State change → App change.

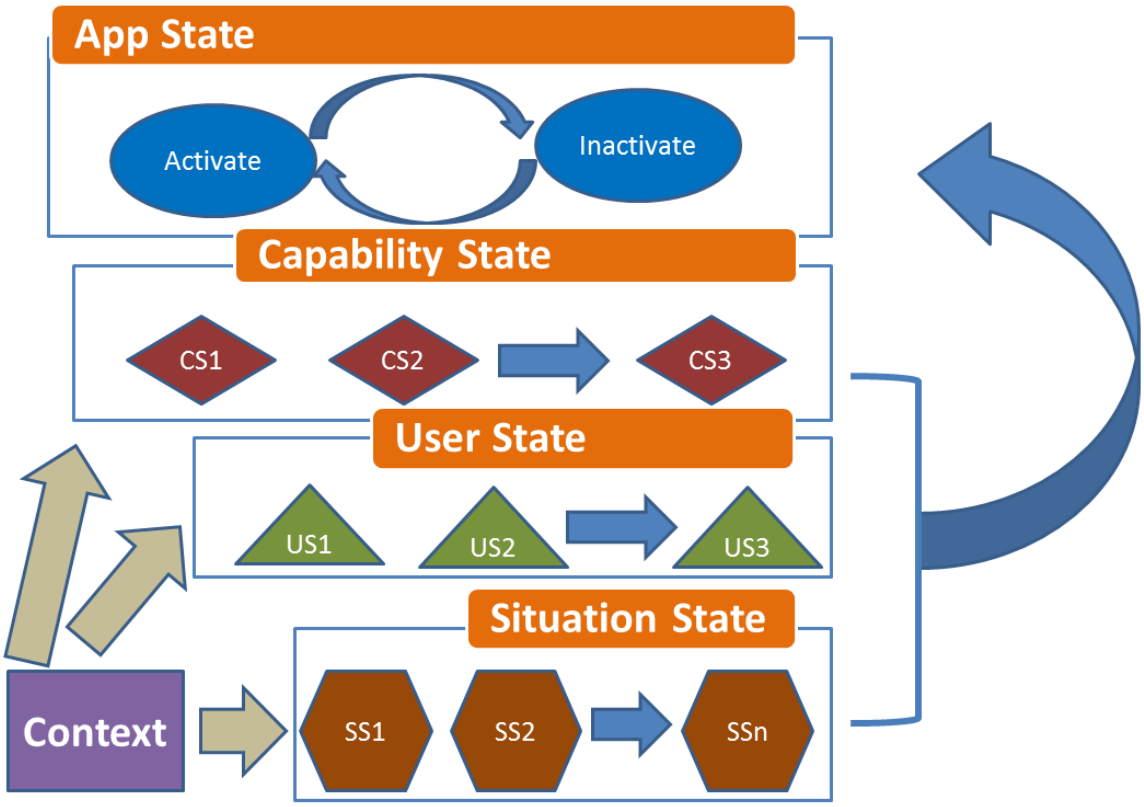

Figure 4 Relationship between State and Context

3.3 Design Pattern Model Overview

3.3.1 Activation

Activation is a module that a mobile phone receives during an event condition and then activates the application that can meet users' needs. Figure 5 shows the state transition among different parts in the activation module. In our case, different event conditions are mapped through the defined ontology in the back end, after a mobile phone gets the information provided by ontology, it will send those event as parameters to the cloud server through a web service. In the web service, intelligent learning will be done by SWRL rules based on ontology. After the web service get learning results, the source code in the cloud server will be modified as the result shows, meaning putting different functions in a proper

order to meet the requirements. After doing that, the compilation component can compile the source code into an install package file. Then that file will be downloaded by download component to the users' mobile phone and the installation work will also be done by installation component right after that. Therefore, the activation module is made up of four parts:

1) Generation of the source code

2) Compilation to the installation package

3) Downloading to the mobile phone

4) Installation on the mobile phone

The greatest feature of the activation module is that every step is dynamically executed. To achieve this goal, design of the ontology and rules in the semantic level plays an important role in our research. Figure 6 is a class diagram to show the relationship among different models in the activation model and how they work together. Figure 7 shows a high level algorithm about the whole process of activation.

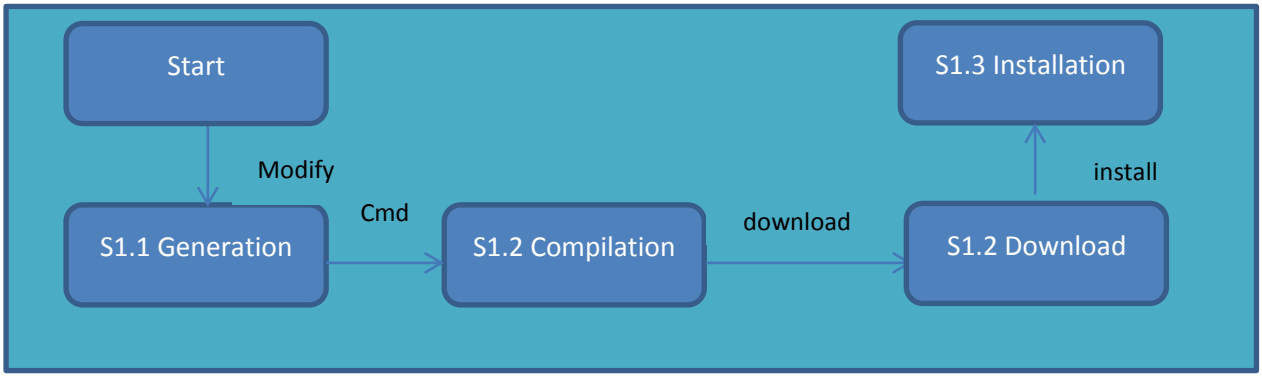

Figure 5 Finite State Machine Diagram of SAMAF Activation

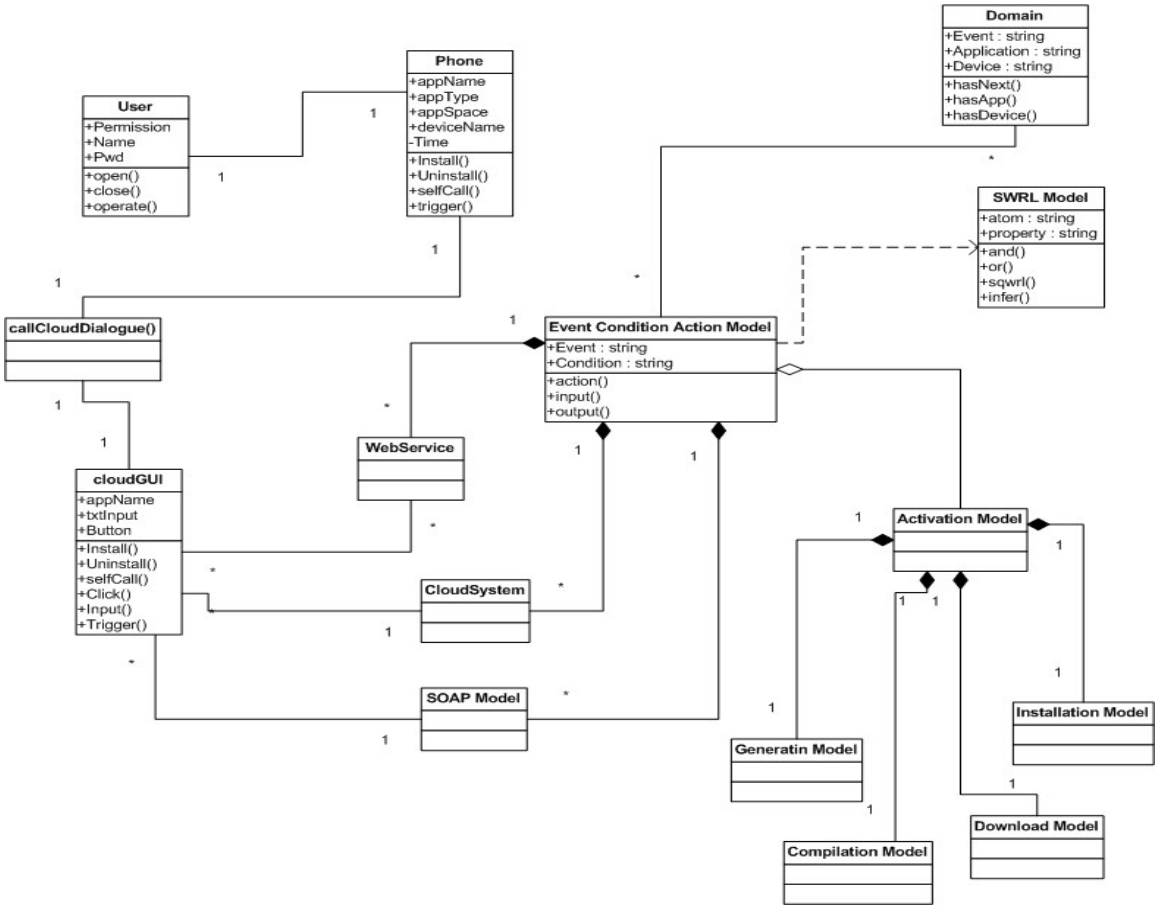

Figure 6 Class Diagram of Activation Model

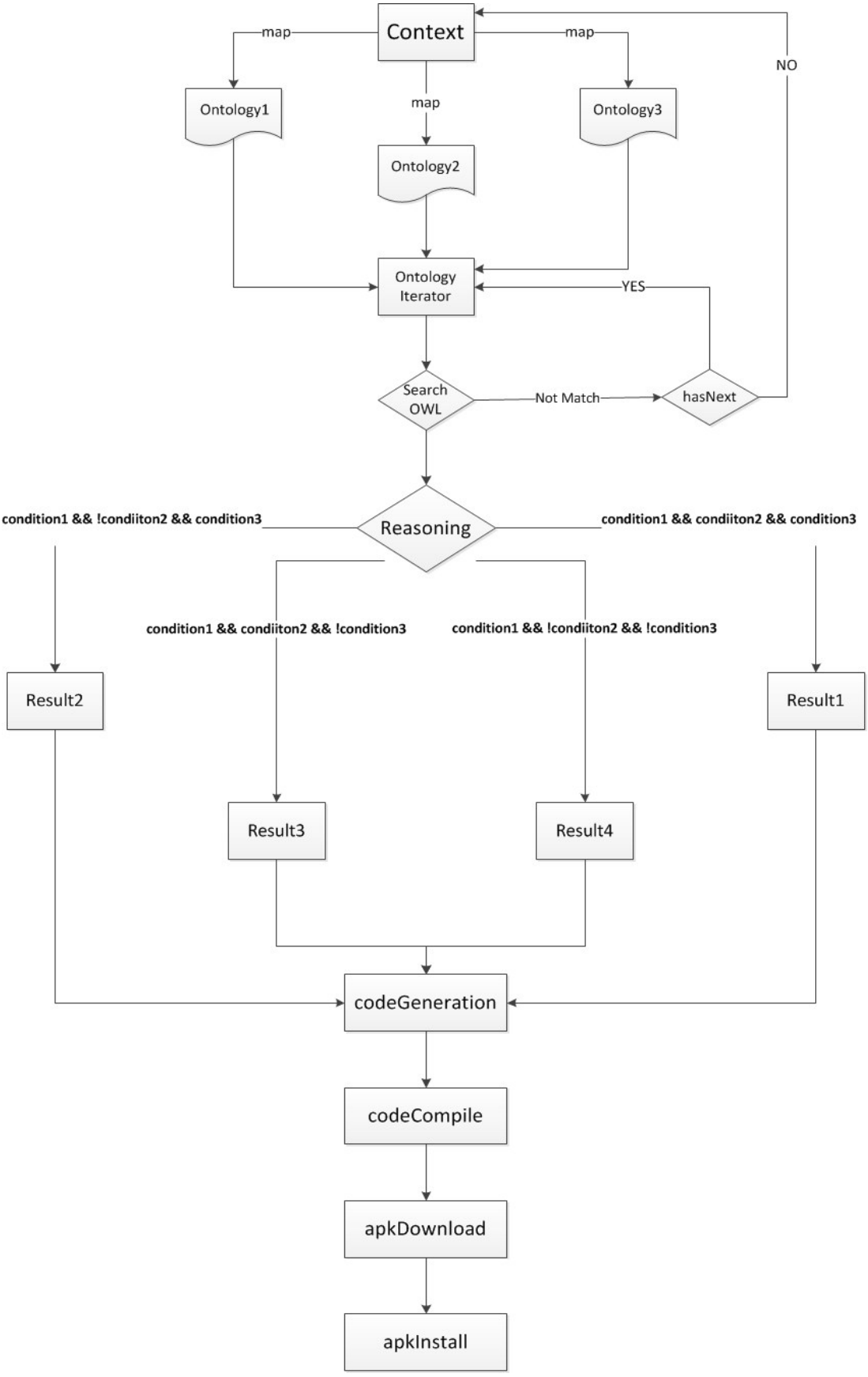

Figure 7 High Level Algorithms in Context Retrieving and Reasoning

3.3.1.1 Generation

The generation model is responsible for the modification and generation of the source code. Four main models are related with this model: the phone model, the event condition action model, the domain model and the SWRL model. The phone model is used to capture the context of the user and environment. The event condition action model is used to receive the information sent by the phone and to do the appropriate action according to the condition result. The domain model is used to predefine the classified domain that could be operated by the event condition action model. The SWRL model is used by the event condition action model to operate on the domain model and infer the result. Figure 8 shows the activity diagram of the generation model. In the diagram, we can see that each decision is made based on ontology and the SWRL rule, which provide an intelligent way to determine the sequence of functions within an application.

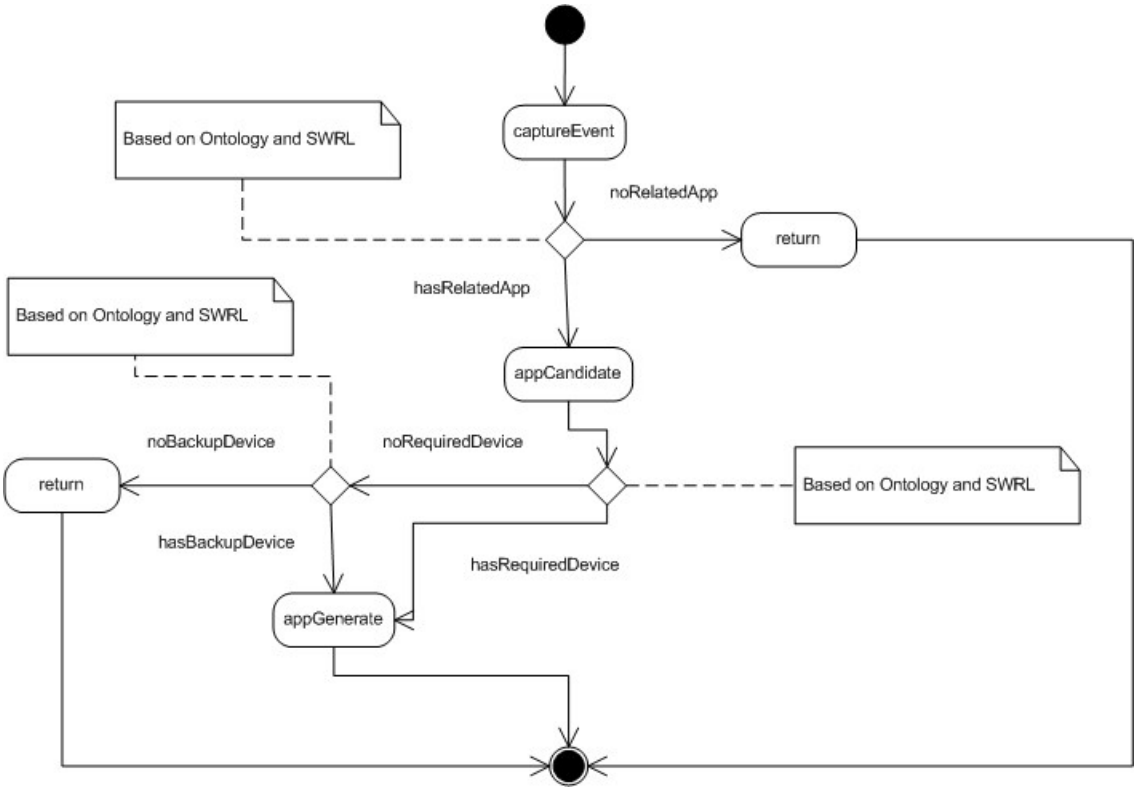

Figure 8 Activity Diagram of Generation Model

Figure 9 is the sequence diagram of how generation model works with other parts and how each step works. As Figure 7 shows, the phone model first gathers context from the outside world then sends such information via the main web service on the cloud server. The main web service then transfer the received information to the intelligent learning web service on cloud server, which will operate on the domain model and the SWRL model to do the inference and get the learning result. According to the result, the generation model will begin to work on the source files on the cloud server and do any relevant modification on them.

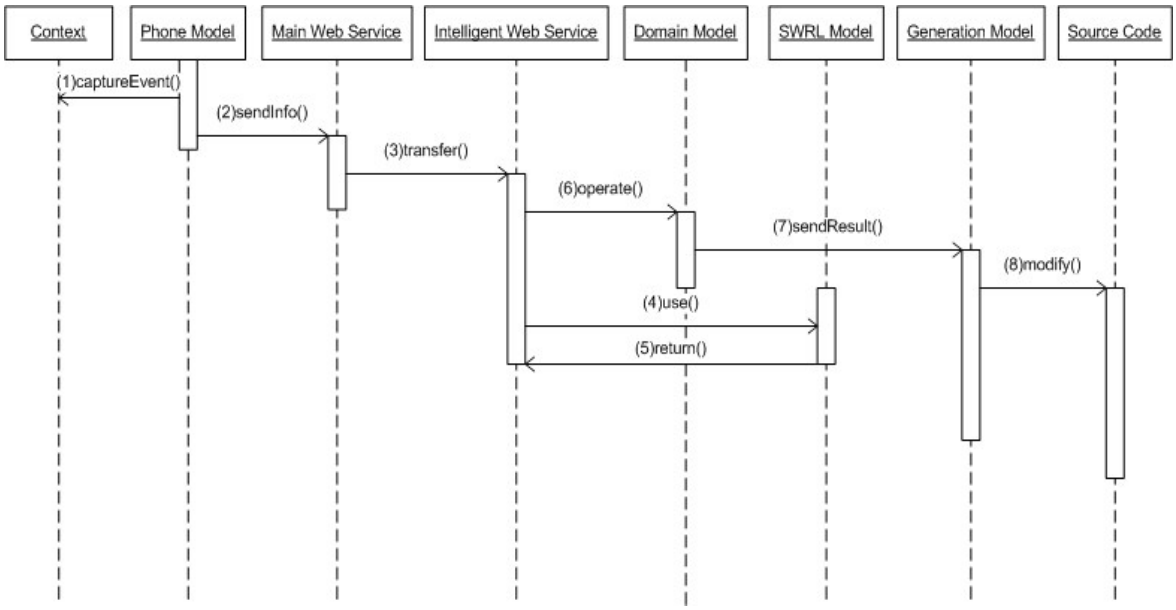

Figure 9 Sequence Diagram of How Generation Model Works With Other Parts

Figure 10 is the workflow for the generation model that describes how functions are chosen and how function sequence is defined. For each application the workflow shows, that we set up an interface or beginning function, that aims at providing an entry for the particular application and prepare for the next functions. After that, there are some bifurcations that can be seen as conditions. According to the result of determination, different functions can be chosen, that means that different actions will be done according the condition. The dash line between the desired function and the end function means that there can be many conditions and desired functions in the middle.

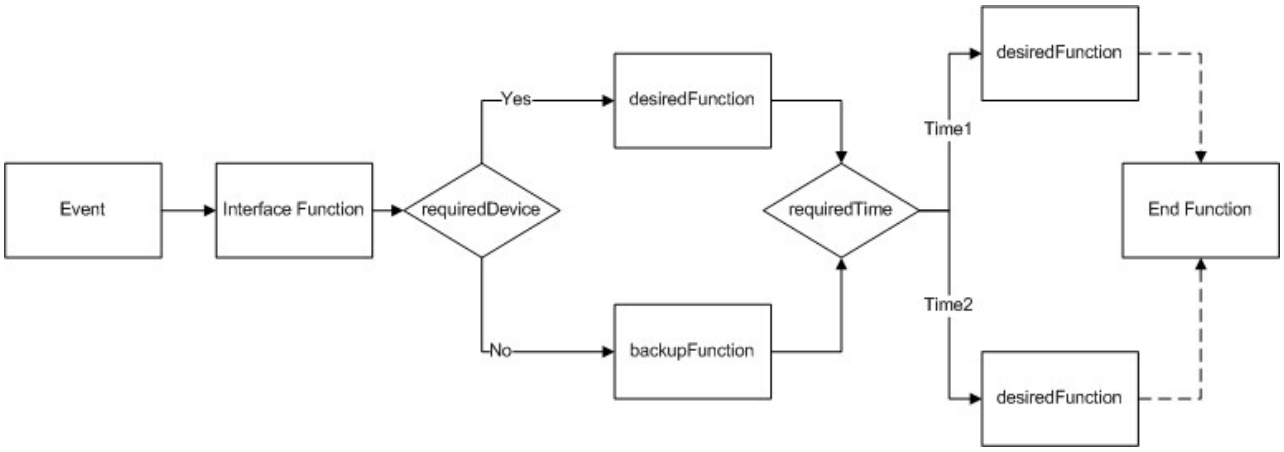

Figure 10 Workflow of Generation Model

3.3.1.2 Compilation

The compilation model is responsible for generating the installation package file based on the existing source code and related configuration files. The compilation model runs on the cloud server and works together with the IDE (Integrated Development Environment) program model, third party mobile phone plug in model and the web service. Figure 11 shows the class diagram of the compilation model. In this model, the web service plays the role of a transmitting command request to the compilation model after making sure the generation model is done with its work. The compilation model performs as the action part in the Event Condition Action (ECA) model. As soon as the compilation model receives the command request, it will call the command model to do the request work. The command model cannot finish the work without the support of the third party plug in model, and the third party plug in model is installed on the IDE model. Therefore, with the help of those three models, the final compilation work can be done.

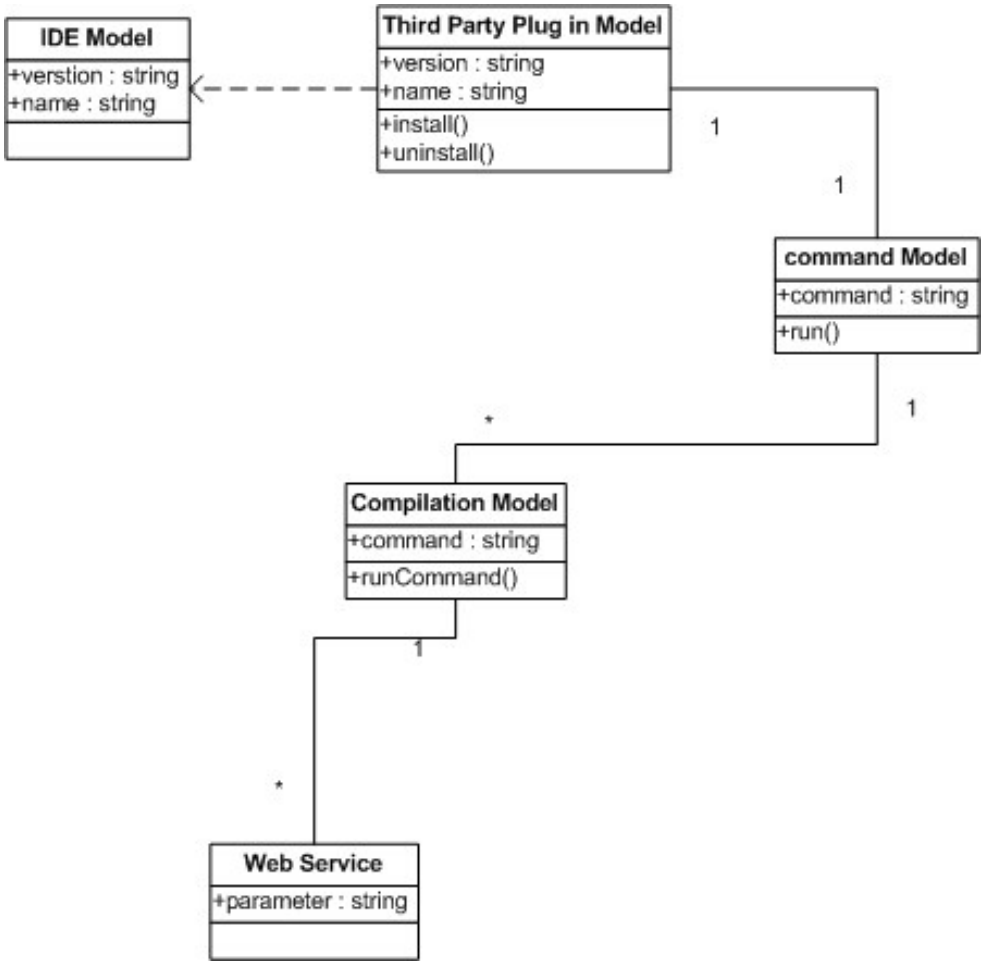

Figure 11 Class Diagram of Compilation Model

Figure 12 shows the sequence diagram of how the compilation model works together with other parts, and describes how dynamic compilation work is done by this model and how the sequence of different models work. As the diagram shows, the compilation model will not be activated by the web service until the generation model finishes its work and sends the finished signal to the web service. After getting the request from the web service, the compilation model will send command statement to the command model, and then the command model will do the compilation work based on the library provided by the third

party plug in model. To make the third party plug in model work, an IDE model must exist to provide the operating environment for the plug in model.

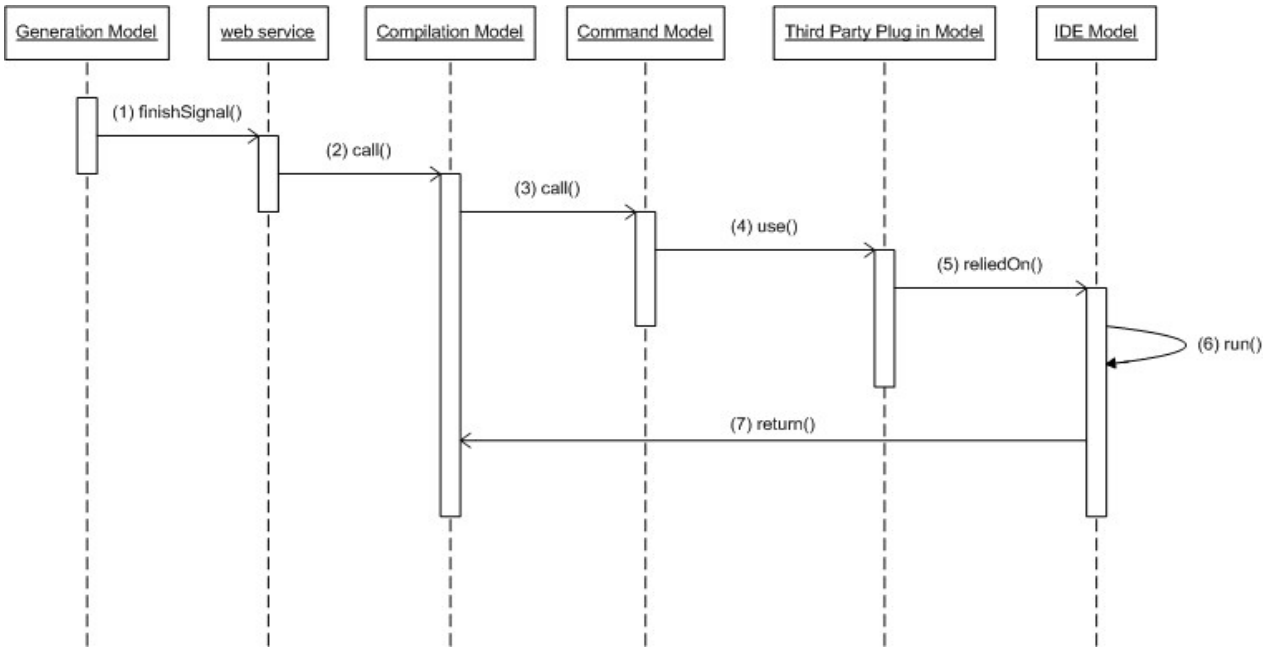

Figure 12 Sequence Diagram of How the Compilation Model Works With Other Parts

Figure 13 shows the workflow of the compilation model. In general, the input of this workflow is command statements, and the output of this workflow is the generation of an installation package.

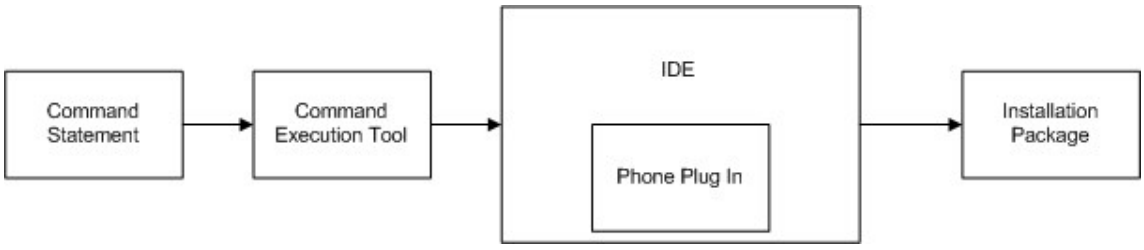

Figure 13 Workflow of Compilation Model

3.3.1.3 Download

The download model is responsible for downloading the installation package from the cloud server to a mobile phone. When the download model is activated, it means that the generation model and the compilation model have already done their work. Figure 14 shows the sequence diagram about how the download model works and cooperates with other models.

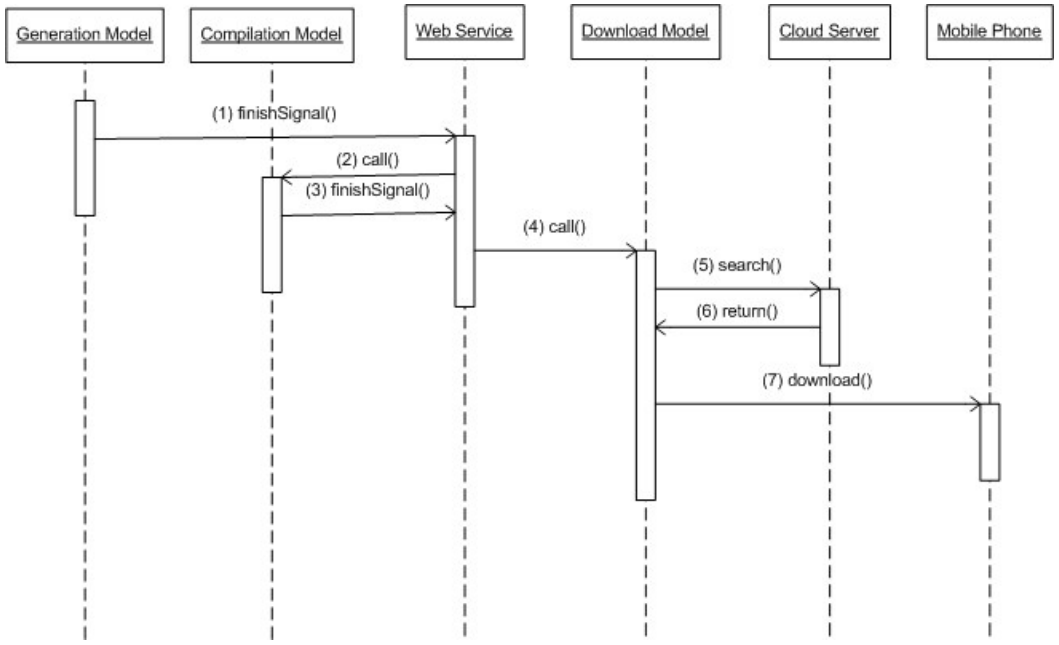

Figure 14 Sequence Diagram of How Download Model Work Together With Other Parts

Figure 15 is the workflow inside the download model. As the figure shows, the mobile phone will first get a URL from the web service. Then it will get a connection with the desired cloud server. The cloud server will provide the request installation package file for the mobile phone part to read. After reading the content into the input stream, the mobile phone will create a temporary file on the disk, and write all the information from the input

stream into the temporary file. After that, the installation package file will have been downloaded successfully from the remote to the local.

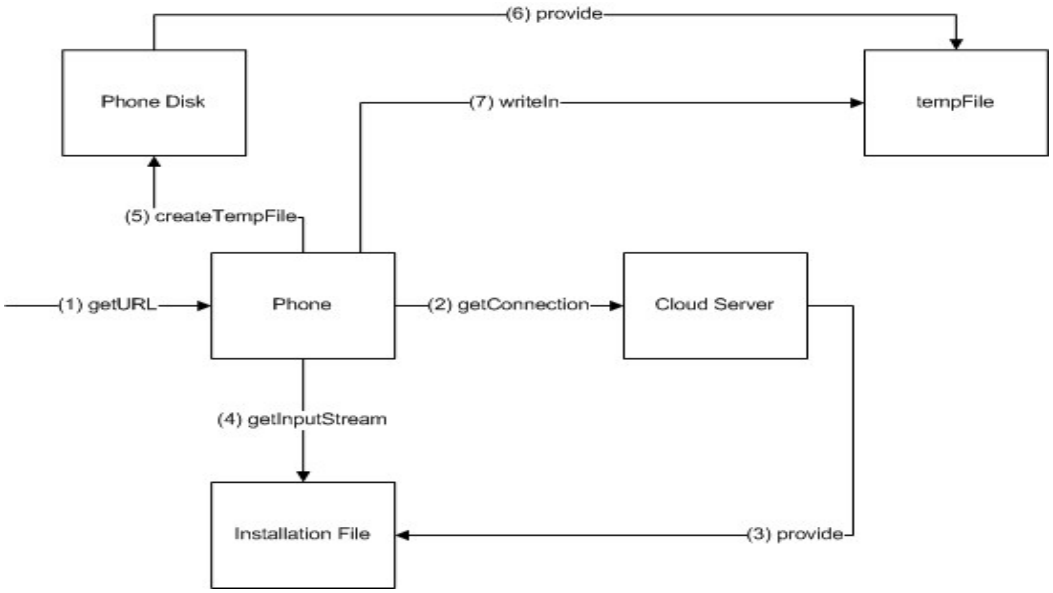

Figure 15 Workflow of Download Model

3.3.1.4 Installation

The installation model is the last model in the activation part. After running this model, the application will automatically run on the mobile phone, and different functions will be activated during a certain time one by one. Figure 16 shows the workflow of this model. The phone can open a new intent to call file activity task, and then get the proper MIME (Multipurpose Internet Mail Extensions) type. In this case, the MIME type is apk, which is an android installation package. After getting data from the temporary file and the MIME type, the installation work is finished.

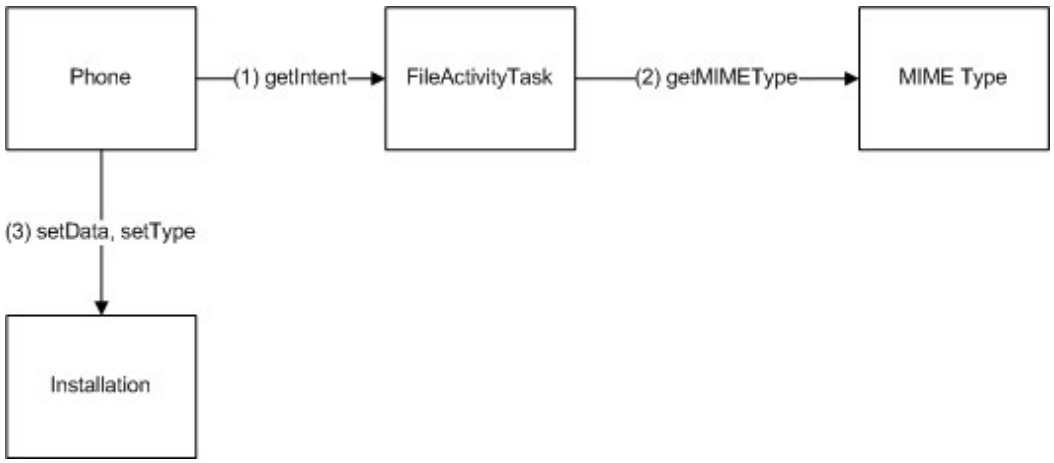

Figure 16 Workflow of Installation Model

3.3.2 Inactivation

The inactivation component is responsible for uninstalling the expired application and deleting the installation package of the application. Figure 17 shows the state transition diagram of this component. When a certain application is not running for a period of time, a message will transmitted to this model. Then this model will send a request to the mobile phone to let it uninstall the unused applications. After the deletion of such applications, the mobile phone will also find the installation package of the particular application in the phone memory card. The reason to do the inactivation is that in each phone, the disk memory is limited, and a mobile phone cannot hold so many contents in a small space. The inactivation model can help the mobile phone use its space in a more effective and efficient way.

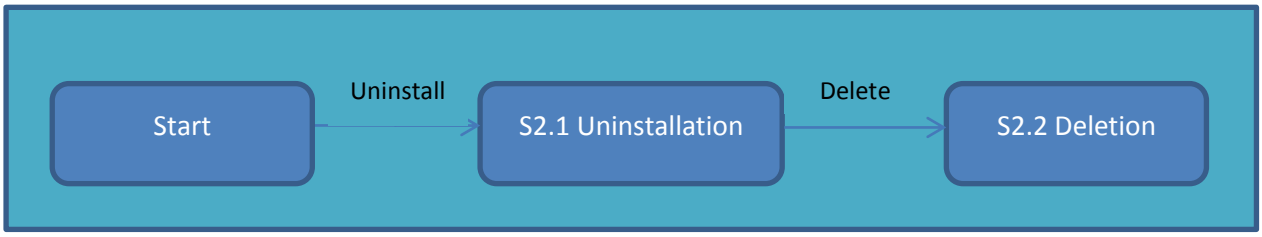

Figure 17 Finite State Machine Diagram of SAMAF Inactivation

Figure 18 shows the class diagram of the inactivation model. In this diagram, we can easily see not only how different components work together, but also the relationship of different parts.

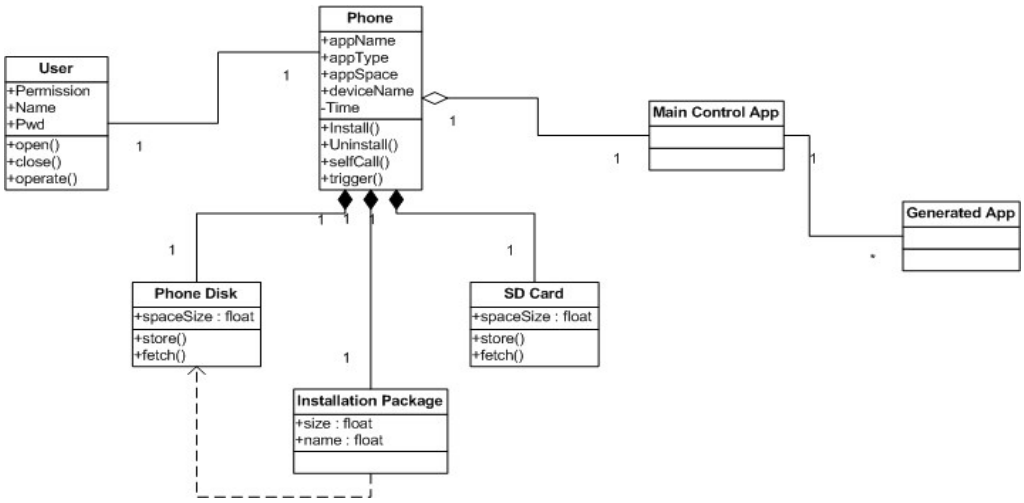

Figure 18 Class Diagram of Inactivation Component

3.3.2.1 Uninstallation

The uninstallation model is responsible for uninstalling expired applications when users do not use them. Our framework first checks whether a particular application is running or not. If the application is not working any more, then after a certain period of time, the main control application will send a uninstall command to the mobile phone to uninstall the application.

Figure 19 shows the activity diagram of the uninstallation model. This model will first check the running status of the application, then determine whether to uninstall it or not. If the application is still running, this model will not do any operation on the mobile phone. Orif the application is not running, then this model will take additional steps to check whether the

time is expired or not. After the time has expired, it is time to uninstall the application from the mobile phone. Otherwise, the model will keep waiting until the time has expired.

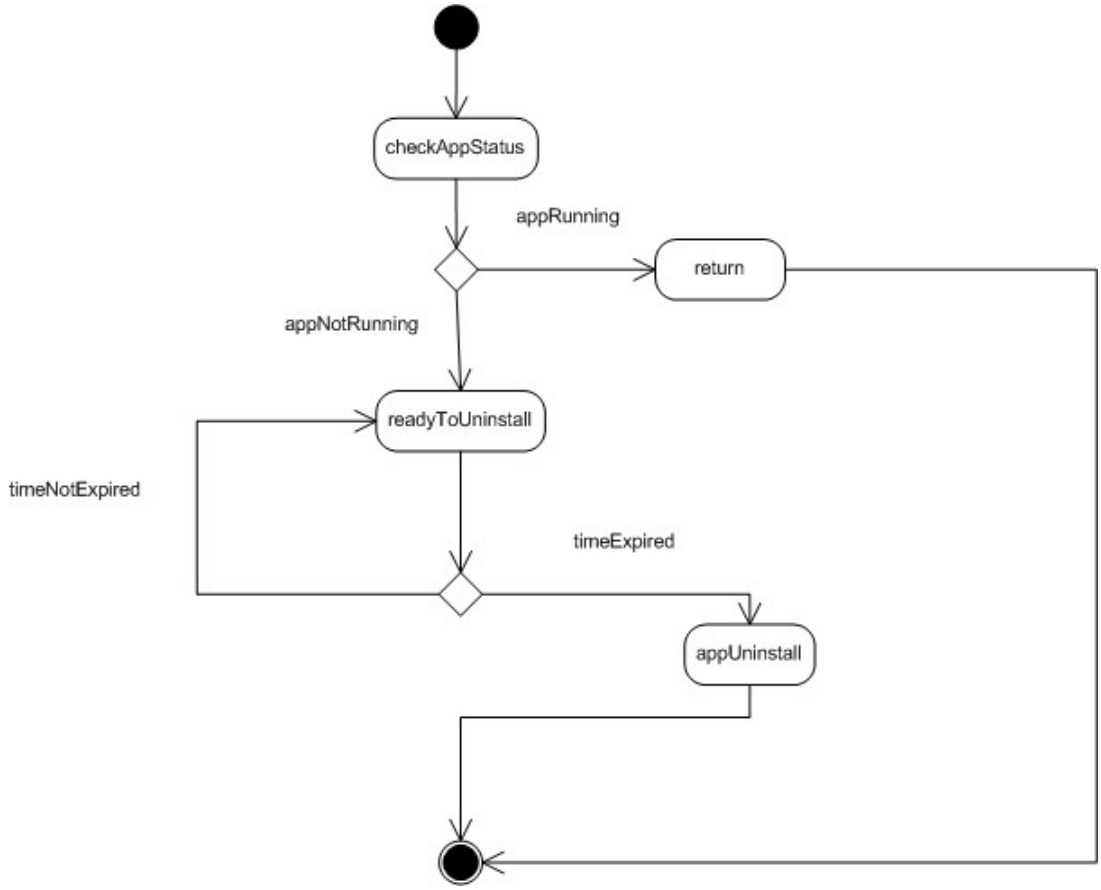

Figure 19 Activity Diagram of Uninstallation Model

3.3.2.2 Deletion

The deletion model is responsible for deleting the installation package on the phone disk or SD card. This model is activated after the operation of the uninstallation model. When the application has been uninstalled, the deletion model will find the related installation package of the application, and remove it from the mobile phone. Figure 20 shows the activity diagram of how the deletion model works. As the figure shows, the deletion model will first

check the existence of the application. If the application does exist, it means it has not been uninstalled, otherwise, it means that this application has been uninstalled. Then the deletion model will check the existence of the installation package file (apk file). If this file does exist then, the deletion model will delete the package, or otherwise do nothing.

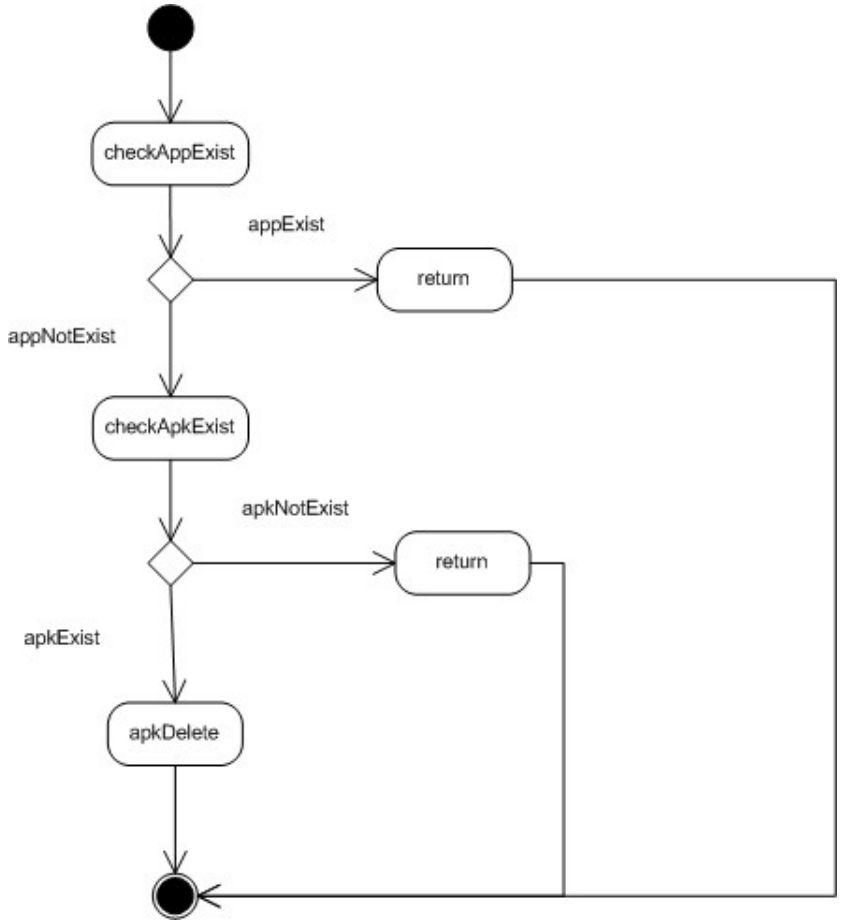

Figure 20 Activity Diagram of the Deletion Model

Figure 21 shows the sequence diagram of how the deletion model works with other parts and how each step works. After the uninstallation model finishes its work, the deletion model will be activated. It will then check the application list about whether a certain application exists or not, and then it checks the phone disk or SD card about whether the apk installation file exists, If it does exist, it will delete the installation file in the end.

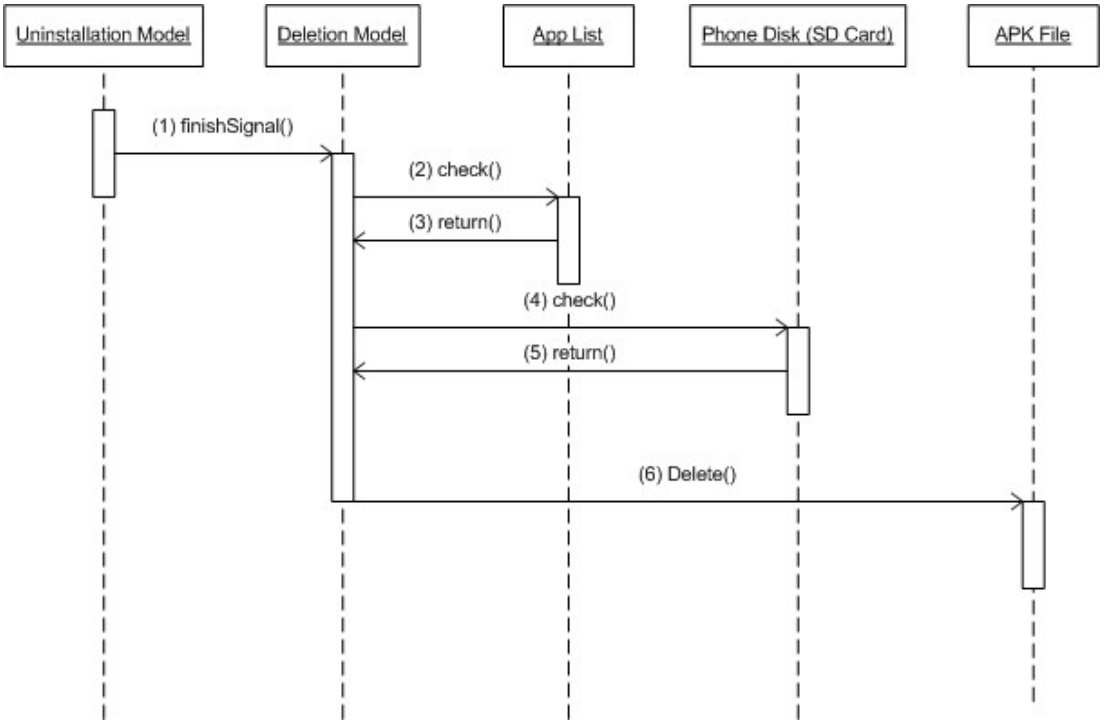

Figure 21 Sequence Diagram of How the Deletion Model Works with Other Parts

3.3.3 Communication

The communication model is responsible for making the connection between different phones in a local environment. Through this connection, users can transfer files between two or more mobile phones. Meanwhile, the users' changed contents can be synchronized to the other side of the connection, which means that mobile phone users can dynamically modify the contents in the real time. Figure 22 shows the level 2 Finite State Machine Diagram of the SAMAF the communication component

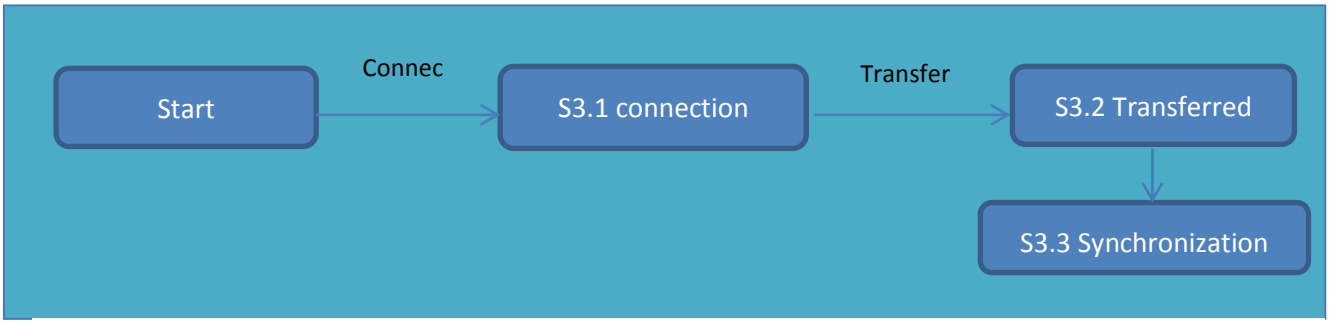

Figure 22 Finite State Machine Diagram of SAMAF Communication

Figure 23 shows the class diagram of the SAMAF Communication component. This diagram just shows a case about the communication between two users. Different phones modify the change and then save the change through a record model. After that, different phones set up the connection through a connection model. After the connection is made, different phones can transfer files through the transfer model.

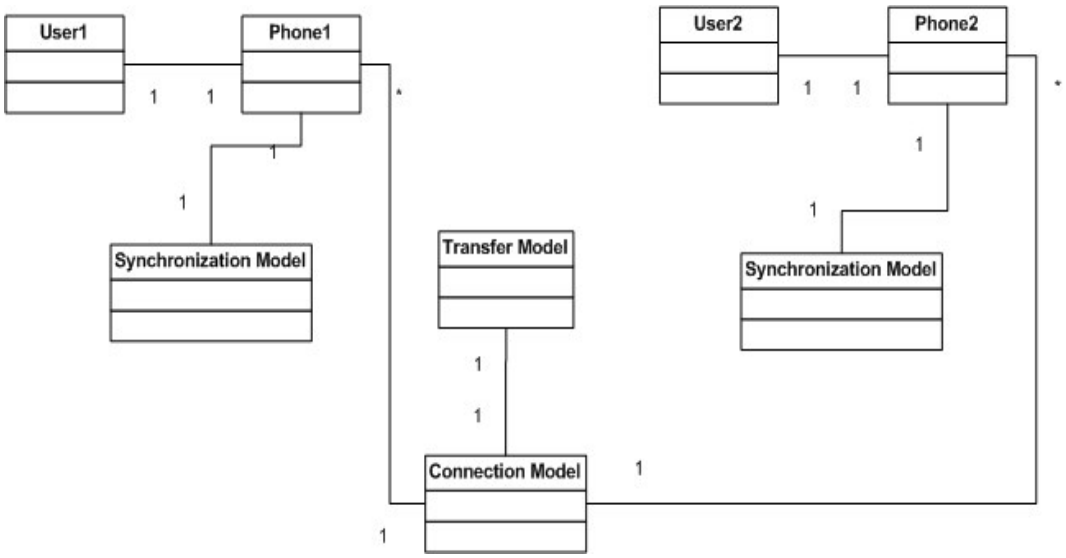

Figure 23 Class Diagram of the Communication Model

3.3.3.1 Connection

The connection model is responsible for setting the bridge between different phones for their future communication. Figure 24 shows the class diagram of the connection model.

A Bluetooth device plays the role of finding other Bluetooth devices. The Broadcast receiver provides the function of listing all available blue tooth devices and their Mac address on the blue tooth list. Therefore, after user clicks on one of the names on the blue tooth list, two mobile devices will try to connect with each other with the help of a Bluetooth socket.

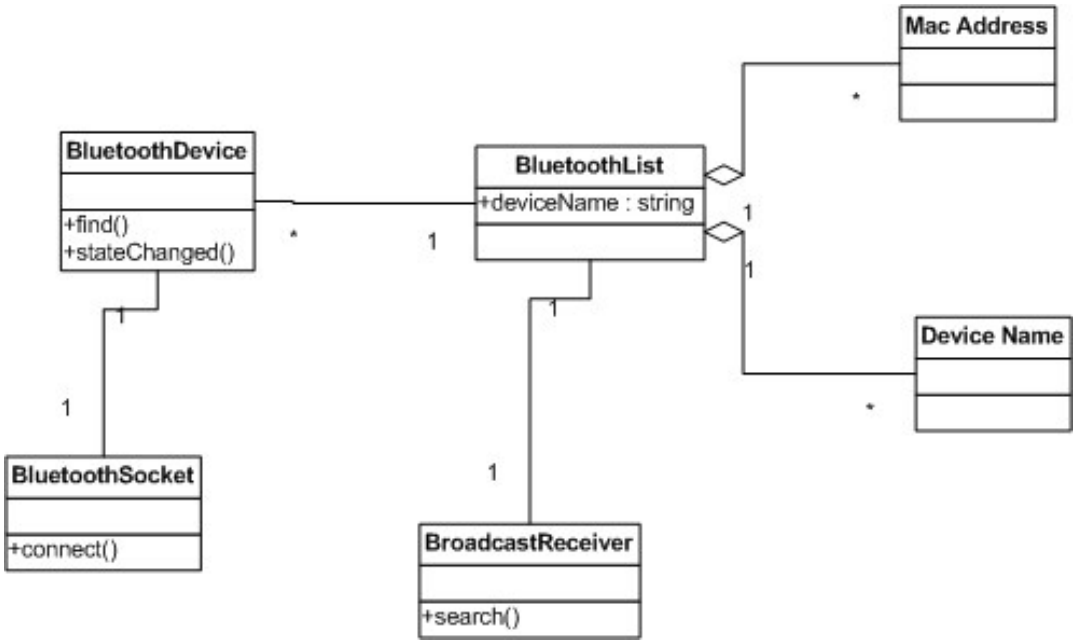

Figure 24 Class Diagram of the Connection Model

Figure 25 shows the sequence diagram of the connection model. The Bluetooth device inside a mobile phone will first open state on and then registers to broadcast to the receiver that aims at receiving the research results of the available Bluetooth devices. Then the broadcast receiver will search by itself and post all available Bluetooth devices' names on a Bluetooth list. When users choose one of the names on the list, the Mac address of the clicked device will be retrieved and returned to the finders' Bluetooth device. After getting

the Mac address, the searcher device will send the address to theBluetooth socket, which provides the function to connect the destination devices by the recognition address.

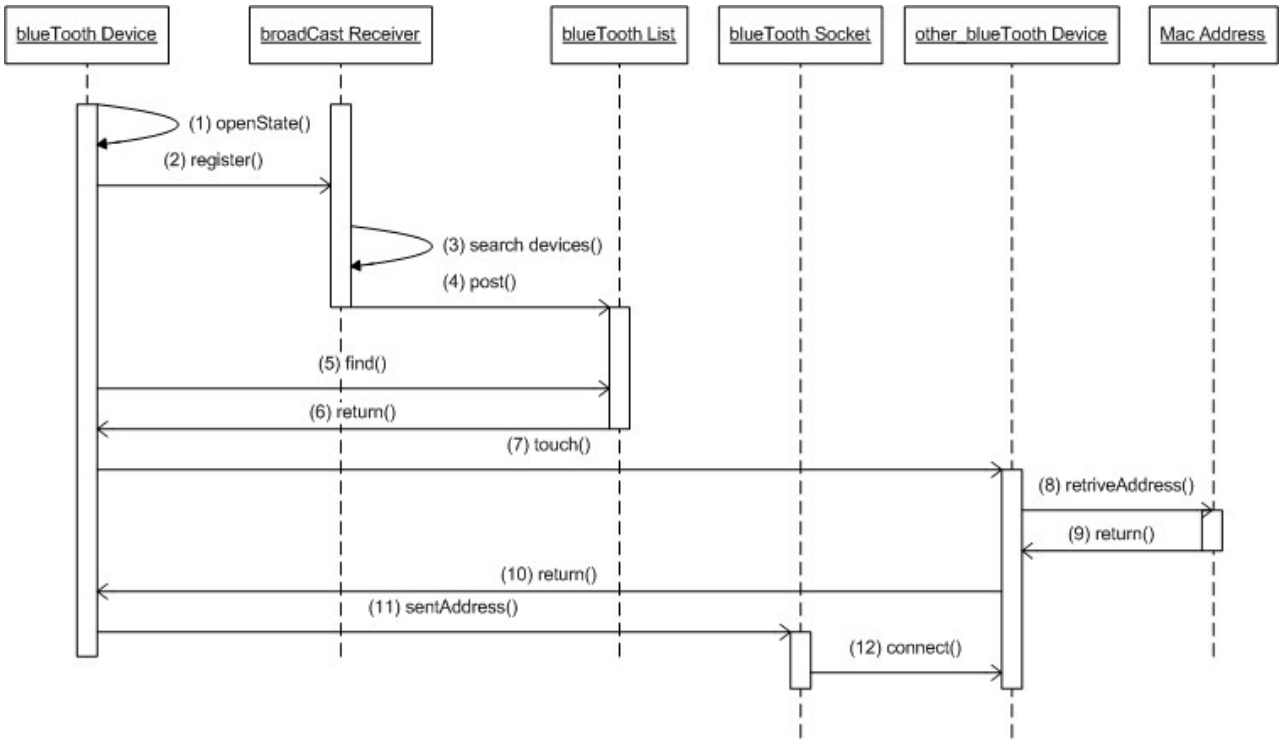

Figure 25 Sequence Diagram of the Connection Model

3.3.3.2 Transfer

The transfer model is responsible for transferring the file between two phone devices through Bluetooth. Figure 26 shows the class diagram of the transfer model. After two devices make the connection with the help of the Bluetooth socket, the sender will give the file path, destination address direction and time stamp as parameters to a content carrier that named the content values. Then all the information in the content values will be inserted to a content resolver to do the transfer work.

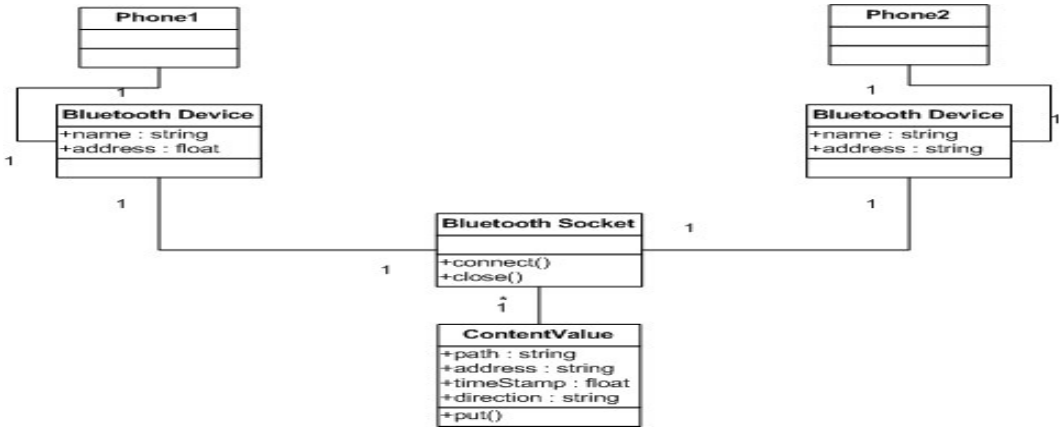

Figure 26 Class Diagram of the Transfer Model

Figure 27 shows the sequence diagram of the transfer model.

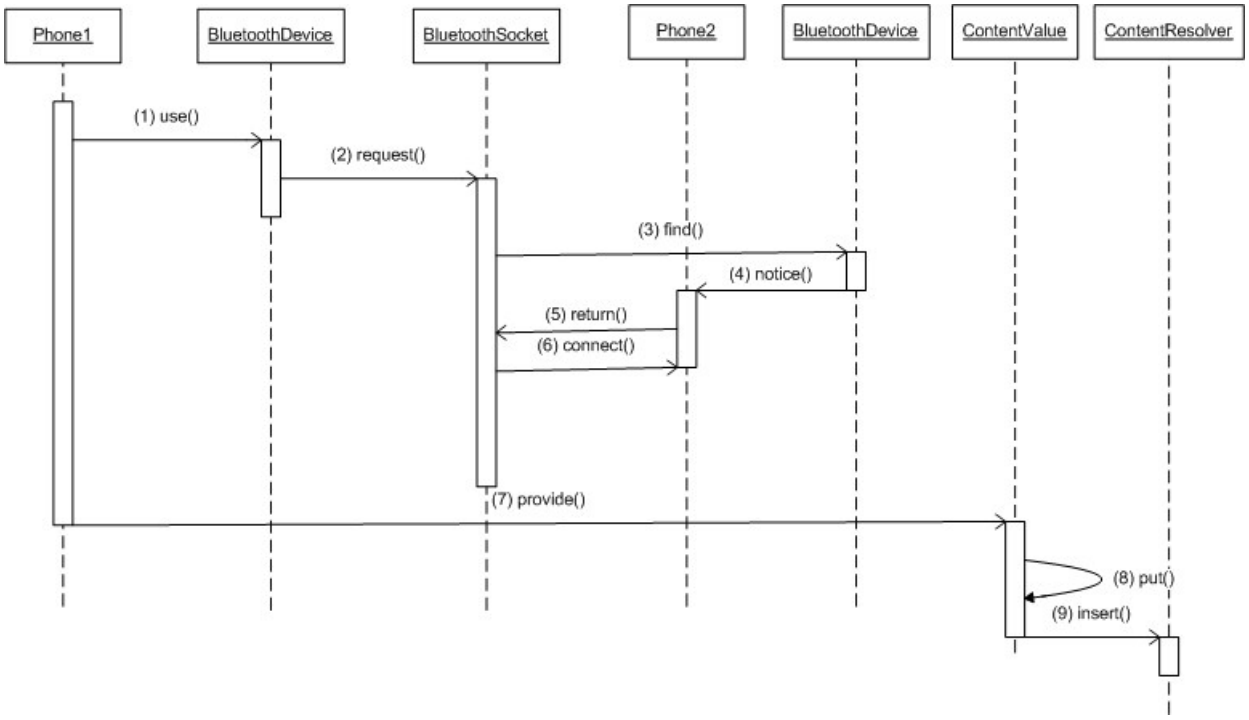

Figure 27 Sequence Diagram of the Transfer Model

3.3.3.3 Synchronization

Synchronization is responsible for synchronizing the transfer file in the receivers' mobile phone after some modifications have been done on the senders' file. To achieve this goal, we plan to use some finely organized format files as configure files to store the changed contents. XML is a pretty good example of such a file. Since we define each tag in the XML file corresponding to the fields in the application file, when we change something in the application, we will store the change in the value part of the related tag. Therefore, when the application file is transferred to the destination user, it will initially load the contents stored in the XML file and write them in the proper place. In this way, the synchronization idea could be achieved.

Figure 28 shows the class diagram of the synchronization model. As the diagram shows, phone 1 and phone 2 can share the information in a consistent way by sharing the configuration file within the application.

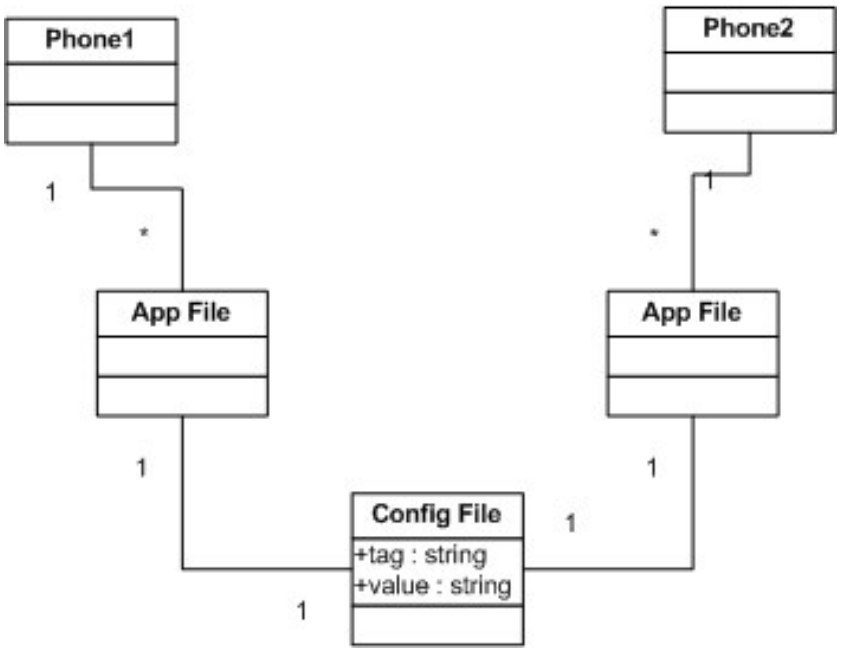

Figure 28 Class Diagram of the Synchronization Model

Figure 29 shows the sequence diagram of Synchronization model

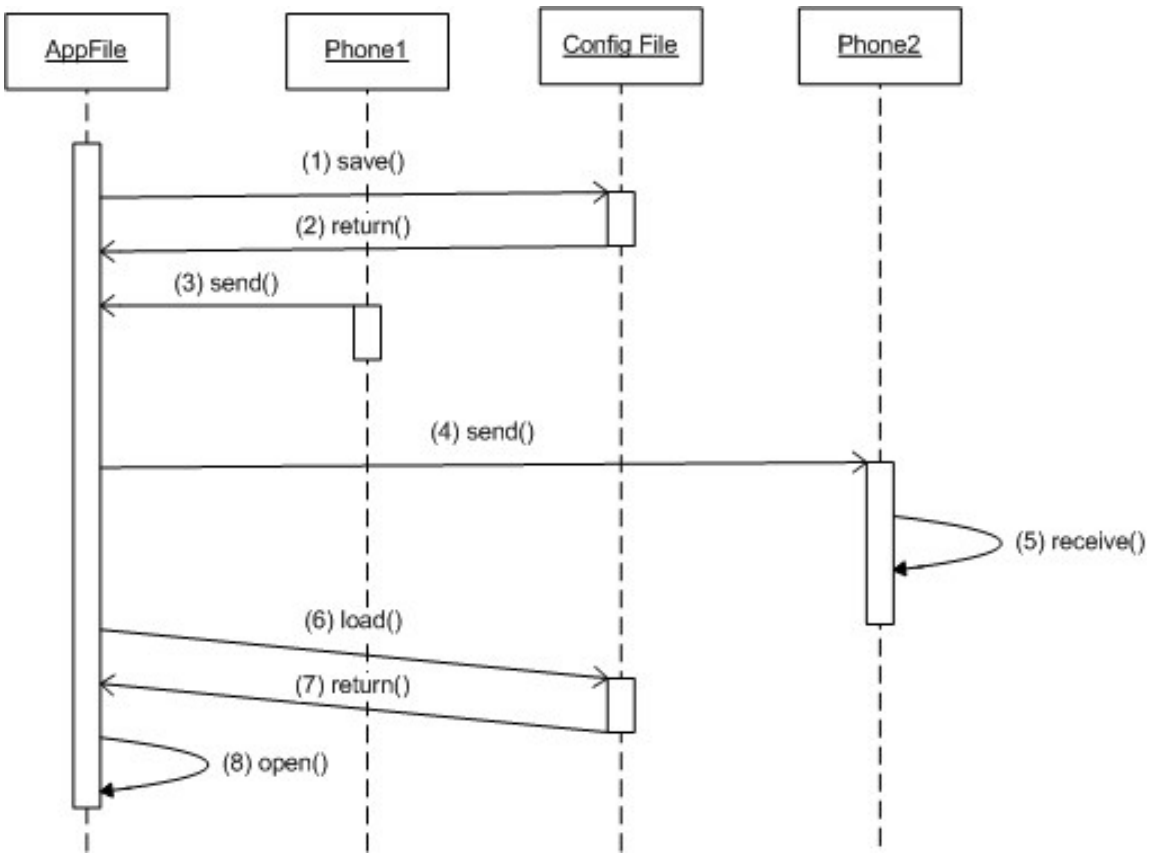

Figure 29 Sequence Diagram of the Synchronization Model

3.3.4 Design Architecture and Pattern Illustration

(1) Behavioral adaptation framework

To achieve the dynamical adaptation feature, a framework that can dynamically change the service by users' changing requirements is necessary. Our framework is a rule based dynamical adaptation framework. A centralized cloud web server will perform the role of monitor to detect the variation of the context in the environment, and then generate the new application based on the situation, and user and capability factors by applied reasoning rules on them. After the adaptation model finishes its work, it will respond to the activation model for the generation, modification, compilation, download and installation work for a specified application. Through the web service, the connection between the client and activated application could be established.

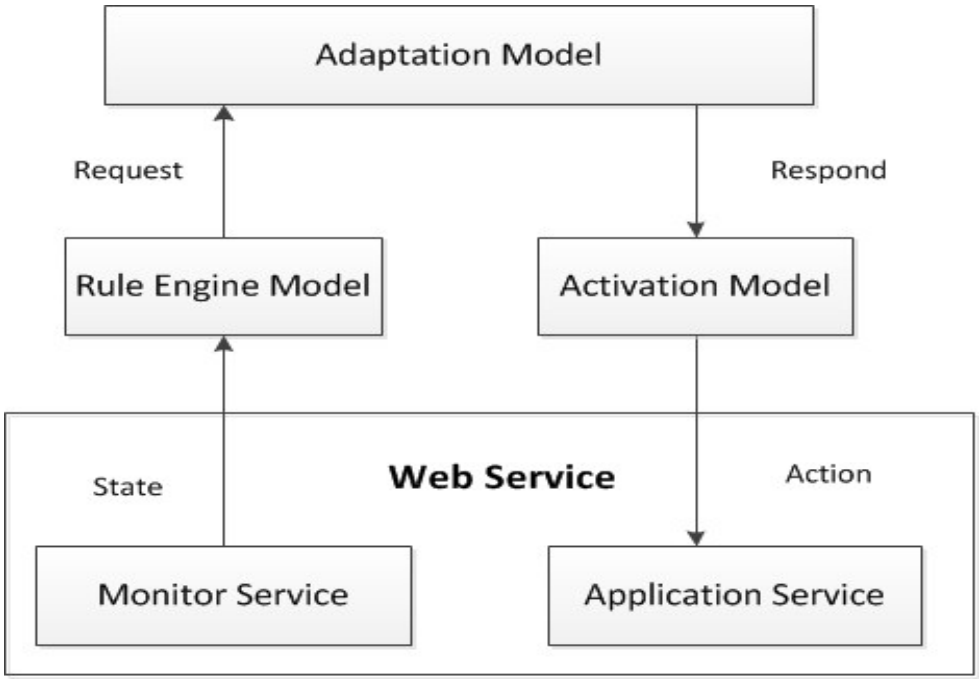

Figure 30 SAMAF Adaptation Framework

(2) Adapter pattern for activation component

In this pattern, the source part is the mobile phone in an inactivated mode without any desired application, the target part is the mobile phone that has installed a specific application and the status is activated. The web service in the cloud platform plays as the adapter to transfer those two above states. Therefore, the functions in adapter class including activate, trigger, generation, compilation, and installation.

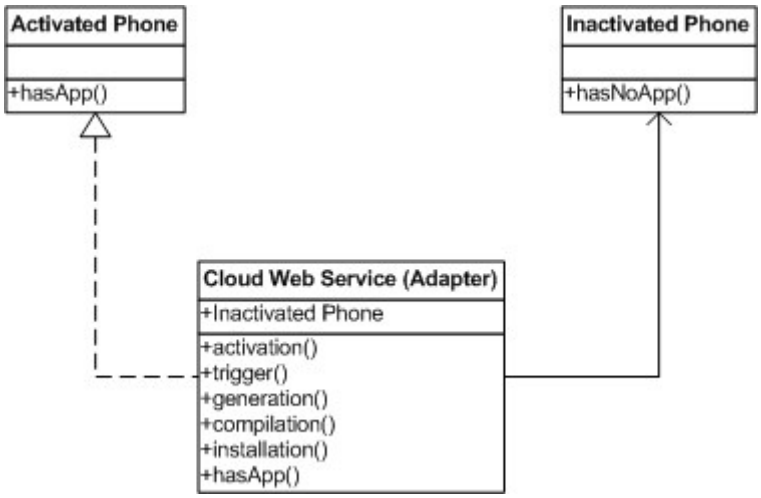

Figure 31 Adapter Pattern 1

(3) Adapter pattern for inactivation component

In this pattern, the source part is the mobile phone that has been activated with the desired application. The target part is a mobile phone in an inactivated state that has uninstalled the desired application. Adapter is a timer class, the functions in such class including alarm and uninstall.

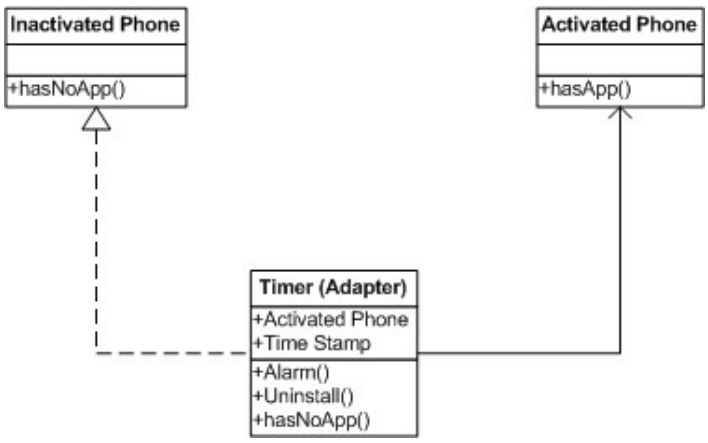

Figure 32 Adapter Pattern 2

(4) Bridge pattern for the relationship between application implementation and application type

The bridge Pattern can separate the abstraction part and the implementation part, so that they can change independently. Abstractions here are some abstract type of applications like music, movie, game and so on. The implementation part will independently do the design work independently for each abstract.

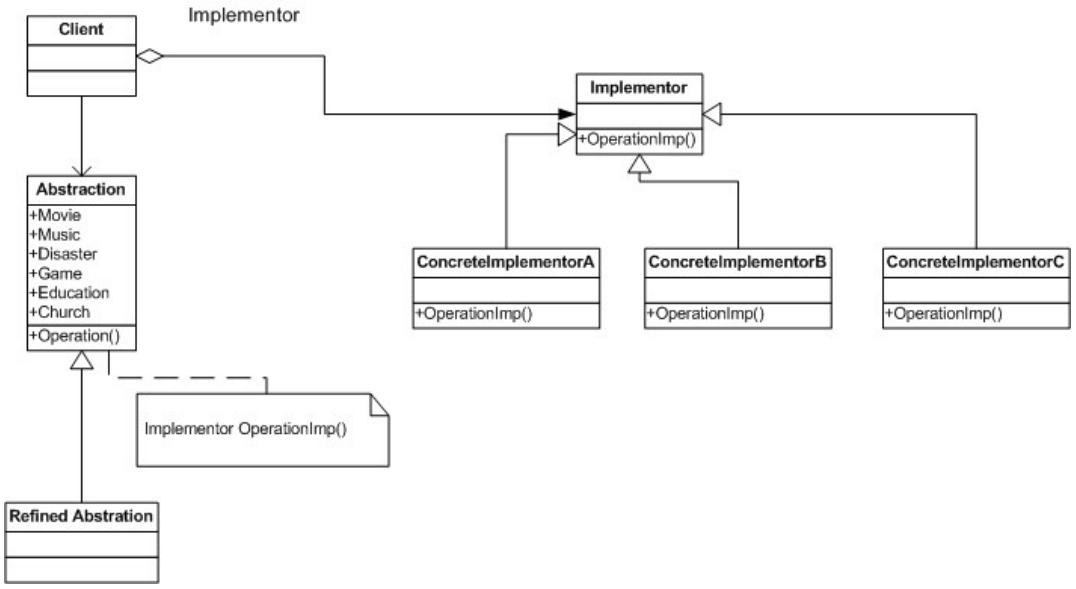

Figure 33 Bridge Pattern

(5) State pattern for the relationship between state and behavior

Each state in the architecture has its own corresponding behavior. With the use of a state pattern, an object can alter its behavior when its internal state changes. An instance exists to maintain a concrete state, and such an instance can define the current state. For each concrete state, each subclass can perform a behavior related to the context.

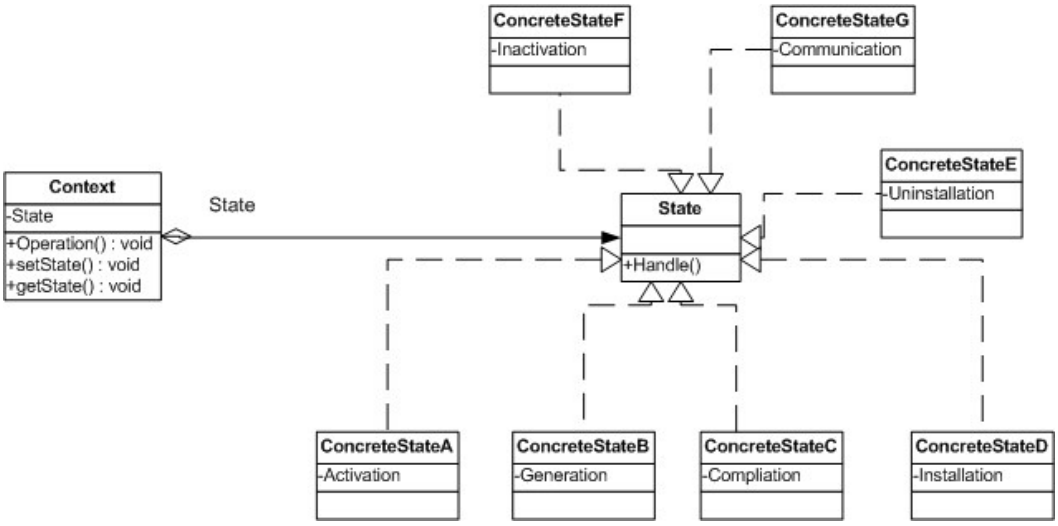

Figure 34 State Pattern

CHAPTER 4

SAMAF IMPLEMENTATION

4.1 Background

The architecture of SAMAF has three parts. Part I is the activation part, and contains the phone, the main control application, the web service, the ontology domain and the SWRL rules. Part II is the inactivation part, and contains the phone, the main control application, the activated application, the phone memory, the installation package and the timer. Part III is the communication part, and contains the phones, the Bluetooth device, the application files and the configuration files.

Figure 35 shows the overall architecture of SAMAF. As the diagram shows, the activation component includes nine units: the phone unit, main control application unit, cloud unit, web service1 unit, web service2 unit, ontology domain unit, SWRL tool unit, installation package file unit and the generated application unit. The inactivation component includes the following six units: 1.the phone unit, 2.main control application unit, 3.generated application unit, 4.phone memory unit, 5.installation package unit and 6.timer unit. The communication component also includes six units: the phone1 unit, the phone2 unit, Bluetooth device unit, application1 unit, application2 unit and configuration file unit.

Each unit in different components plays an important role. In this section we give a brief description about the functionality and importance of each stage.

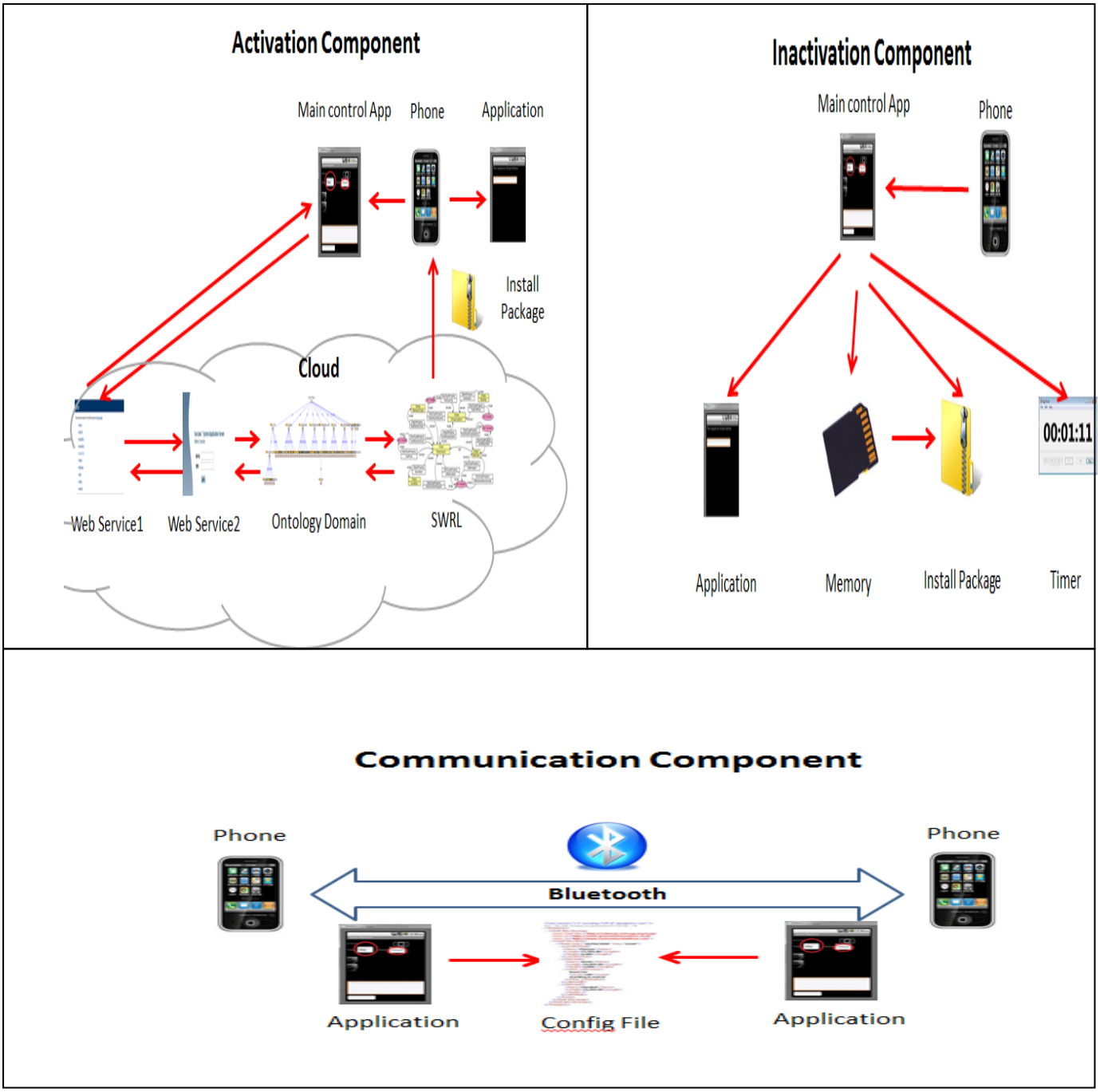

Figure 35 SAMAF Architecture

Main Control Application: This is the main entry for the SAMAF application framework. It is an android application that captures the outside context and communicates with web services in the cloud server. Both the activation and inactivation components need the main control application to serve as an entry for the event and the bridge between phone and context.

Web Service 1: This is the main web service that is responsible for the following: receiving the coming parameters sent by the main control application, sending received information to web service 2 to do further learning, sending generation information, sending compilation orders and sending download requests . Web Service 1 is used in the activation component because this component plays an important role to connect different devices (phone and cloud server) and let them communicate with each other. We use C# to develop web service 1.

Web Service 2: This is the intelligent web service that deals with the context that is sent by web service 1. This web service will operate on ontology by using received context and SWRL rules to infer the proper results. The same as web service1, web service 2 is also used in the activation component to receive the parameter and do reasoning work. We use Java to develop web service 2.

Ontology Domain: This is a set of representational primitives used to model a domain of knowledge. The representational primitives are classes (or sets), attributes (or properties), and relationships (or relations among class members). In our SAMAF framework, we define classes like Event, and Device and Application. We define attributes like hasNext(),

hasApp(), and hasDevice(). Based on each class, we also define some individuals that could be seen as instances of each domain. Because there are relationships between classes, such relationship can be represented among individuals. We use Protégé to develop ontology and use ontology in a code level by operating the owl file. This unit is used in the activation model to do the inference and intelligent learning work.

SWRL Rule: SWRL (Semantic Web Rule Language) is a proposal for a Semantic Web rules-language, combining sublanguages of the OWL Web Ontology Language (OWL DL and Lite) with those of the Rule Markup Language (Unary/Binary Datalog). To achieve the dynamic and intelligent goal in the SAMAF framework, we mainly use SWRL rule operated on ontology to do the inference work. This unit belongs to the activation component.

Application (generated): This is the generated application after getting the reference result from the intelligent web service. Such applications are generated in the cloud server according to the command transferred by web service 1. The generated application unit is a product in the activation component and an object that needs to be uninstalled in the inactivation component.

Apk File (Installation Package): The Apk file is the android installation package that is stored in the phone disk or SD card. The generated application is unpacked from this Apk file.

Timer: This is used to set the alarm for deleting the apk file on the mobile phone. This unit is mainly used in the inactivation component.

Bluetooth device: This is used to make the connection and set a tunnel between two or more phones. Different phones will transfer application files through this way. Therefore, the Bluetooth device unit is used in the communication component.

Configuration file: This is used to record the value for each field. The advantage of using the configuration file in the communication component is that it can trace the change of value from phone to phone. In other words, the synchronization of files between different mobile phone devices can be achieved. In SAMAF, we plan to use shared XML files in the cloud server as a configuration file because of its fine organized format and easy implementation for reading/writing. This unit is a part of the communication component.

4.2 Activation Component

4.2.1 Overview

The activation component is the component for choosing the application with the most appropriate functions for mobile phone users. It provides the generation function for the composition of functions in a proper order. It plays the role of project complier to get the apk installation package. It serves the way for downloading the apk installation package from the cloud server to the mobile phone. It also supports the installation of the apk file on the users' mobile phone.

4.2.2 Composition of Component

As we discussed in the previous section, the activation component is comprised of nine units: There are the phone unit, the main control application unit, cloud unit, web service 1 unit, web service 2 unit, ontology domain unit, SWRL rule unit, installation package unit and generated application unit. In this section, we will give a detailed description of how each unit is composited and what is the ingredient in each unit.

4.2.2.1 Phone

In this research, we use Android smart mobile phones as experiment equipment. The version we use is Android 2.2. We use both mobile phone simulation installed on an Eclipse environment and a real mobile phone to do the implementation work. The reason we use phones as tools to do the implementation is that a phone is the pocket electrical equipment for users, and users are most familiar with the use of phones, so mobile phones provide a user friendly advantage when compared to other equipment. Moreover, because users will carry their mobile phones every day, it is very easy and convenient to get the environment context around or near them.

4.2.2.2 Main Control Application

The main control application is the main entry of the whole activation component. It is responsible for getting various types of context, reactions to the context by calling the web service to get any response operations from the cloud side, setting up the connection between the mobile phone device and the remote device. To achieve dynamic, the existence of the

main control application is pretty important. It can be seen as an application behind the scenes. The main control application receives the context and according to the ontology and inference rules, deciding what to do is the further step. An important thing need to notice is that because of a lack of some sensor or transducer to let our mobile phone got the information, we use some string variables as parameters to present the situation that the sensor may transfer. Therefore, the main control application must be opened and the string variables must be predefined to guarantee that our research can be conducted. In our research, we assume any dynamic operation is based on the prerequisite that the main control application starts.

4.2.2.2.1 File Structure Introduction

In this research, we use Eclipse as the development environment and Java as the programming language to implement our ideas. The project file includes the following: the source folder, generation folder, library, resources folder, manifest file, and property file. The source folder contains the activity files. The generation folder contains the R.java file that lists all the widgets in our project. The library in this project includes Android 2.2 and the soap library. The resources folder contains the layout design file that designs the interface of each activity. Figure 36 shows the file structure of our project. In general, the most important folder related to the design is the source and resource folder. The Java files in the source folder are used to design the function; xml files in source folder are used to design the user interface layout.

Figure 36 File Structure of Android Project

4.2.2.2.2 Function Introduction

Functions in the main control application are responsible for receiving information from the user and presenting the result to the user to achieve the dynamic communication feature between the user and remote server. In this section, we will give a detailed introduction of each function and how we make those functions to work together to achieve the dynamic and automatic features.

(1) Context Retrieval Function

To get context in our research means that a main control application could have the ability to retrieve or capture environment context from a mobile phone device or out of the mobile phone itself, and then update the existing Ontology with those customized context information. Context inside the mobile phone includes time, direction and the capability of the phone. Context outside the mobile phone includes location and disaster type. All such

contexts should be retrieved by the main control application. Figure 37 shows an example of context in a disaster scenario. As this figure shows, the time and event is the context or event to be captured. In this situation, the time is “day” and the event is “earthquake”. After getting the input, the main control application will update the ontology and perform as a communication bridge to send those variables to the remote cloud server to do further intelligent learning work. Next, we will describe how the main control application transfers these parameters to remote destinations.

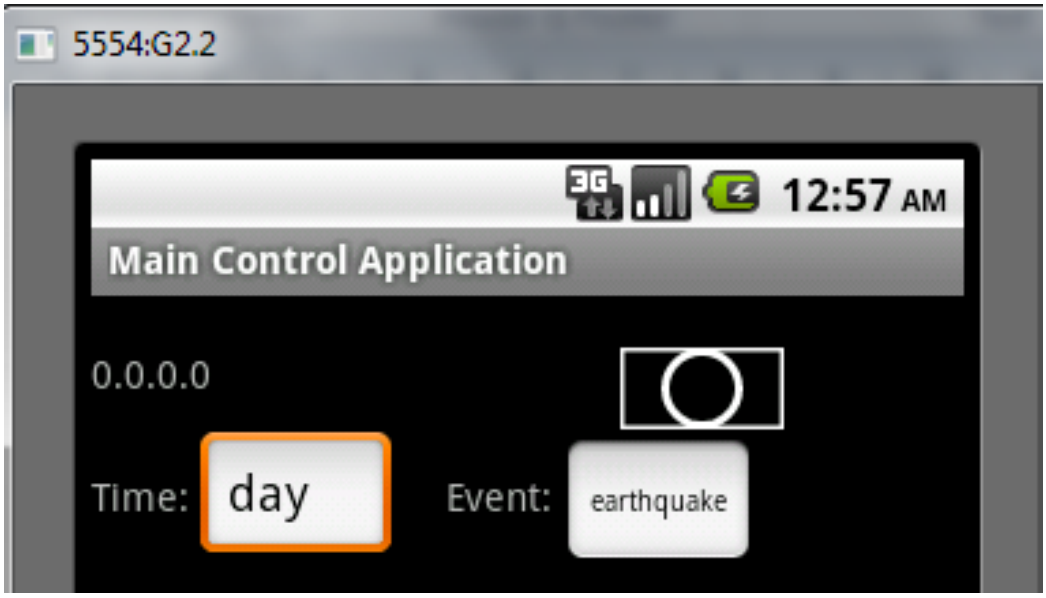

Figure 37 Context Capture Simulation for Main Control Application

(2) Transfer Context and Return Value to Remote Server

After getting the context contents as parameters, the main control application will try to send those parameters to the cloud server to do further operations. We use the SOAP protocol as the communication protocol between phone devices and remote devices in our research. The SOAP protocol is known as the Simple Object Access Protocol. It is a protocol for

communicating formal information in the implementation of the web service. In our main control application, we add a soap library to achieve the communication goal. The library is named the “ksoap2-android-assembly-2.5.2-jar-with-dependecies.jar”. After using it, we import the following: the Soap envelop, Soap object, Soap serialization envelope, HTTP transport SE and XML pull parser exception in our program that aims to using the SOAP object to do the transfer work. Figure 38 shows how SOAP in the application work transfers to the parameters.

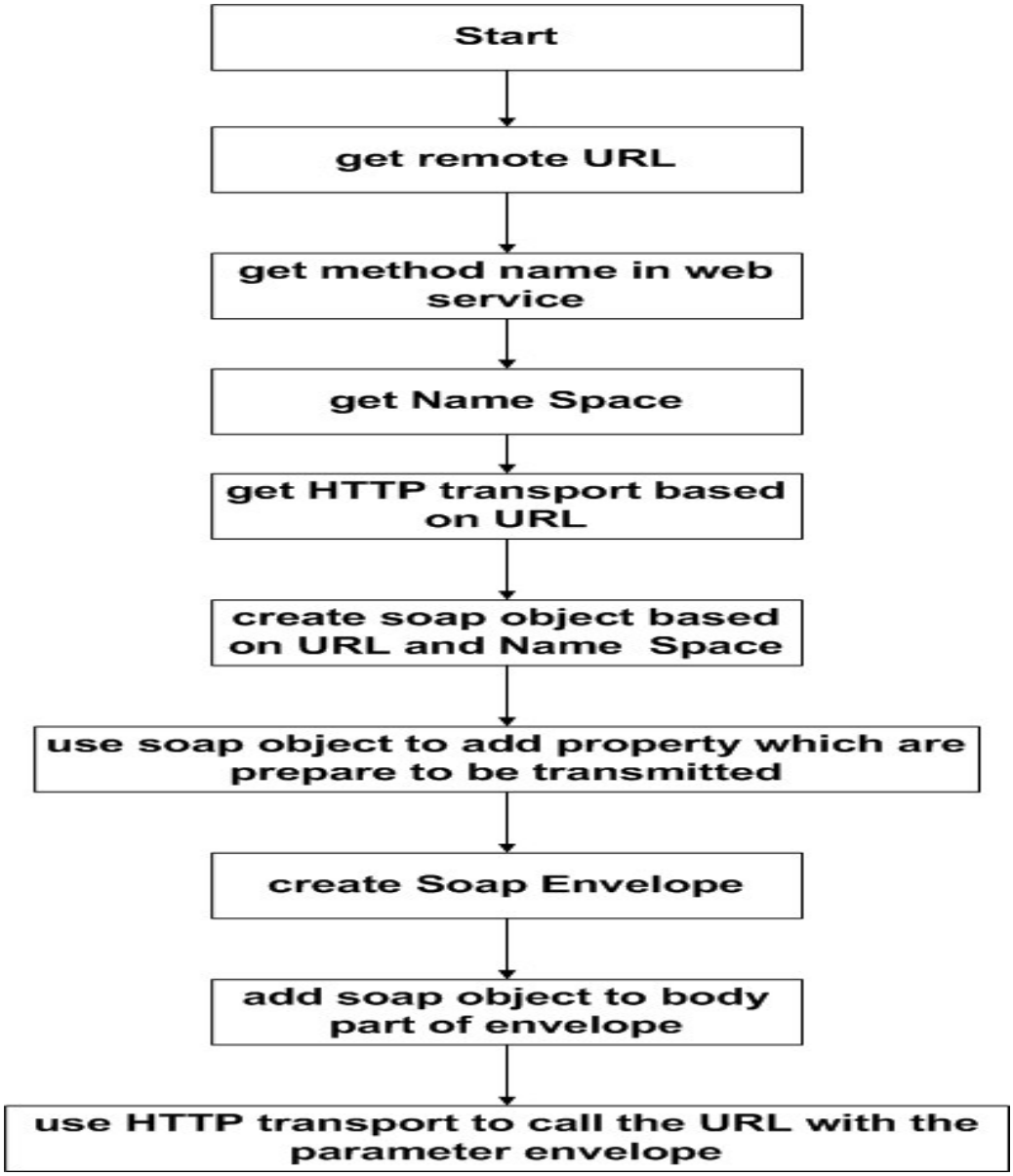

Figure 38 The Sequence Process for Building Connection Using SOAP

The detailed steps of how to establish SOAP connections is showed in Figure 39. The URL mentions the address of the web service on the cloud server. The method name points to the calling method name defined in the web service. The name space is specified as http://tempuri.org/, that is the test's default namespace URI used by Microsoft development products. After getting all the information, we use HTTP to make the connection based on the URL, and create a SOAP object based on the URL and namespace. After the SOAP

object is defined, we create a SOAP envelope and add parameters we want to transfer into the SOAP object and insert the SOAP object in the body part of the envelope. In the end, we use the transport to call the URL with the envelope that carries the transfer information. The main control application will get the return response from the web service side. Such a return value is the intelligent inference result. We will discuss about how web services, ontology and SWRL rules work together to get the inference result in later sections. Inference results should be some function names related to the context, after getting such application name string. Figure 34 shows the inference return value gotten from the remote. The return name string is “earthquaketip, map, call, end”. Each one represents a function within a single application. The main control application is responsible for transferring the return value to the web service to do further analysis. The transfer method is the same as above.

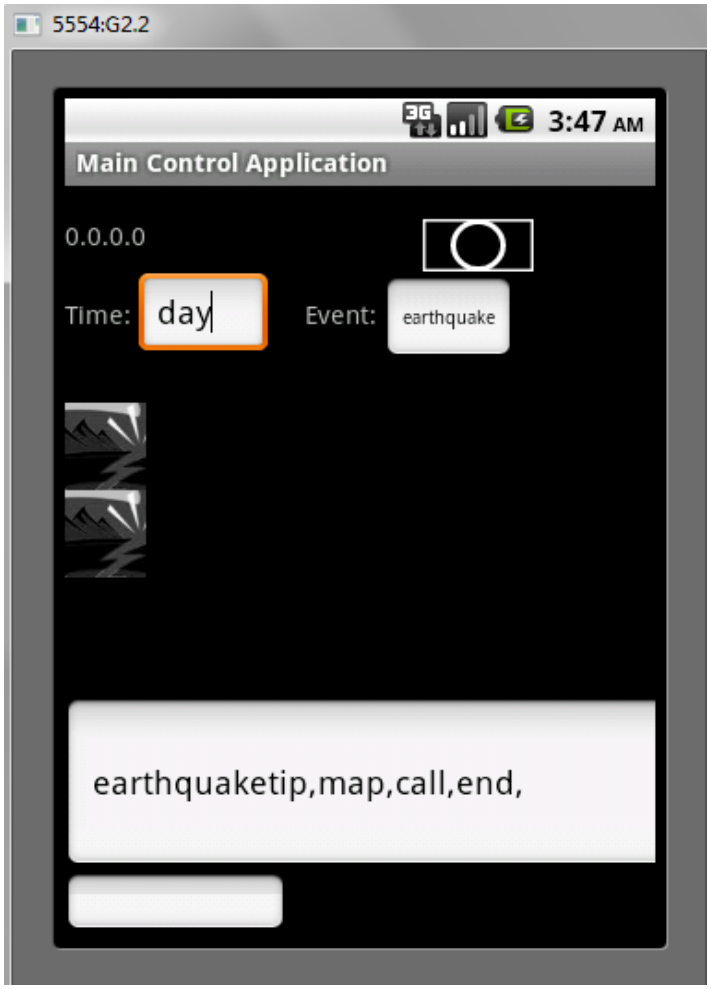

Figure 39 Return Name String to Main Control Application

(3) Transfer command to remote for compilation

After receiving the name string sent by the main control application, the web service will change the source codes based on the name string. After the modification work is done, the web service will send a success message to the main control application. After receiving the message, the main control application will be responsible for transferring the compilation command to the web service aimed at generating the installation file based on the modified source codes. We will discuss this operation in this section.

We use “Apache ant” in this process to achieve the goal of dynamic software building. Before using ant, we need to set both the JAVA_HOME_PATH and the ANT_HOME_PATH. After that, we can use ant, but the prerequisite is the existence of a build.xml file that contains several targets we need ant to do. The method to build the project and generate a build.xml file is the “android create project -p”. If we already have a project, the “android update project -p” can be used to update the change of a project. After the generation of the build.xml file, we can use an “ant” command to compile the android project, and then use an “ant debug” command to generate the installation package file. The main control application is only responsible for transferring those commands in the back end to the remote server. After receiving the commands, the compilation work will be done by the web service. Figure 40 shows the basic architecture of how commands work between the main control application and web service. This section only gives a brief introduction of how ant works in our project and how the main control application transfers such commands to the cloud server. In the web service part we will give the command transfer work flow in a different prospective. After the command transfer, the main control application will receive a confirmation message, and then will download the apk file from the remote.

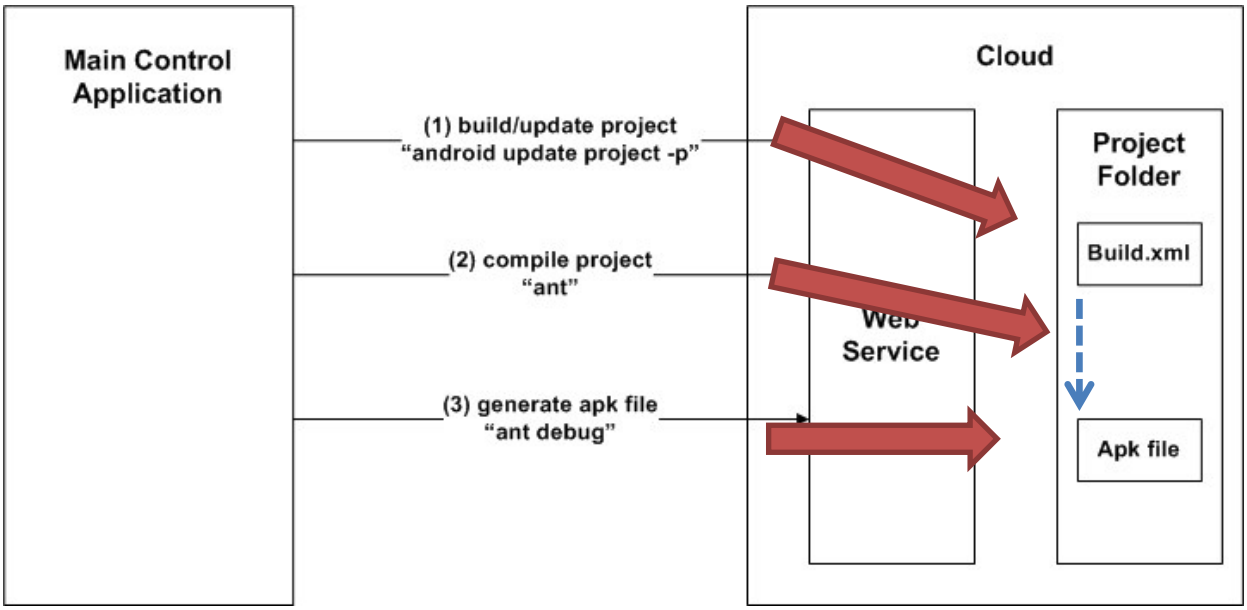

Figure 40 Architecture of the Command Workflow

(4) Downloading the Apk File from Remote

After the generation of apk in the remote is done, the main control application will be responsible for downloading its installation package from the remote.

The apk file will be put in the web service folder dynamically when generated in the cloud server. So the first step is to get the URL of the apk's location. After that, the main control application will get resources provided by the URL. To achieve that, a URL connection must be set. Then an input stream will be created to get the information to the certain URL. A temporary file will then be created and all the contents in the input stream will be written to a temporary file. Now the remote apk file has been stored in the temporary file in the mobile phone. Figure 41 shows the work flow of how the download process works. As Figure 41 shows, all the contents in the apk file have been moved to the temp file. The next step is to open the temp file and install the application.

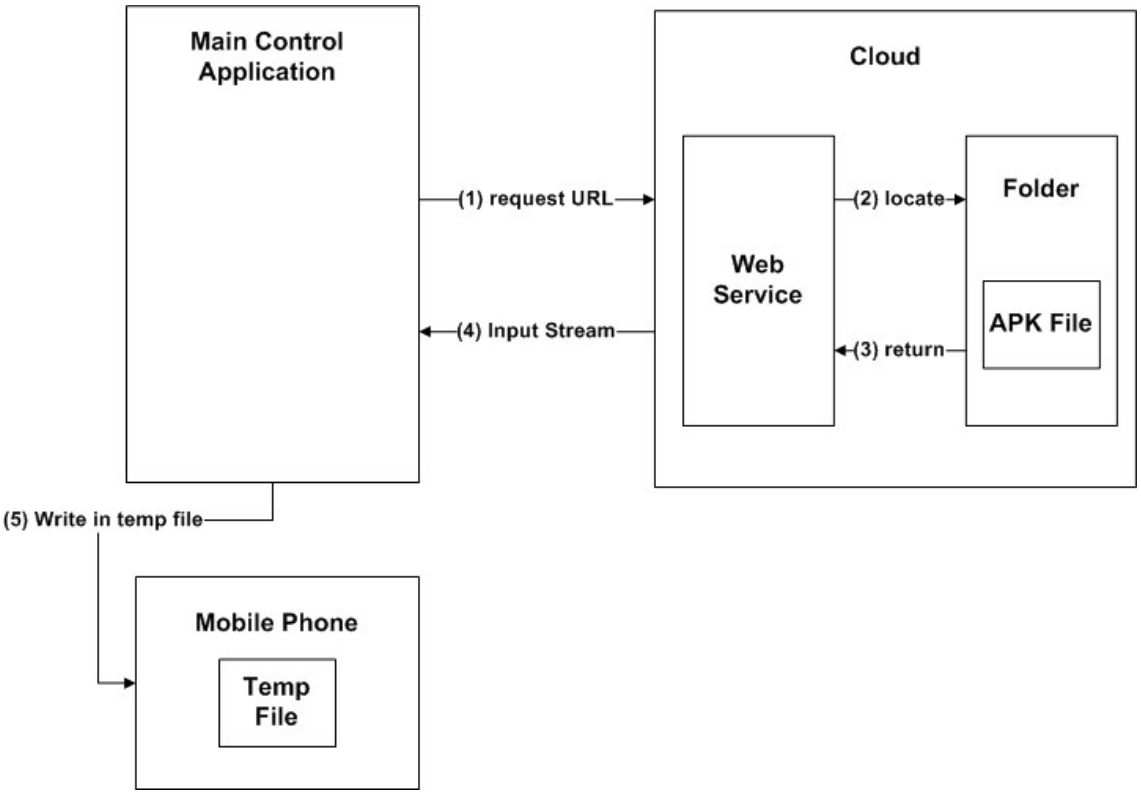

Figure 41 Workflow of the Download Process

(5) Install Application in a Mobile Phone

After downloading the apk to a local disk, the main control application will open the installation file and do the installations work immediately.

First of all, the main control application will open the temporary file on the disk. Then the main control application will check the MIME type of the file to be opened. For a different MIME type, IIS in the cloud server can provide different downloading type options. For the apk file, the type defined in IIS should be "application/vnd.android.package-archive". Then the main control application will create new task intent to set up an open file notice interface on the phone screen for users. The

generated application can now be opened. After that, the temporary file will be deleted. Figure 42 shows the installation and open file work flow.

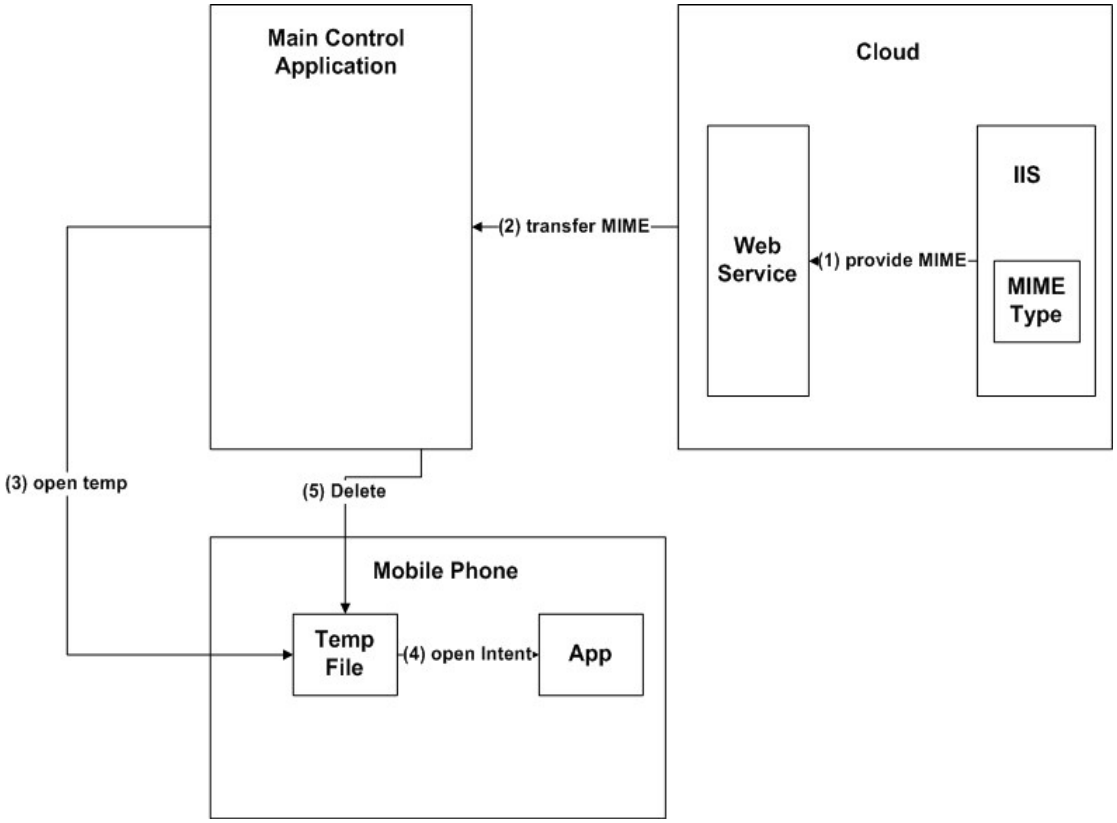

Figure 42 Workflow of Open Generated Application

After opening the downloaded application, the main control application is done with its work. However, main control application is mainly responsible for transferring the parameter to the remote and receiving the parameter from the remote. Some detailed work is done by the web service and IDE in cloud server. In the next section, we will introduce the cloud server in our research.

4.2.2.3 Cloud Server

The cloud server is a server dedicated to use in a cloud computing environment. A cloud server is different from a regular server in part in that all of the resources on the cloud server, like computing power and data storage space, can be used more efficiently. A cloud server can be booted independently and can run on its own operating system. In our research, we use an IBM cloud as the cloud server, and detailed configuration properties are listed in the following table:

Table 2 Configuration Properties in IBM Cloud Server

Server Name	IBM Cloud Server Instance
System	Windows Server 2008 R2 Datacenter
Processor	QEMU Virtual CPU version 0.9.1 2.27Ghz (2 processors)
Installed Memory (RAM)	4.00 GB
System type	64-bit Operating System
Pen and Touch	No Pen and Touch
Computer Name	Vhost0466
Full computer name	Vhost0466.dc1.co.us.compute.ihost.com

We also installed some related software for research use on the IBM cloud server. Such software includes Internet Information Service Manager (IIS), GlassFish Server, JAVA JDK, JAVA Eclipse Environment, and Android SDK. The usage and purpose of each software is listed in the following table.

Table 3 Software Names and Purpose in IBM Cloud Server

Software Name	**Purpose**
IIS	Host, publish and management of C# web service
GlassFish	Host, publish and management of Java web service
JAVA JDK	Used for Java development
JAVA Eclipse Environment	Platform to run Java program
Android SDK	Used for Android development

The name space for our cloud server is: vhost0466.dc1.co.us.compute.ihost.com. We could access our web service by adding the right path under this name space. In the next section, we will introduce the web service deployed on the cloud server.

4.2.2.4 Web Service1– Main Web Service

Web Service 1 is the main entry web service to bridge the connection between the mobile phone device and the remote cloud server. It is responsible for receiving context from the main control application, calling Web Service 2 to do the intelligent learning, returning the learning name string back to the main control application, modifying source codes in the cloud server based on the name string and executing the command sent by main control application. We will discuss the detailed function of each method in this section.

4.2.2.4.1 File Structure Introduction

In our research, we developed Web Service 1 using C# language and deploying Web Service 1 on IIS in the cloud server. Web Service 1 contains the following: the App_Code

folder, App_Data folder, App_References folder, Android Mobile Phone Application Project folder, ASMX file, Solution file and web.config file. The App_Code folder contains the execution source code that could be automatically compiled at run time. App_Data contains a data store file like the SQL Server database file, XML file, text file and any other formats the application may support. App_References contains any other URL reference of the web service. The Android project is used to be compiled when getting the commands from the main control application. ASMX files are files used with ASP.NET programming and can be opened with any program that codes in ASP.NET (like Microsoft Visual Web Developer). Web.config file is an XML format file that is used to store the configured information for the ASP.NET web application. Figure 38 shows the file list in IIS for Web Service 1.

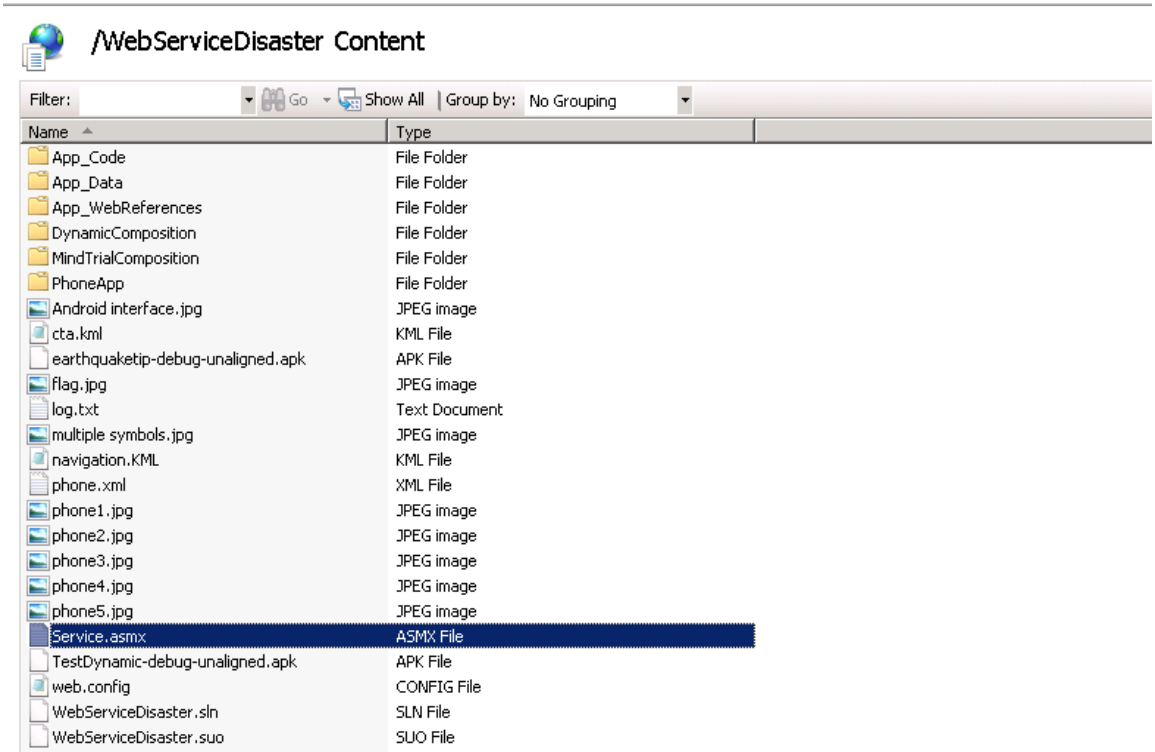

Figure 43 Folders Included in a Web Service

4.2.2.4.2 Function Introduction

Functions in Web Service 1 are responsible for receiving the context from the main control application, transferring context to Web Service 2 to do the intelligent learning, getting the function name string from the main control application to do the modification of source code, getting the command from the main control application to do the compilation and generation of the apk file, providing a name space to the apk file for the main control application to download. Figure 44 is all the methods in Web Service 1. Some methods are used in the activation component and some of them are used in communication component.

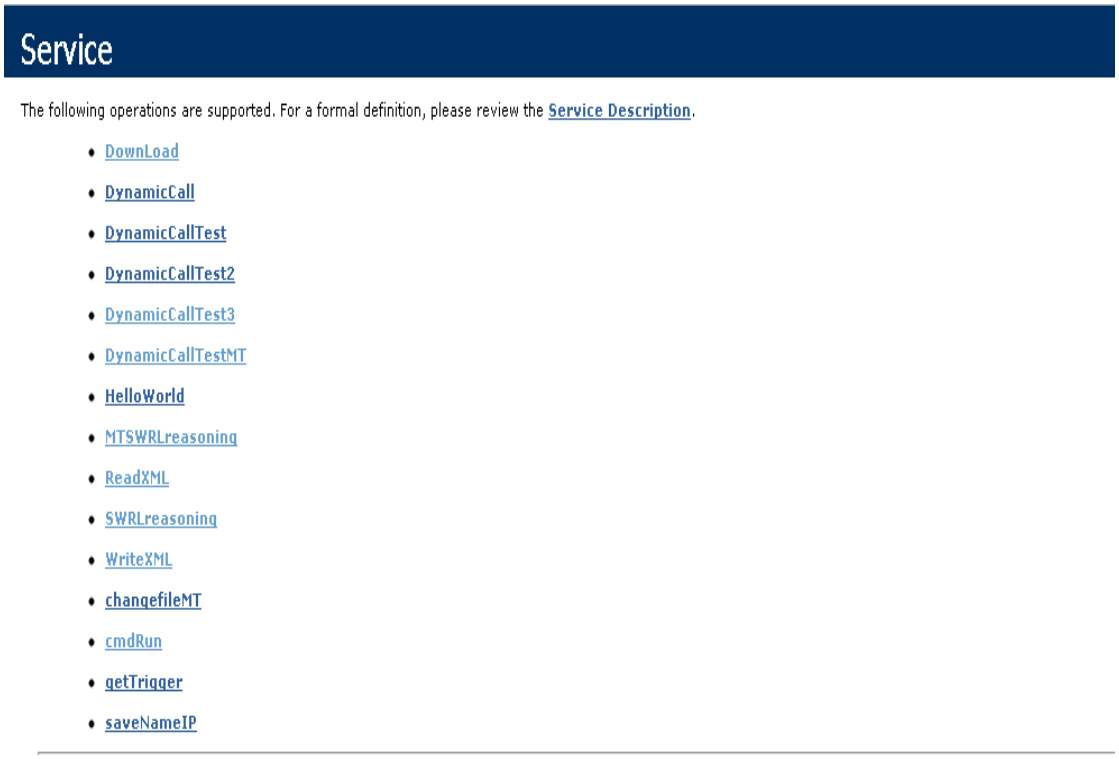

Figure 44 Methods in Web Service 1

(1) Receiving Context from the Main Control Application

Through the URL and proper web service method name, Web Service 1 could get the transferred message successfully from the main control application. The method in Web Service 1 responsible for receiving context from the main control application named "SWRLreasoning" and "MTSWRLreasoning". The former one is used in disaster scenarios and the later one is used in clinical trial scenarios. After the main control application transfers the string to the web service through those two methods, the parameters of those methods have already been valued by those string variables. After getting the sent string, the body of those methods will be responsible for doing further learning work. Therefore, we next will give a brief view of the transfer mechanism between Web Service 1 and Web Service 2.

(2) Transfer String to Web Service 2 for Intelligent Learning

In the body of methods "SWRLreasoning" and "MTSWRLreasoning", Web Service 1 will send the received string to Web Service 2. This is responsible for further intelligent learning and giving back the learning results. First of all, web service 1 should get a reference of Web Service 2 and obtain an object for Web Service 2. Then, in the body of the methods "SWRLreasoning" and "MTSWRLreasoning", such objects will call the methods written in Web Service 2 and transfer the received string variable to Web Service 2. Figure 45 shows the work flow between Web Service 1 and Web Service 2. After Web Service 1 send the parameters, Web Service 2 will receive them and do further intelligent learning based on the Ontology domain and SWRL rules.

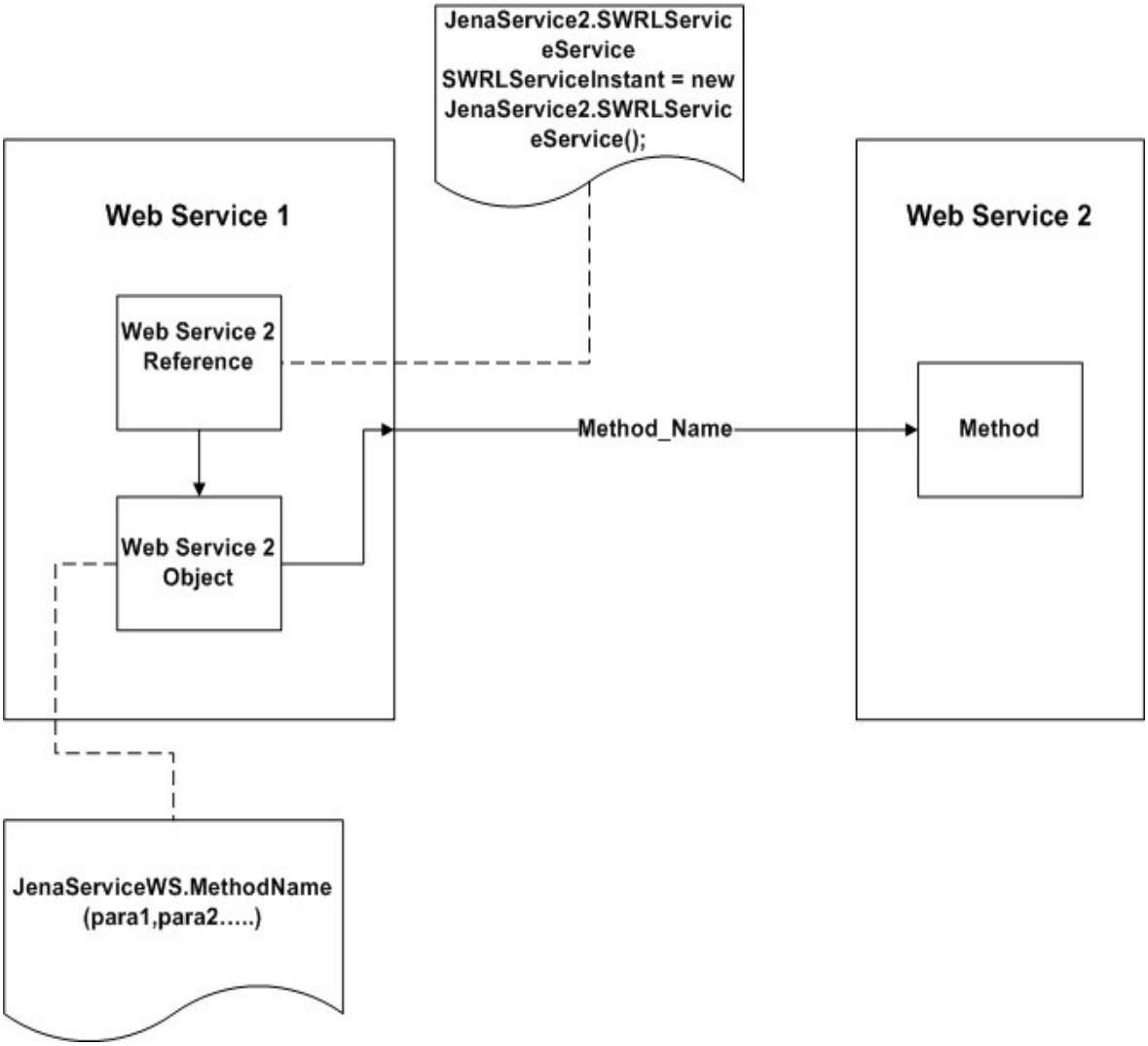

Figure 45 Workflow between Web Service 1 and Web Service 2

(3) Transfer Name String to Web Service 1 for Source Code Modification

After the main control application gets the intelligent results generated from Web Service 2, it will again send the results to Web Service 1 for modification of the source code based on the results. We pre-make two JAVA Eclipse project templates and add some marks in the source code for the convenience of modification work. We will take the clinical trial as an example to show how modification works. Figure 46 shows the main interface for clinical

trial scenario, which will be responsible for giving an overview of the sequence of appearance of functions within the application.

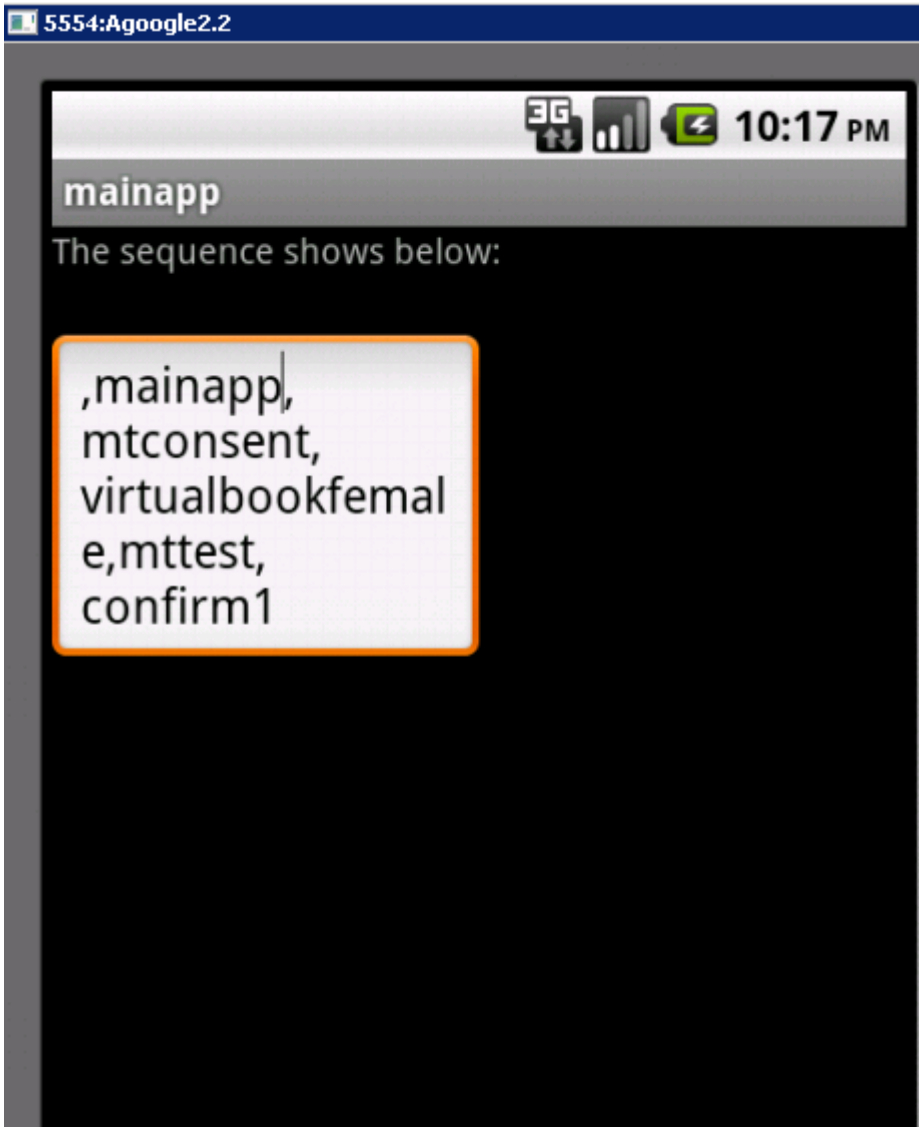

Figure 46 The Main Interface for a Generated Application

As this figure shows, the text in the edit box is the received name string from the main control application by Web Service 1. Such text is dynamically inserting to edit text. To achieve this, we make a mark in the source code. As Figure 47 shows, we make a mark above the edit text set text function, and the initial value we give to this edit text is null.

```
@Override
public void onCreate(Bundle savedInstanceState) {

    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    edittext01 = (EditText)findViewById(R.id.EditText01);

    //sequence
    edittext01.setText("");
```

Figure 47 Main Interface Code before Modification

For Web Service 1, the method responsible for receiving the name string is "DynamicCallTestMT". After receiving the name string as parameters, this method will call the "changefileMT" function to dynamically change the source code according to the name sting, that is, insert the name string to the edit text. To achieve this, we make a mark in the source code as Figure 47 shows. "//sequence" is the mark we use. The "changefileMT" function will first scan the file line by line, and once it finds the line that contains "//sequence", it will replace the next line with "edittext01.setText (name string)", where the name string is the receiving string from the main control application. Figure 48 shows the source code after the modification.

```
@Override
public void onCreate(Bundle savedInstanceState) {

    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    edittext01 = (EditText)findViewById(R.id.EditText01);

    //sequence
    edittext01.setText(",mainapp,mtconsent,virtualbookfemale,mttest,confirm1");
```

Figure 48 Main Interface Code after Modification

The above modification is regards the main interface. Next, we look at how to modify each function and make the activation sequence for those functions.

First of all, Web Service 1 will split the received name string from the main control application into every single word, and store them in an array. As showed above, each function named in the name string is divided by a comma, so the first step Web Service 1 should do is to split the name sting by comma. Figure 49 shows the detailed code about how to do the split work. After that, each word will be stored in to an array, that makes it easy for getting each element easily, However, the most important thing is that, through this way, we can easily identify which function would be the next one based on the current one by looking at the array index.

```
applist = appnamestring.Trim().Split(',');

        for (int i = 0; i < applist.Length; i++)

        {

            if (applist[i] != null)

            {

                applisttemp[i] = applist[i];

            }

            else

                break;
```

Figure 49 Split Name String into Single Word

After the split work, Web Service 1 will call the "changefileMT" method to modify the function code to determine which function will be the next based on the current function. Figure 50 shows the source code and mark in function code before modification. As we can see, when before modification, the new intent destination class and original class is the same. And the mark is pre-defined which is the same in the name in ontology and in name string.

```
new Handler().postDelayed(new Runnable(){

public void run() {
    // TODO Auto-generated method stub
    //mtconsent
    Intent it = new Intent(mtconsent.this,mtconsent.class);
    startActivity(it);  }, 10000);
```

Figure 50 Mark and Source Code for Function before Modification

Figure 51 shows the workflow of generation work for function source code. Splitting string into array will be done first after Web Service 1 getting the name string from main control application. After that, the function will retrieve the first and the second element and send to “changefile” method. In this method, it will first locate the source code file in the disk of server, and then open the file and scan it line by line. Once it finds a line which contains a comment statement with the same name as array[i], the function will begin to replace the next line with the statement:

“Intent it = new Intent(" + array[i] + ".this," + array[i+1] + ".class);”, which means that current activity is array[i], and the next activity would be array[i+1]. After that, increase “i” by 1 and do the loop again.

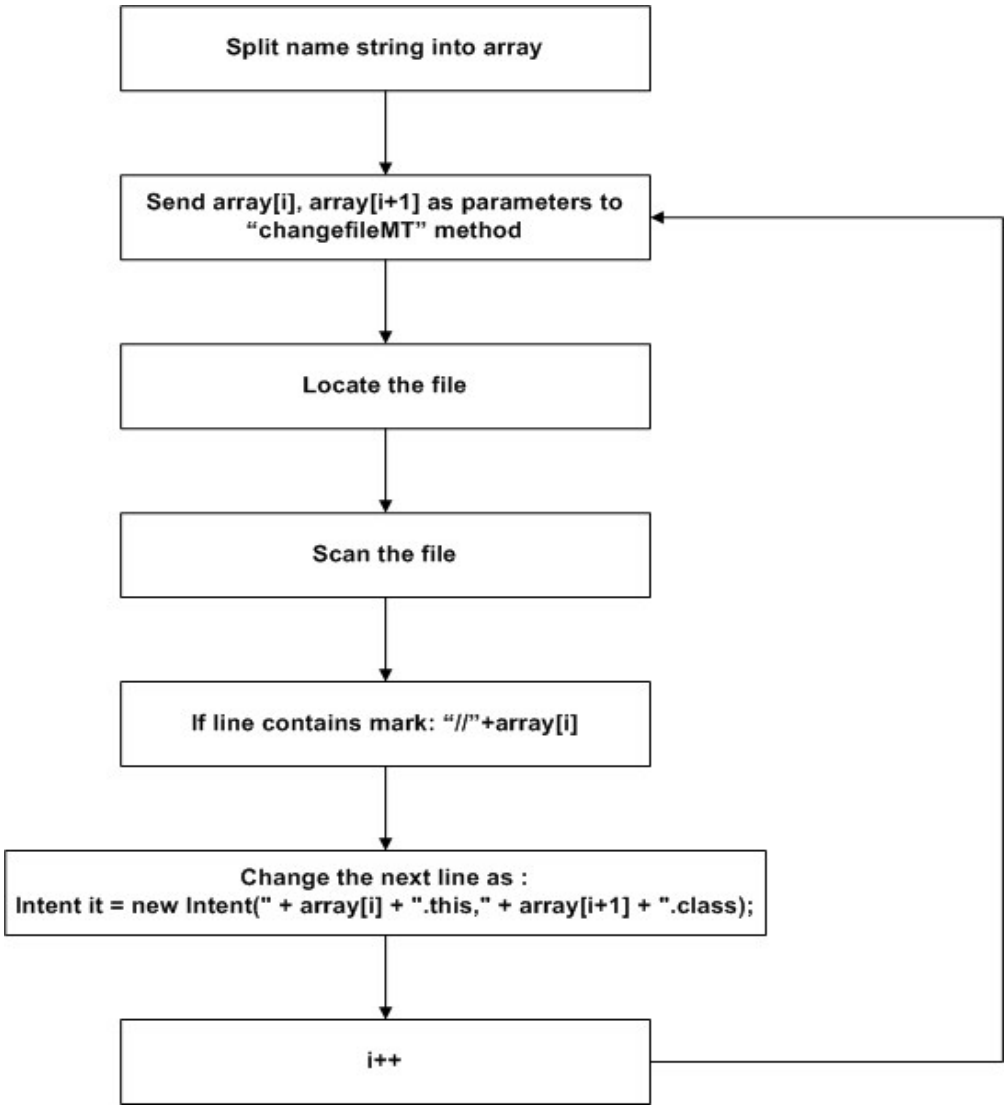

Figure 51 Modification of Function Source Code Workflow

Figure 52 shows the mark and source code for function after modification. As we can see in the figure, the current activity maintains the same, and the next activity has been changed. Another thing needs to be mentioned is that the number "10000" in Figure 52. It means that current activity will be turned to next activity by ten seconds. The transition time here is fixed, in the future work, we plan to make the transition time be dynamically modified by the change of context for individuals.

```
public void run() {
    // TODO Auto-generated method stub
    //mtconsent
    Intent it = new Intent(mtconsent.this,virtualbookfemale.class);
    startActivity(it);  }), 10000);


}
```

Figure 52 Mark and Source Code for Function after Modification

The workflow of the modification, as shown in Figure 53, indicates that activity will transition one by one until to the end application and go back to the beginning. The sequence is defined by the SWRL rule, we will discuss it later.

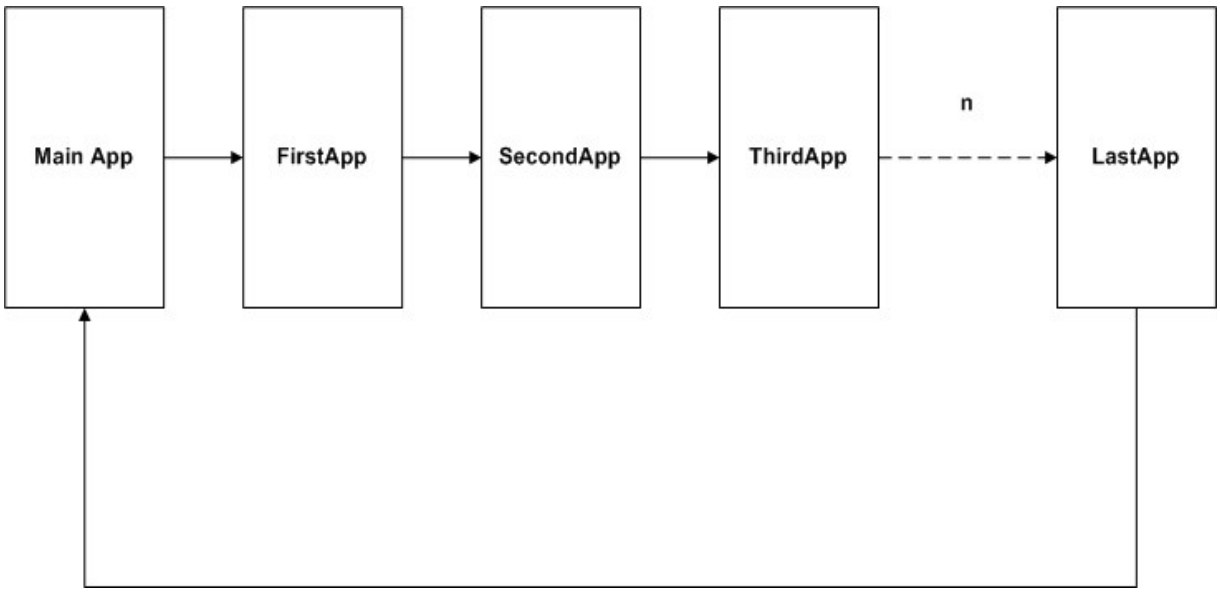

Figure 53 Transition from App to App

(4) Get the Command Request from Main Control Application and Execute

After modification of source code, it is time to compile the code and generate the installation package for downloading. As soon as main control application gets the positive response after Web Service 1 done the modification work, it will send some commands to Web Service 1 to request a compilation operation on the existing code and project. The commands are showed in the table below. The method to combine different command statement together is the use of "&" symbol: "cmd1 & cmd2 & cmd3".

Table 4 Commands and their Purpose

Command Number	Command Statement	Command Purpose
Cmd1	cd.. & cd.. & cd..& cd c:\\Program Files (x86)\\Android\\android-sdk\\tools & android update project -p c:\\inetpub\\wwwroot\\WebServiceDisaster\\DynamicComposition	Locate the android folder first and update the project if any change has been made. Finally generate the build.xml file for ant command
Cmd2	cd c:\\inetpub\\wwwroot\\WebServiceDisaster\\DynamicComposition & c:\\apache-ant-1.8.2\\bin\\ant	Locate the project folder, do ant command on it, compile the project
Cmd3	cd c:\\inetpub\\wwwroot\\WebServiceDisaster\\DynamicComposition & c:\\apache-ant-1.8.2\\bin\\ant debug	Locate the project folder, do ant debug command on it, generate the apk installation file

In Web Service 1, method "cmdRun" is used to handle the uploading commands and execute what commands request. The workflow of execute command are showed in Figure 54. First of all, "cmdRun" method will create a new process. Then some properties must be set to complete the initialization work. We do not want to show the command window out, so set this option as false; We set the program name as "cmd.exe"; We do not use Shell, so set this option as false; We need Standard Error report, Standard Input and Standard Output, so

we set those properties as true. Then we start the process with the input command as the initialization properties and mark an end symbol after all the command has been executed. After input operation, the process will send all the execution results to the users. In the end, the process will be closed.

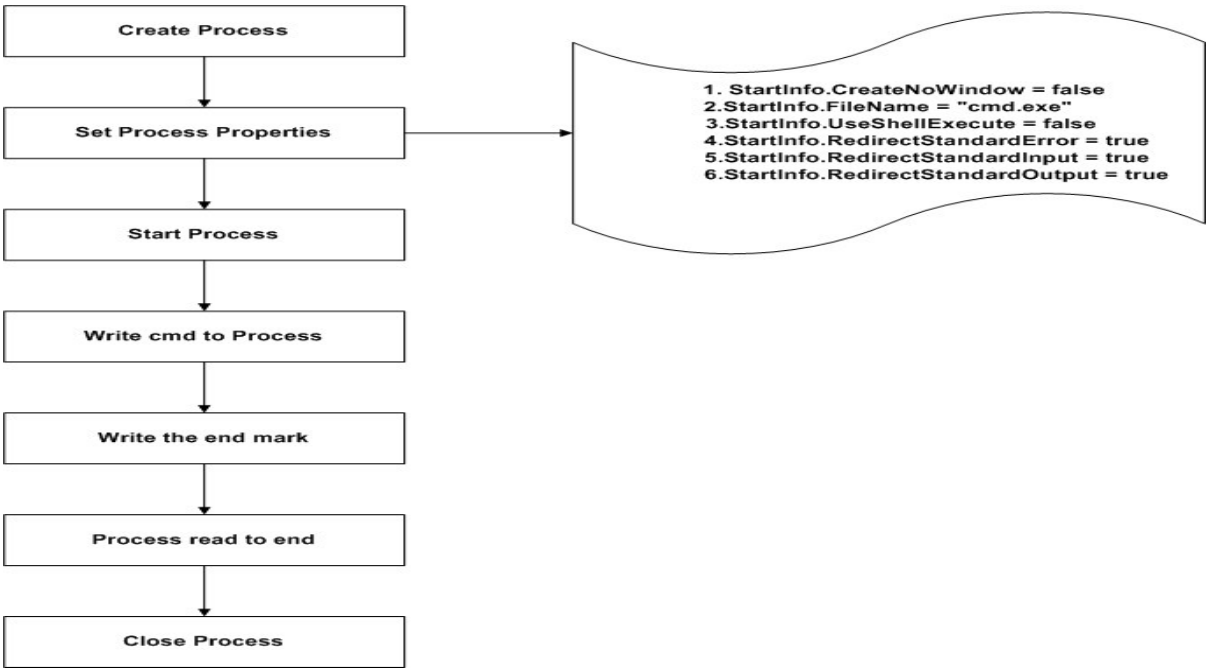

Figure 54 Command Execution Workflow

After the execution of command, build.xml file and apk installation package will be generated and showed on the project folder. Figure 55 shows the contents in project folder. After the generation of apk installation file, we will discuss the downloading part in next section

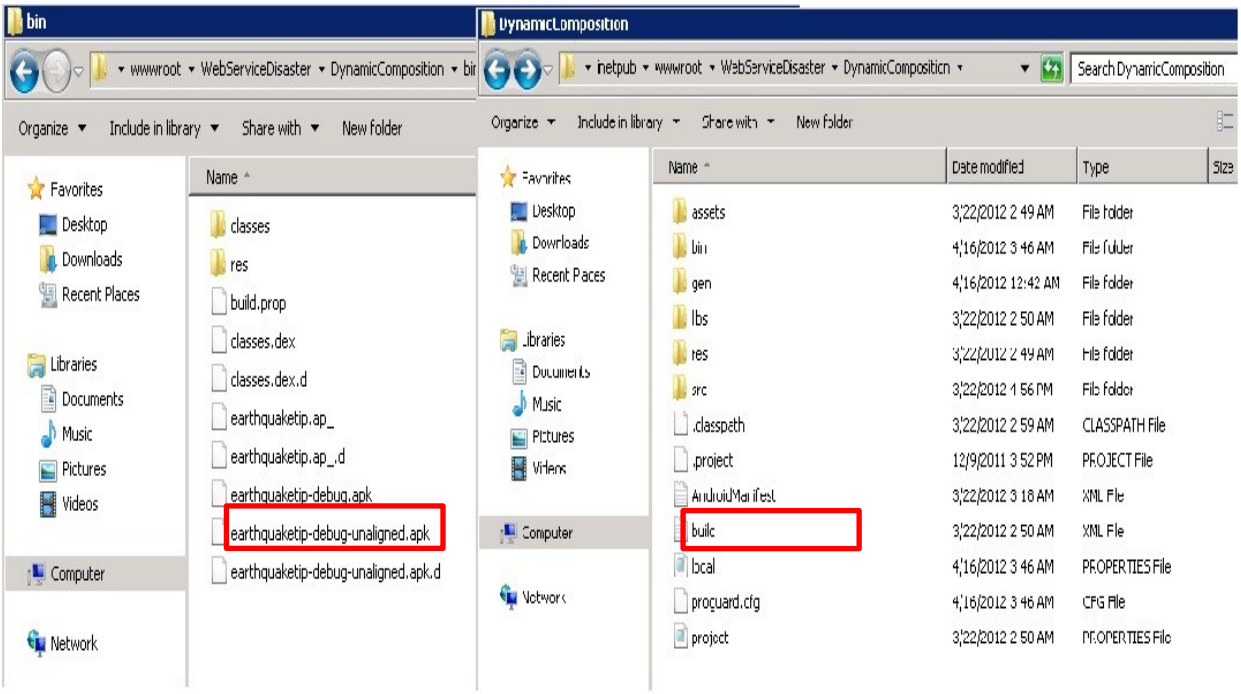

Figure 55 Project Folder after Execution

(5) Provide the apk File Downloading URL

Apk installation package is located in the Android project folder. And Android project folder is located in the web service folder. Web service folder is located in the IIS web application folder. So the format of URL should be:

IIS host name + web service name + Android Project name + apk name.

Table 5 shows the name analysis for apk file downloading URL.

Table 5 Downloading URL Analysis for Apk File

Name	Address
IBM Cloud Host Name	vhost0466.dc1.co.us.compute.ihost.com
Web Service Name	WebServiceDisaster
Android Project Name	DynamicComposition
Apk name	bin/earthquaketip-debug.apk
Whole URL address	
vhost0466.dc1.co.us.compute.ihost.com/WebServiceDisaster/ DynamicComposition/bin/earthquaketip-debug.apk	

For now, all the main methods in Web Service 1 have been introduced. Web Service 2 has a close relationship with Web Service 1. But before the introduction of Web Service 2, we will give the concept and detailed introduction of Ontology and SWRL rules, which plays the most important role in Web Service 2.

4.2.2.5 Ontology Domain

4.2.2.5.1 Overview

In computer science and information science, an ontology formally represents knowledge as a set of concepts within a domain, and the relationships between those concepts. It can be used to reason about the entities within that domain and may be used to describe the domain. Ontology domain definition is one of the most important parts in our research to achieve the dynamic and automatically operation goal. In our research, we have made two scenarios, and we have made a specific ontology domain for each scenario. The tool we used to design the ontology is protégé 3.4.4, and we will give detailed introduction of disaster scenario in this section.

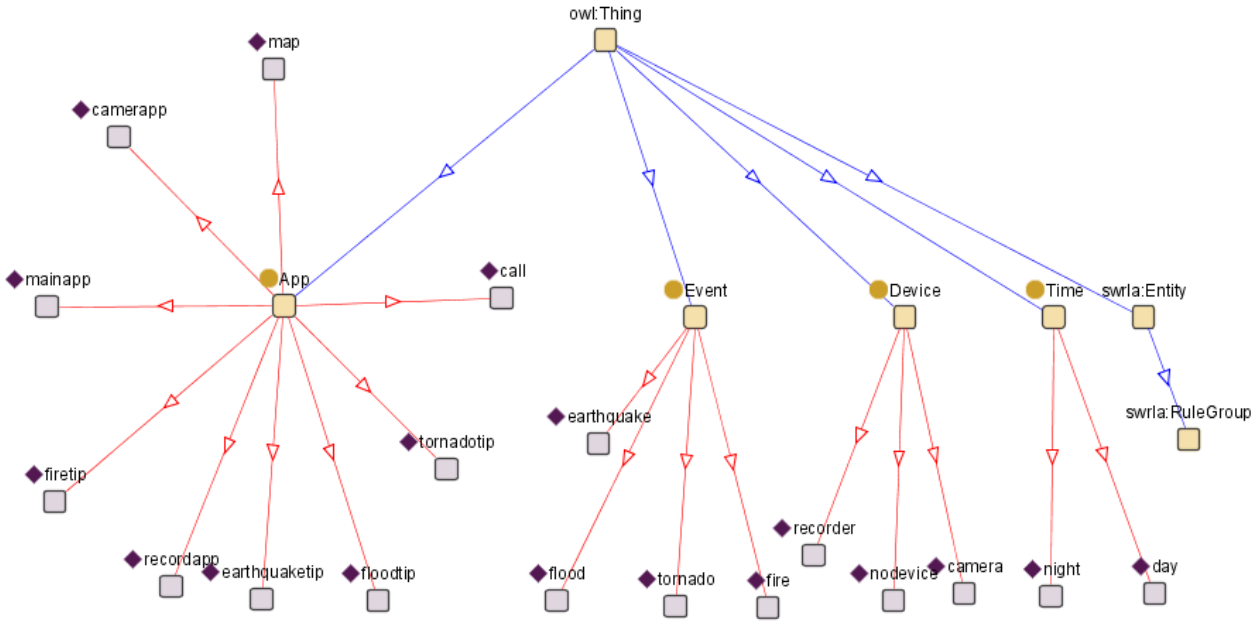

Figure 56 Ontology Relationship Diagram for Disaster Scenario

4.2.2.5.2 Class, Individual and Property

As Figure 56 shows, the node which in yellow represents each class we defined. And the node which in purple represents each instance which belong to a certain class. There also exists some properties to build the relationship between different classes. Thus those relationships will be inherited to instances which can be seen as individuals of class. Table 6 shows classes, individuals we defined in disaster handling scenario. Table 7 shows properties we defined in disaster handling scenario.

Table 6 Classes and Individuals Defined in Disaster Ontology

Class	**Individual**
App	mainapp
	call
	camerapp
	earthquaketip
	firetip
	floodtip
	map
	Recordapp
	tornadotip
Device	camera
	nodevice
	recorder
Event	earthquake
	fire
	flood
	tornado
Time	day
	night

Table 7 Properties Defined in Disaster Ontology

Property1	**Property2**	**Property3**	**Property4**	**Property5**
hasApp	hasNext	useDevice	useTime	hasMain

4.2.2.5.3 Triple: Subject, Predicate and Object

Tables above show the definition of classes, individuals and properties. Then we will show how properties connect different classes and individuals together. Because ontology is a special concept we defined, it should be made up with a triple as subject, predicate and object. In general, subject and object can been seen as classes and predicate can be seen as properties. With the usage of properties, different classes / domains could be related together. Table 8 shows the relationship between different classes by the use of properties. As the figure shows, once the relationship between classes has been set up, the property name will be showed under the individual, which is belong to a certain class. We can manually choose the proper individuals and add them to object side. Figure 57 shows an example of the relationship between individuals. In the coding level, we actually operate on individuals or instances instead of classes.

Table 8 Triple Defined for Ontology in Disaster Scenario

Subject(Classes)	**Predicate(Properties)**	**Object(Classes)**
Event	hasApp hasMain	App
App	hasNext	App
App	useDevice	Device

App	useTime	Time

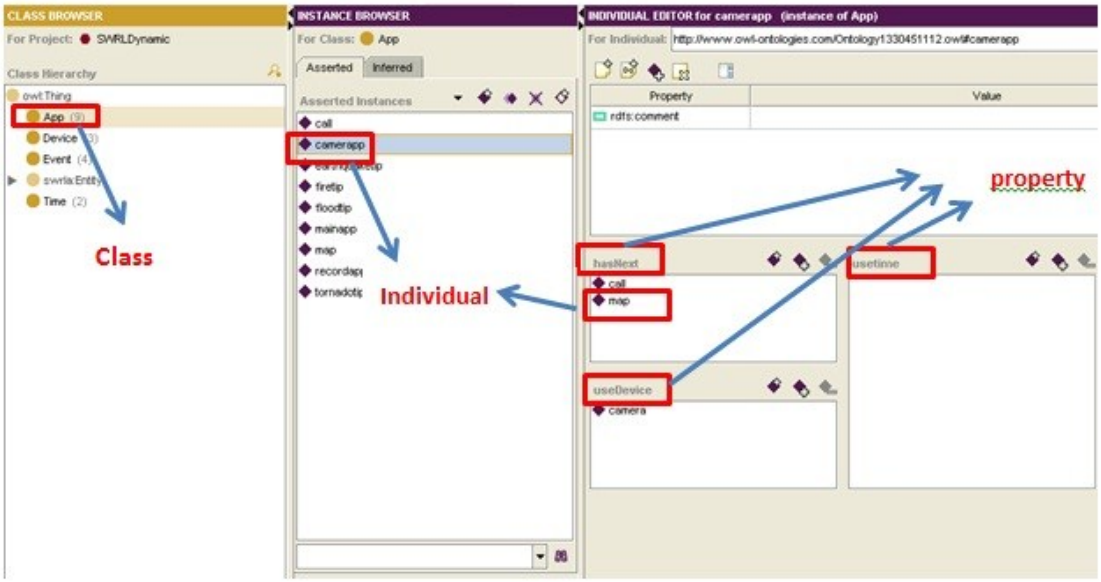

Figure 57 Property Used between Individuals

4.2.2.6 SWRL Rules

4.2.2.6.1 Overview

SWRL (Semantic Web Rule Language) is a proposal for a Semantic Web rules-language, combining sublanguages of the OWL Web Ontology Language (OWL DL and Lite) with those of the Rule Markup Language (Unary/Binary Datalog). (Wikipedia) SWRL is a very crucial part in our research. The dynamic and intelligent features are all implemented by the use of SWRL rules. SWRL rules provide some constrains based on the current individuals and the relationship between them. Machine can choose the most appropriate option based on the SWRL rule. In this section, we will use disaster scenario to illustrate how SWRL rules help to make the application framework intelligent.

4.2.2.6.2 Make the Rule

The application generated in disaster scenario has the feature that each function will be generated automatically and the activation sequence for those functions are dynamically changing based on the changing of input context. For doing that, we need to formulate some rules to adding some constrains on the relationship between different individuals to make sure that when certain constrain condition happens, the result would be the best answer for that certain constrain. Table 9 is a table to show the idea about how to make the inference prototype. As we can see in the table, the inference is based on constrains in each level. We can make a conclusion that if the more constrains we make in the rules, the more accurate and intelligent inference result we could get.

Table 9 Idea about How to Make the Rule for Disaster Scenario

Rule1 idea	If one disaster event comes, and user in the situation boundary, directly go to main function first (user situation identification)
Rule2 idea	If such disaster has an related function about the tip to rescue from such disaster in ontology, then main function transition to such function, we mark it fun1 (event situation identification)
Rule3 idea	If fun1 has next function in ontology, and such next function use a certain device which current mobile phone has the same capability, then fun1 makes a transition to such function, and we mark it fun2 (device identification)
Rule4 idea	If fun2 has next function in ontology, and such next function must be used in a certain time which is the same as the current context time, then fun2 makes a transition to such function, and we mark is as fun3 (time situation identification)

Above table only shows the human idea about how to make the rule to get the appropriate we want. But machine cannot understand the human high level languages. So we need to use machine readable language to do this work. SWRL rule can be made as the following table shows.

Table 10 SWRL Rule for Dynamic Idea for Disaster Scenario

Rule1	User (?u) ^ Event(?x) ^ Involving (?u,?x) ^hasMain (?x,?y)-> sqwrl: select App(?y)
Rule2	Event(?x) ^ App (?y) ^ hasNext(?y,?z) ^ hasApp(?x, ?z)-> sqwrl: select (?z)
Rule3	App(?x) ^ hasNext(?x, ?y) ^ useDevice(?y, ?z) -> sqwrl:select(?y)
Rule4	App(?x) ^ hasNext(?x, ?y) ^ useTime(?y, ?z) -> sqwrl:select(?y)

In Rule1, x could be any type of disaster pre-defined in ontology. To make y is “main function”, we must make the object part for “hasApp” predicate for any disaster subject as “main function”, thus, we can make sure that when any type of disaster comes, the first function is “main function”. So y in rule1 is fixed, that is “main function”.

In Rule2, x could be any type of disaster pre-defined in ontology. Y should be “main function”. And then we will do a determination. If “main function” has next function, and such function is related the event by the predicate “hasApp”, then we will select such function z. Otherwise do nothing.

In Rule3, x could be any function we got from Rule2. Then we will do a determination. If such function has next function and such next function will use device z which is compatible by mobile phone, then we will choose such function y. Otherwise do nothing.

In Rule4, x could be any function we got from Rule3. Then we will do a determination. If such function has next function and such next function will be used in time z, then we will choose such function y. Otherwise we do nothing.

Figure 58 is an example the workflow diagram to show how SWRL rule help to dynamically choose the next function based on the current context for the current function. Based on different context, the inference result might be different. It is a kind of intelligent inference according to the current context and situation.

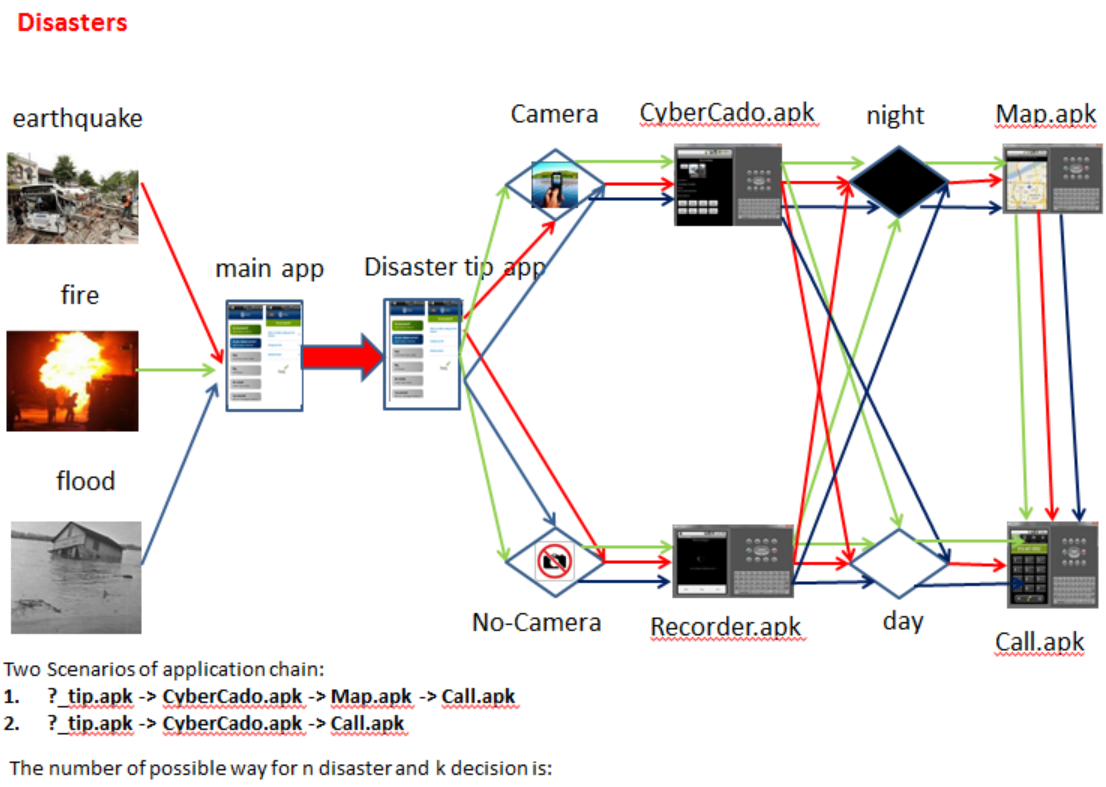

Figure 58 An Example of Workflow for Disaster Scenario

After the introduction of Ontology and SWRL rules, we will introduce Web Service 2, which is responsible for getting the context from Web Service 1 and implement SWRL rules on Ontology in code level.

4.2.2.7 Web Service2– Intelligent Web Service

4.2.2.7.1 Overview

Web Service 2 is responsible for getting the context string from Web Service 1 and implementing SWRL rules on Ontology to getting the inference results and returning them to users. This section we will give a detailed illustrate of how Web Service 2 works.

4.2.2.7.2 Composition of File

We use Java language on NetBeans programming environment with several libraries using in SWRL to do the implementation. The detailed list is showed in Table 11.

Table 11 File List for Web Service 2

File Type	File Name	File Usage
Web Page	Index.jsp	To open a web page when accessing
Web Service	SWRLService.java	Methods to do SWRL rules reasoning work
Library	edu.stanford.smi.protegex.owl	Library for using SWRL
Library	Jess.jar	Library for using Jess, which is the rule engine to execute SWRL rules or

		SQWRL queries
Library	Protégé.jar	Library for using Ontology load exception
Library	JDK 1.6	Library for using java environment, java tools and java class library
War	SWRLService.war	A package file including all the contents of web service for deploying on the server

4.2.2.7.3 Functions introduction

Functions in Web Service 2 are mainly responsible for executing SWRL rules on an owl ontology file to get the inference results. Based on a different scenario, Web Service 2 will operate on a different ontology domain that has been pre-defined and uses different SWRL rules on them. In this section we will give a detailed introduction on how to implement SWRL reasoning.

(1) Getting Context & SWRL Rules for a Disaster Scenario

After Web Service 1 sends the context to Web Service 2, the method “findApp” is responsible for getting such context and calling the OWL ontology file related to the disaster domain in the disk to do the SWRL reasoning for getting the proper application. The work flow of how to read the ontology file and do the intelligent learning is shown in Figure 59.

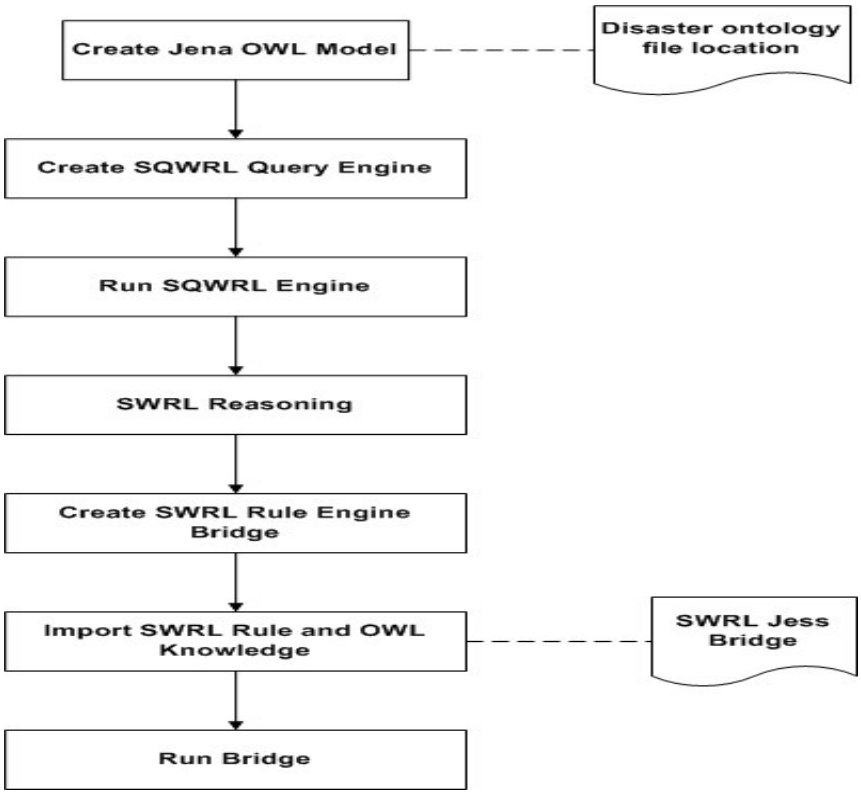

Figure 59 Workflow of Implementing Reasoning (Disaster)

The SWRL rules are implemented as shown in the table 12. The basic idea is like the way we design the SWRL rule before, but the only difference here is that all the SWRL rules are implemented in Java language.

Table 12 Implement SWRL Rules in Java Language (Disaster Scenario)

Rule1	queryEngine.runSQWRLQuery("FirstQuery", "Event(" +eventtype + ") ∧ hasMain("+eventtype+",?y) → sqwrl:select(?y)");
Rule2	queryEngine.runSQWRLQuery("SecondQuery", "Event(?x) ^ App(" +firstAppArray[0]+ ") ∧ hasNext("+firstAppArray[0]+",?y)∧ hasApp (?x,?y) → sqwrl:select(?y)");
Rule3	queryEngine.runSQWRLQuery("ThirdQuery", "App("+secondAppArray[0]+") ∧ hasNext("+secondAppArray[0]+",?y) ∧useDevice(?y,"+ device +")→ sqwrl:select(?y)");
Rule4	queryEngine.runSQWRLQuery("fourthQuery", "App("+thirdAppArray[0]+") ∧ hasNext("+thirdAppArray[0]+",?y) ^ useTime((?y,"+ time +") → sqwrl:select(?y)");

(2) Getting Context & SWRL Rules for a Clinical Trial Scenario

After Web Service 1 is sending the context to Web Service 2, the method "findAppMT" is responsible for getting such context and call another owl ontology file related to clinical trial domain in the disk to do the SWRL reasoning. The workflow of how to read the ontology file and do the intelligent learning is showed in Figure 60.

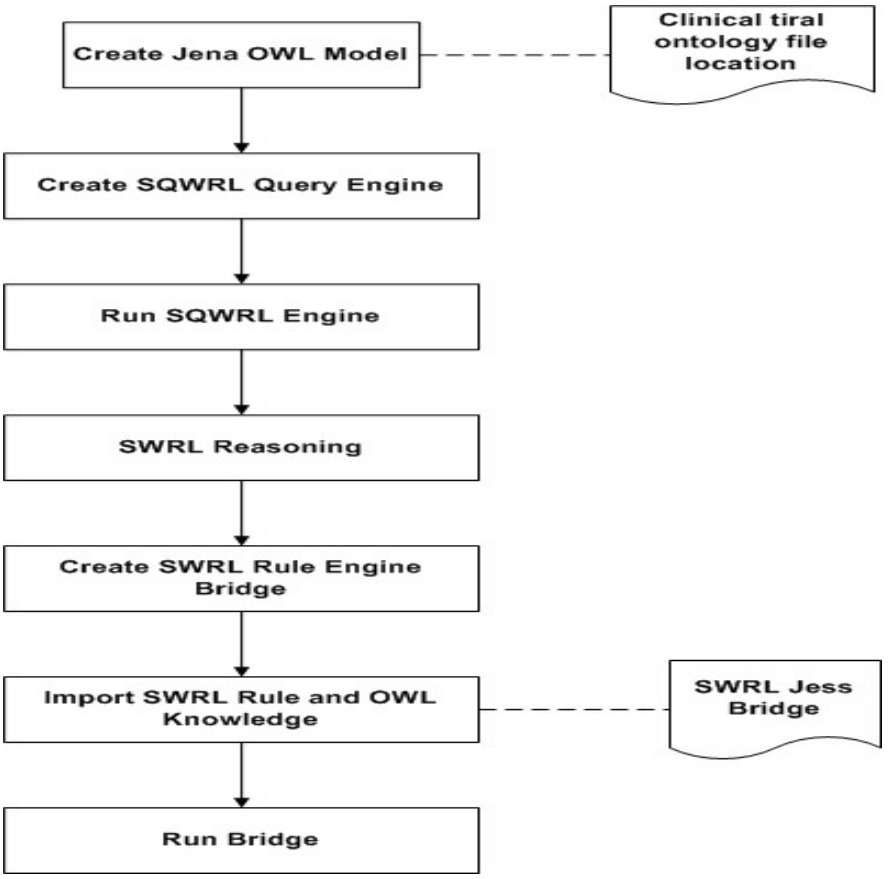

Figure 60 Workflow of Implementing Reasoning (Clinical Trial)

The Inference concepts that are described using human language are listed in Table 13. The SWRL design idea for clinical trial scenario is shown in Table 14, and the SWRL rules are implemented as shown in the Table 15.

Table 13 Idea about How to Make the Rule for Clinical Trial Scenario

Rule1 content	If schedule time is coming, directly go to the main interface function (fun1)
Rule2 content	If the next function of fun1 exists, go to next. In the ontology, we will make the next one of the main interface function fixed as inform consent. So fun2 must be inform consent
Rule3 content	If fun2 has the next function, and such function has a specified gender z, then choose such function as fun3
Rule4 content	If fun3 has the next function, and such function has a specified age z, then choose such function as fun4
Rule5 content	If fun4 has the next function, and such function has a specified history z, then choose such function as fun5

Table 14 SWRL Rule for Dynamic Idea for Clinical Trial Scenario

Rule1	schedule(?x) hasApp (?x,?y)-> sqwrl: select App(?y)
Rule2	app(?x) ^ hasNext(?x,?y) -> sqwrl: select (?y)
Rule3	app(?x) ^ hasNext(?x, ?y) ^ hasGender(?y, ?z) -> sqwrl:select(?y)
Rule4	app(?x) ^ hasNext(?x, ?y) ^ hasAge(?y, ?z) -> sqwrl:select(?y)
Rule5	app(?x) ^ hasNext(?x, ?y) ^ hasHistory(?y, ?z) -> sqwrl:select(?y)

Table 15 Implementation of SWRL Rules in Java Language (Clinical Trial Scenario)

Rule1	queryEngine.runSQWRLQuery("FirstQuery", "schedule(" +schedule + ") ∧ hasApp("+schedule+",?y) → sqwrl:select(?y)");
Rule2	queryEngine.runSQWRLQuery("SecondQuery", "app(" +firstAppArray[0]+ ") ∧ hasNext("+firstAppArray[0]+",?y) → sqwrl:select(?y)");
Rule3	queryEngine.runSQWRLQuery("ThirdQuery", "app(" +secondAppArray[0]+ ") ∧ hasNext("+secondAppArray[0]+",?y)∧ hasGender(?y,"+gender+") → sqwrl:select(?y)");
Rule4	queryEngine.runSQWRLQuery("fourthQuery", "app("+thirdAppArray[0]+") ∧ hasNext("+thirdAppArray[0]+",?y) ∧ hasAge(?y," + age +") →

	sqwrl:select(?y)");
Rule5	queryEngine.runSQWRLQuery("fifthQuery", "app("+fourthAppArray[0]+") ∧ hasNext("+fourthAppArray[0]+",?y) ∧ hasHistory(?y," + history +") → sqwrl:select(?y)");

4.3 Inactivation Component

4.3.1 Overview

The inactivation component is the component to uninstall the application in a certain period of time when such application is not used and deletes the related installation package when such application is uninstalled. It provides a method to make sure the mobile phone disk has enough space for further read and write operations.

4.3.2 Uninstall Function

The uninstall function is responsible to uninstall the designated application that has finished its job. The basic workflow about the uninstall function is showed in Figure 61.

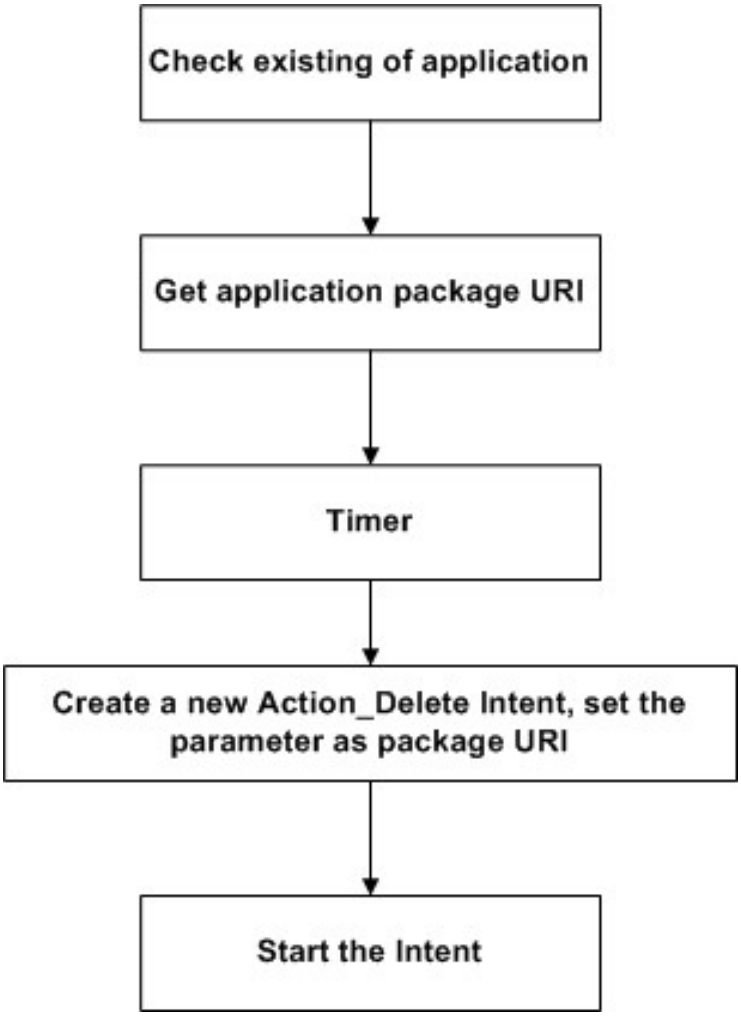

Figure 61 Workflow of Uninstall Function

As Figure 61 shows, we first use the “checkProgram(package,activity)” method to check the existence of the application. Then we use the “Uri.parse(package)” to get the package URI of a certain application. Next we will check the timer, and when it is coming, so then we will open a new intent by using the method “new Intent(Intent.ACTION_DELETE, packageURI)”. Finally, we start the intent and then the uninstallation method can be done.

4.3.3 Delete Function

The delete function is responsible for deleting the installation package for a certain application that has been uninstalled successfully. This function is used to achieve the goal of cleaning the mobile phone disk by deleting unused installation packages to keep enough space on the disk for future read and write operation. The basic work flow of the delete function is shown in Figure 62.

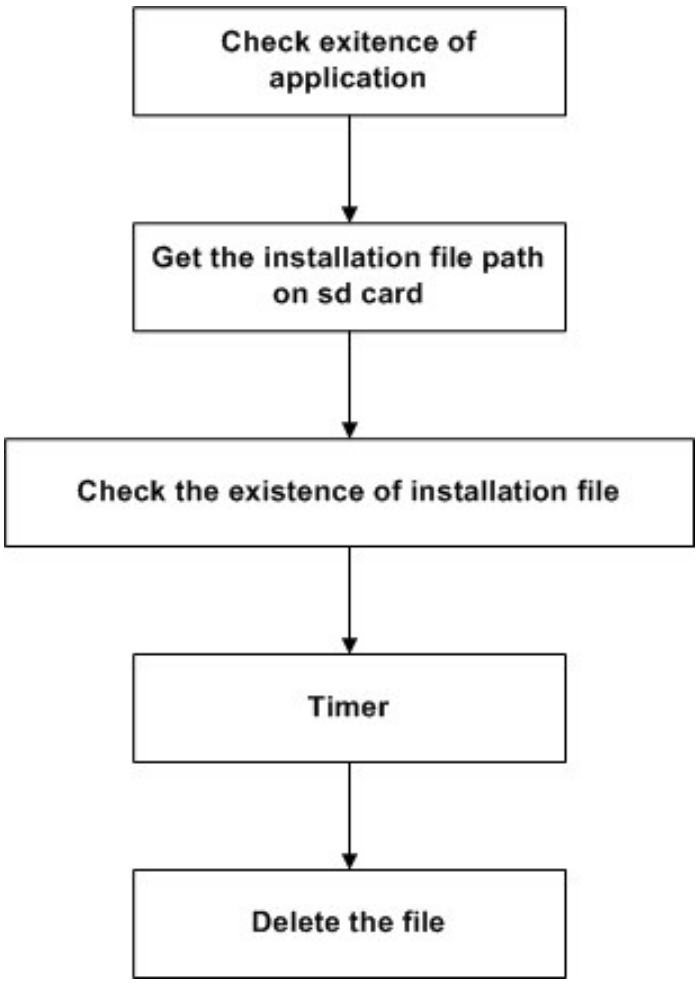

Figure 62 Workflow of Delete Function

As showed in Figure 62, we first check the existence of the application. If it exists, then we are going to delete the installation file. Otherwise, we will do nothing. Then we will find the saving path of the installation file, and we are going to check the existence of such installation file. If this file exists, then after the timer is activated, we will execute the delete function to delete the file.

4.4 Communication Component

4.4.1 Overview

The communication component is the component for sharing the file between different mobile phones using a Bluetooth connection and doing the synchronization work on the file if some contents have been changed.

4.4.2 Transfer Function

The transfer function is used to complete the sending/receiving work between different mobile phone devices. In this function, we use Bluetooth as a transfer method to establish the connection between different devices. Figure 63 shows the workflow of the transfer function.

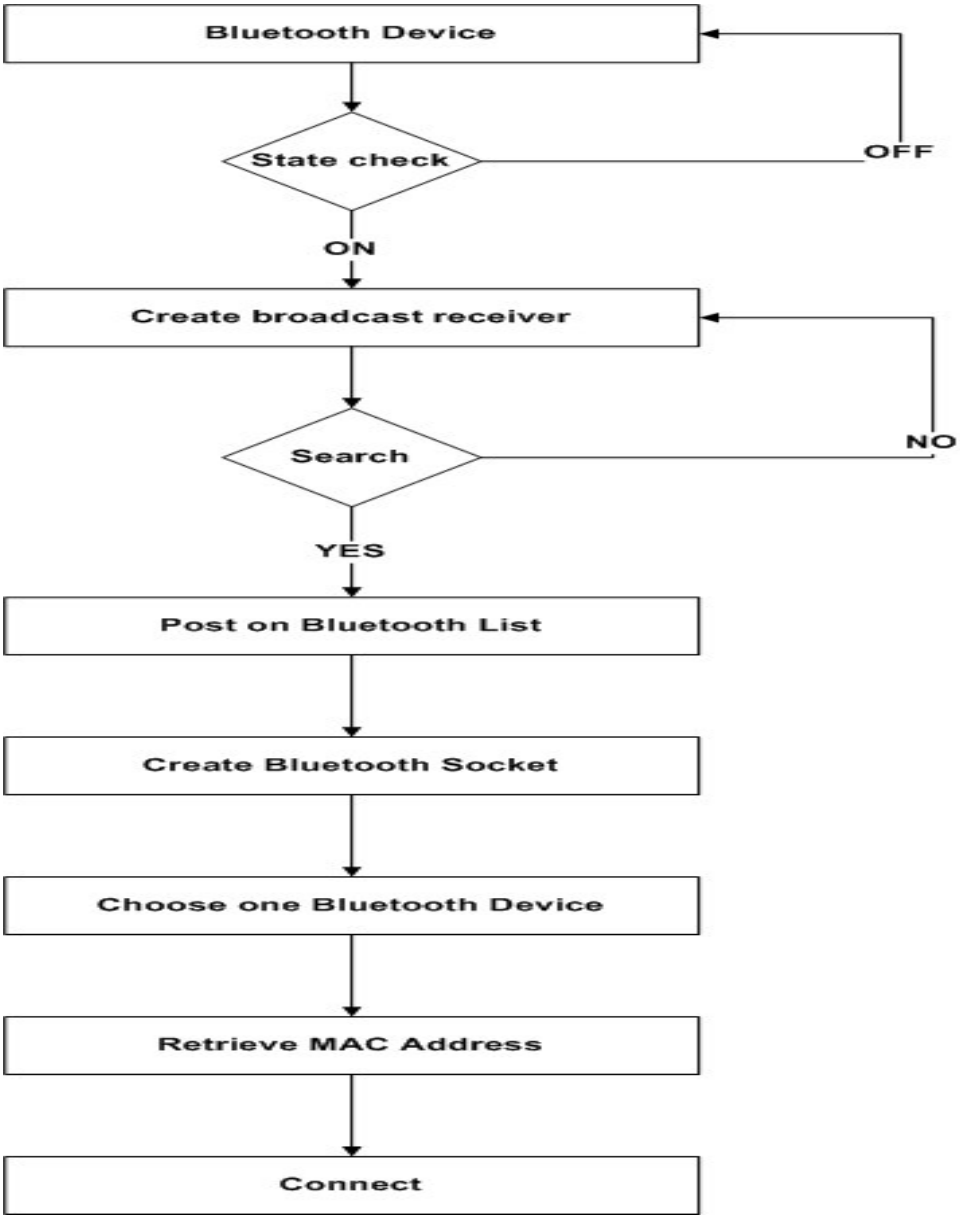

Figure 63 Workflow of Transfer Function

4.4.3 Synchronization Function

The synchronization function is used to achieve the goal of sharing the same contents in a file when such file is transferred from one mobile phone to another. The advantage of using the synchronization function is that the change of content could be maintained for different devices. To implement this feature, we create an XML file as a configure file on the cloud

server for a certain file read and write at the beginning. Figure 64 shows the work flow of the synchronization function.

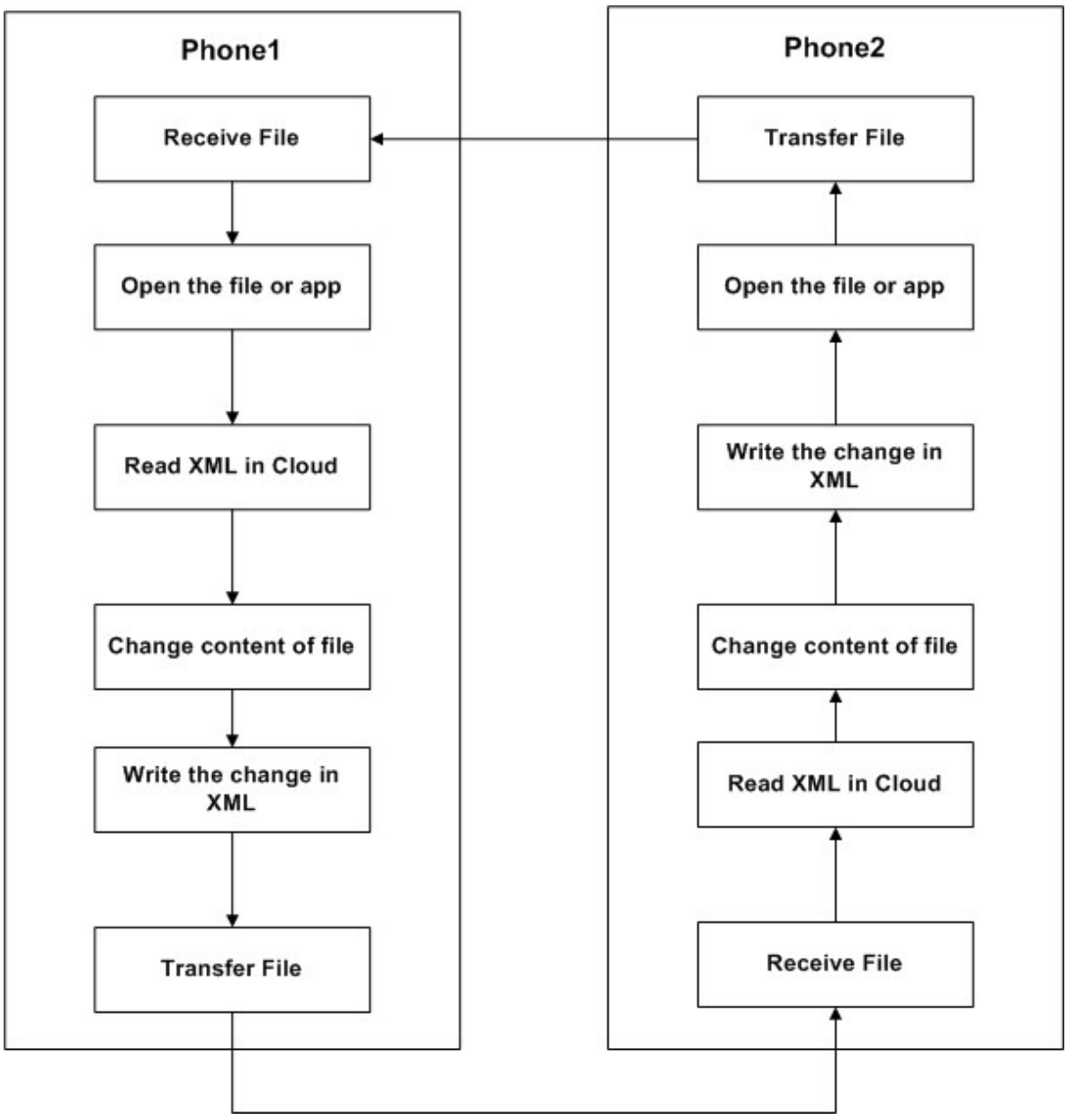

Figure 64 Workflow of Synchronization Function

CHAPTER 5

SCENARIO ILLUSTRATION

5.1 Overview

In this chapter, we will give an introduction of two different scenarios prototypes in our research and see how the activation component, the inactivation component and the communication component work in those two scenarios.

We will also use some real phone setting use cases and simulated multiple phone use cases to evaluate the performance of the SAMAF framework.

5.2 Case Study

In this section, we will give several case studies on the basic process of two scenarios. Case study describes the usage of operation, pre-condition, post-condition, extension point, service flow and several functional diagrams to illustrate main process of both scenarios in a detailed way.

5.2.1 Activation Component Case

(1) Functions are dynamically-generated based on context

Table 16 Use Case Analysis of the Generation Function in both Scenarios

Number	**Analysis Steps**	**Analysis Results**
1.1	Operational description	Describes how different functions are generated based on the context of different event.
1.2	Pre-condition	Internet is on, cloud server is on, context is coming.
1.3	Post-condition	Corresponding functions are generated and sequence has been adjusted

1.4	Extension Points	no
1.5	Basic Flow	a. User open main control application b. This use case will be activated when the context of certain event conditions comes c. Each context will be sent as a parameter to the cloud server d. In the cloud server, according to the parameter, web service 1 will modify the source code automatically e. Functions related to the certain condition will be placed in a proper order in one centralized application (E-1)
1.6	Alternative Flow	E-1: The sequence of functions cannot meet the requirement provided by the ontology based on the received disaster context

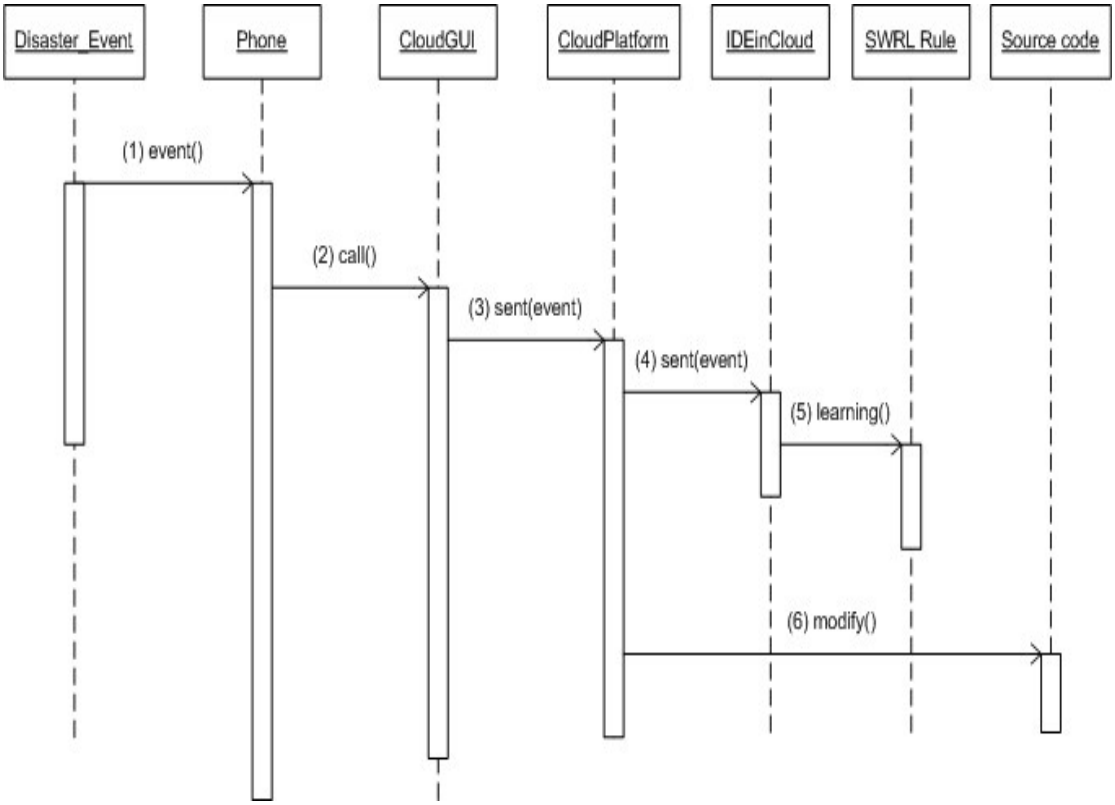

Figure 65 Sequence Diagram of Generation Function

(2) Project Compilation and Installation Package Generation

Table 17 Use Case Analysis of Compilation in both Scenarios

Number	Analysis Steps	Analysis Results
1.1	Operational description	Describes how to compile the project folder in cloud
1.2	Pre-condition	Internet is on, cloud server is on, generation is done
1.3	Post-condition	Corresponding apk installation package will be generated
1.4	Extension Points	no
1.5	Basic Flow	a. The generation of function is done b. Main control application receives the “done” message

		c. Main control application sends command for the compilation job to Cloud d. Web Service 1 in Cloud is responsible for compile the project to an installation package (E-1)
1.6	Alternative Flow	E-1: fail to generate apk installation package

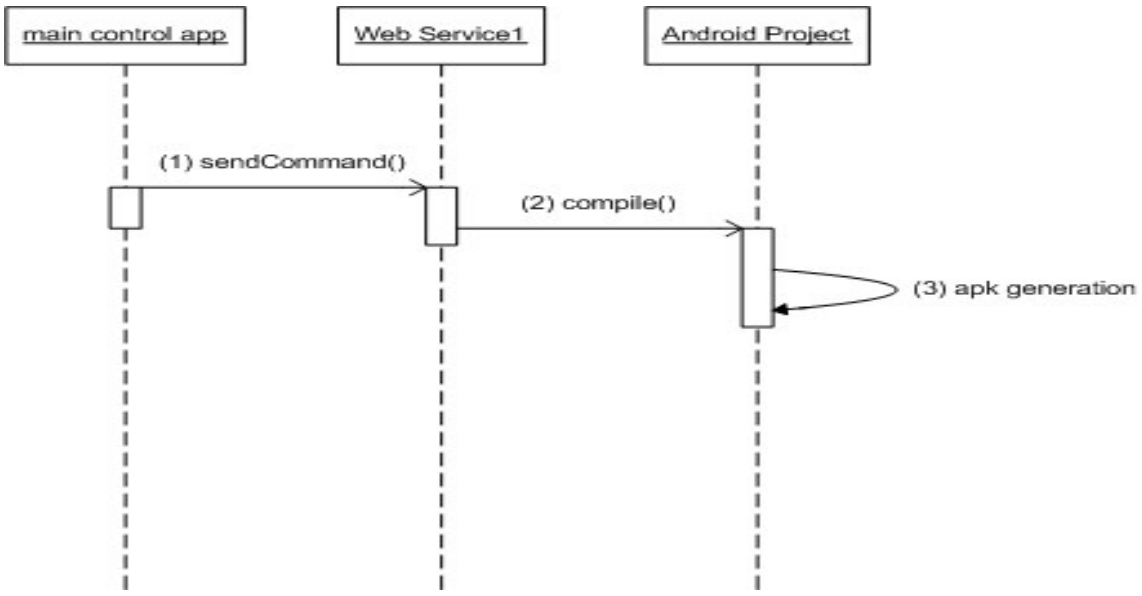

Figure 66 Sequence Diagram of Project Compilation

(3) Download apk Package from Cloud Server and Install

Table 18 Use Case Analysis of Download and Installation in both Scenarios

Number	**Analysis Steps**	**Analysis Results**
1.1	Operational description	Describes how main control application download apk from the cloud server.
1.2	Pre-condition	Internet is on, cloud server is on, apk exists
1.3	Post-condition	Apk file will be downloaded and opened
1.4	Extension Points	no

1.5	Basic Flow	a. User opens main control application b. This use case will be activated when apk is generated (E-1) c. Phone in the local will open apk file and install app (E-2)
1.6	Alternative Flow	E-1: no apk or wrong apk exists in cloud server E-2: mobile phone fails to open apk file and installation

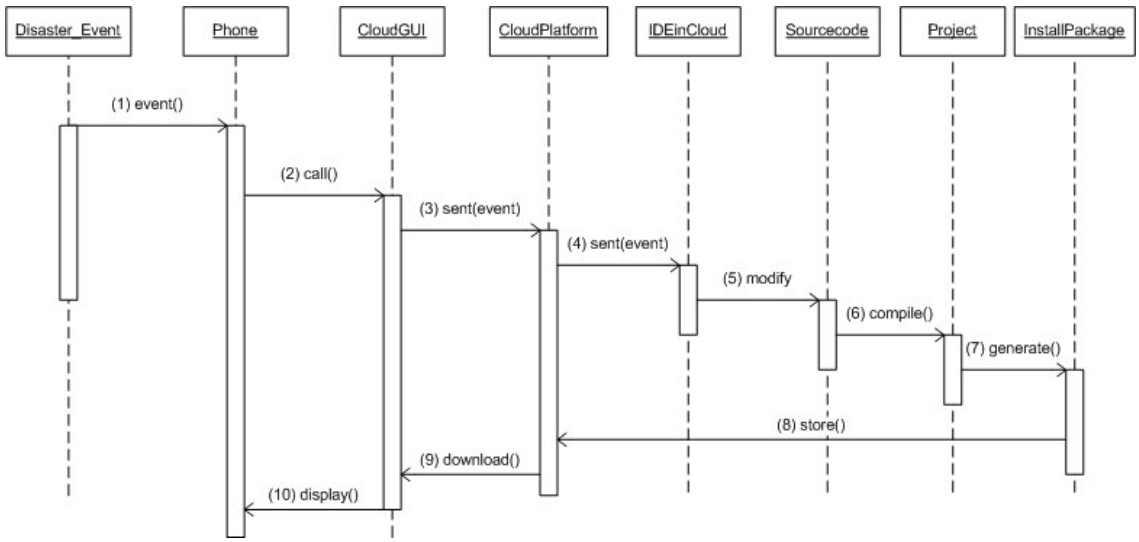

Figure 67 Sequence Diagram of Downloading Apk File and Installing Apk File

5.2.2 Inactivation Component Case

(1) Uninstall Application

Table 19 Use Case Analysis of Uninstallation of Application in both Scenarios

Number	**Analysis Steps**	**Analysis Results**
1.1	Operational description	Describes how the generated application will be uninstalled by the control of the main control application

1.2	Pre-condition	Generated application exists
1.3	Post-condition	Generated application is uninstalled
1.4	Extension Points	no
1.5	Basic Flow	a. User opens main control application b. This use case will be activated when the existence of the generated application has been tested and the timer is activated (E-1) (E-2)
1.6	Alternative Flow	E-1: no such application exists E-2: timer is not set properly

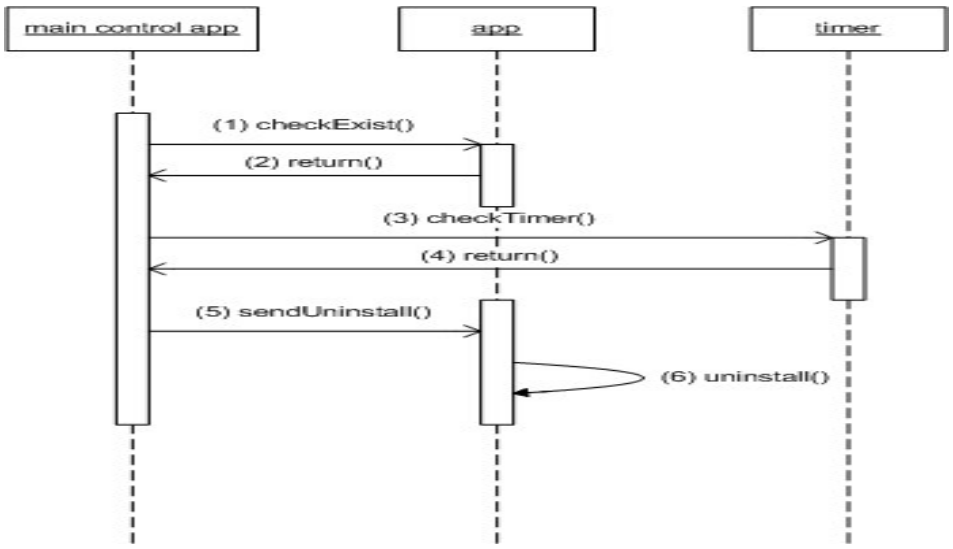

Figure 68 Sequence Diagram of Uninstallation of App

(2) Delete the Installation Package

Table 20 Use Case Analysis of Deletion & Installation Package

Number	**Analysis Steps**	**Analysis Results**
1.1	Operational description	Describes how apk installation file is deleted by the control of the main control application

1.2	Pre-condition	Apk installation package exists
1.3	Post-condition	Apk installation package is deleted
1.4	Extension Points	no
1.5	Basic Flow	a. User opens main control application b. This use case will be activated when apk installation package exists in the mobile phone disk and timer is activated (E-1) (E-2)
1.6	Alternative Flow	E-1: no such apk package exists E-2: timer is not set properly

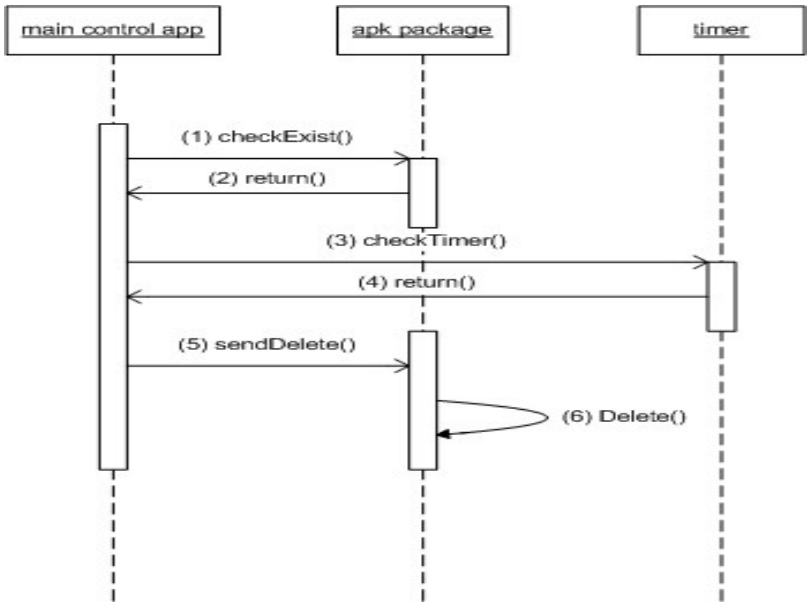

Figure 69 Sequence Diagram of Deletion of Apk

5.2.3 Communication Component Case

(1) File Transfer Process

Table 21 Use Case Analysis of File Transfer

Number	Analysis Steps	Analysis Results
1.1	Operational description	Describes how files are transferred through the communication tunnel
1.2	Pre-condition	File is ready, Bluetooth is on
1.3	Post-condition	File is transferred from one mobile phone to another
1.4	Extension Points	no
1.5	Basic Flow	a. User open transfer application b. This use case will be activated when users click transfer function in transfer application c. Choose the device to transfer (E-1)
1.6	Alternative Flow	E-1: no devices exist in the file transfer area

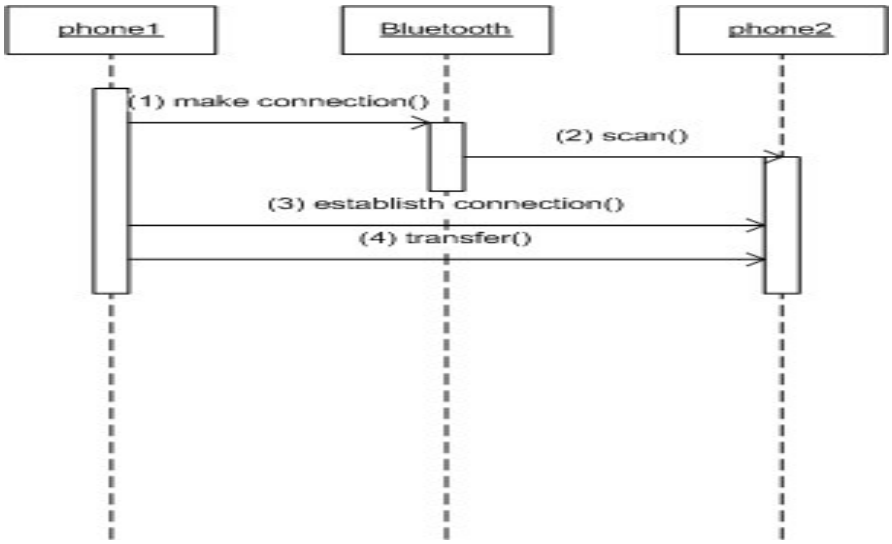

Figure 70 Sequence Diagram of File Transferation

(2) File Synchronization Process

Table 22 Use Case Analysis of Synchronization

Number	**Analysis Steps**	**Analysis Results**
1.1	Operational description	Describes how files are synchronized between phones
1.2	Pre-condition	Files are transferred from one phone to the other
1.3	Post-condition	Configure file is read to set up the initial parameters
1.4	Extension Points	no
1.5	Basic Flow	User open transfer application This use case will be activated when file has been transferred successfully (E-1)
1.6	Alternative Flow	E-1: file has not been transferred successfully

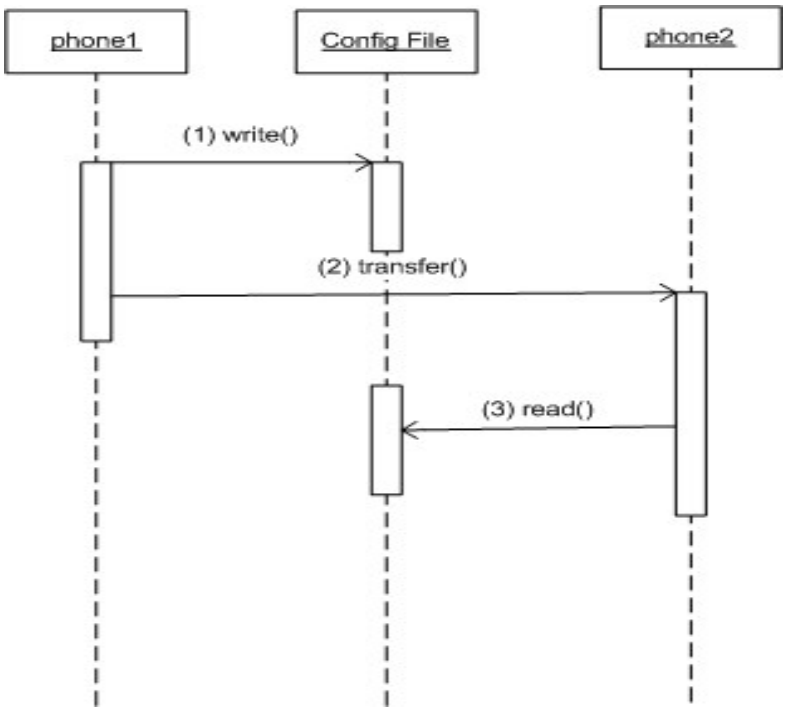

Figure 71 Sequence Diagram of Synchronization

5.3 Real Phone Settings Evaluation

5.3.1 Time Evaluation for Activation Model and Inactivation Model

(1) Time Evaluation for Activation Model – Generation Time

Generation time is a period of time that occurs from the time when contexts have been transferred to the cloud server to the time when the Android application installation package has been generated. We have conducted performance evaluation for generation time in three cases: for small installation file size, for medium installation file size, and for

large installation file size. Table 23 shows the relationship between different sizes of applications and generation time.

Table 23 Time Evaluation for Generation Process

APP Size (KB)	Generation Time (ms)
53	7312
342	7875
3236	9805

(2) Time Evaluation for Activation Model – Update Time

When the Android installation package already exists, and based on the identified situation, some new features will be added to the existing application; thus, the IDE model will compile the Android project again to involve new functions in the existing Android installation package. The time to re-compile the Android project and generated the new version of the Android installation package is named *update time*. This period of time includes the function change time, command transfer time, and compile time. Table 24 shows the relationship between different sizes of applications and update time.

Table 24 Time Evaluation for Update Process

APP Size (KB)	Update Time (ms)
53	6992
342	7094
3236	7368

(3) Time Evaluation for Activation Model – Installation Time

Here, installation time could be divided into two parts. The first part of time is downloading time. This is the time for downloading the Android application installation package from the cloud server to the smart phone. The second part of time is installation time. This is the time cost for installing the Android application installation package on the smart phone. Table 25 shows the relationship between different sizes of application and installation times.

Table 25 Time Evaluation for Installation Process

APP Size (KB)	**Installation Time (ms)**
53	2497
342	3759
3236	10369

(4) Time Evaluation for Inactivation Model – Uninstallation Time

Uninstallation time is the time period to uninstall an application on the smart phone when the users' situation could not be identified. Table 26 shows the relationship between different sizes of applications and uninstallation times

Table 26 Time Evaluation for Uninstallation Process

APP Size (KB)	**Uninstallation Time (ms)**
53	1040
342	1033
3236	1068

After all functions in both the activation model and inactivation model were tested, a histogram was made to show the relationship among them in an intuitive way. From the result, we can see that as time increased, the generation time grew slightly and the installation time grew significantly. Also, we can see that even the longest time was about 10 seconds. Because our framework is a cloud phone based communication, and the phone we used to do the evaluation is just a low end device, such a long time is reasonable, and therefore our approach is feasible.

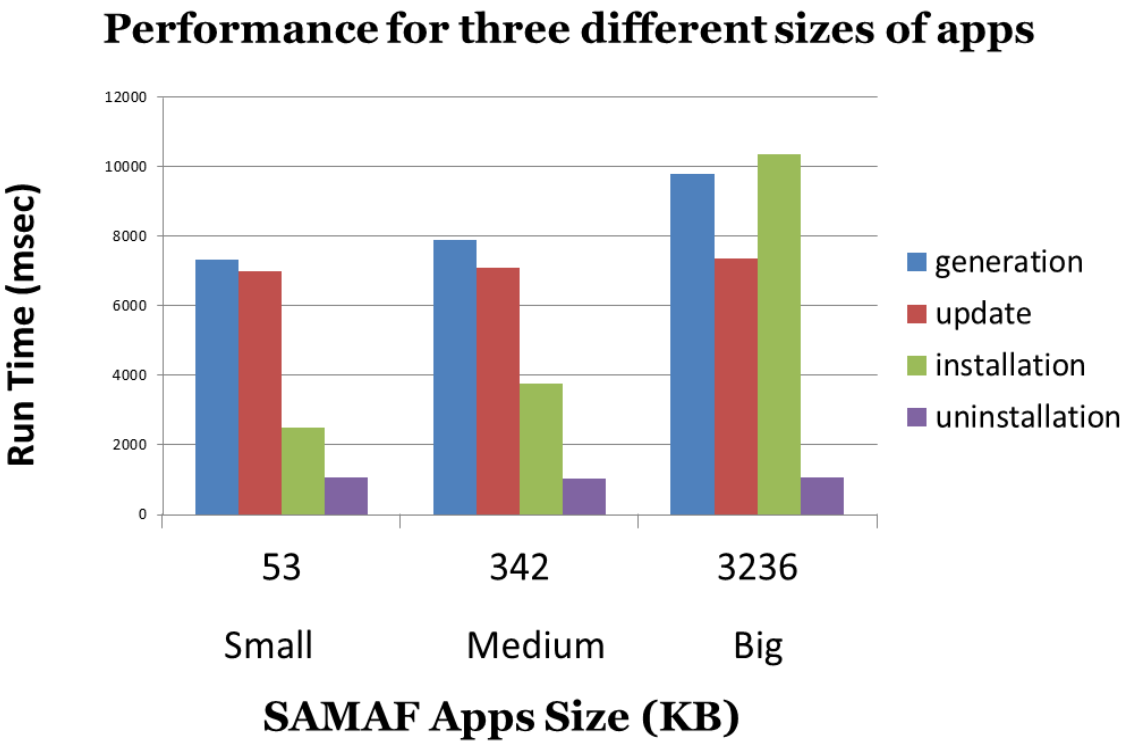

Figure 72 Relationship between App Size and Performance

5.3.2 Time Evaluation for Communication Model

(1) Time Evaluation for Transfer File

Transfer time is a period of time from the time when a file is sent by one mobile phone to the time when such file has been received successfully by another phone. We first make an

evaluation on the relationship among transfer distances between phones, file sizes and time consumption. The mobile phone we used for sending was a Motorola milestone android smart phone and the mobile phone for receiving was a NOKIA E71 smart phone. Table 27 shows the relationship among time, file size and distance between phones in file transfer process.

Table 27 Relationship among Time, File Size and Distance

Time (s)	**Distance between phones**			
	1m	3m	5m	7m
File Size(KB)				
53	3.5	3.8	3.3	3.2
342	5.5	5.7	5.3	5.2
3236	48.9	49.2	50.3	46.5

The relationship could be described in a histogram as figure 73 shows.

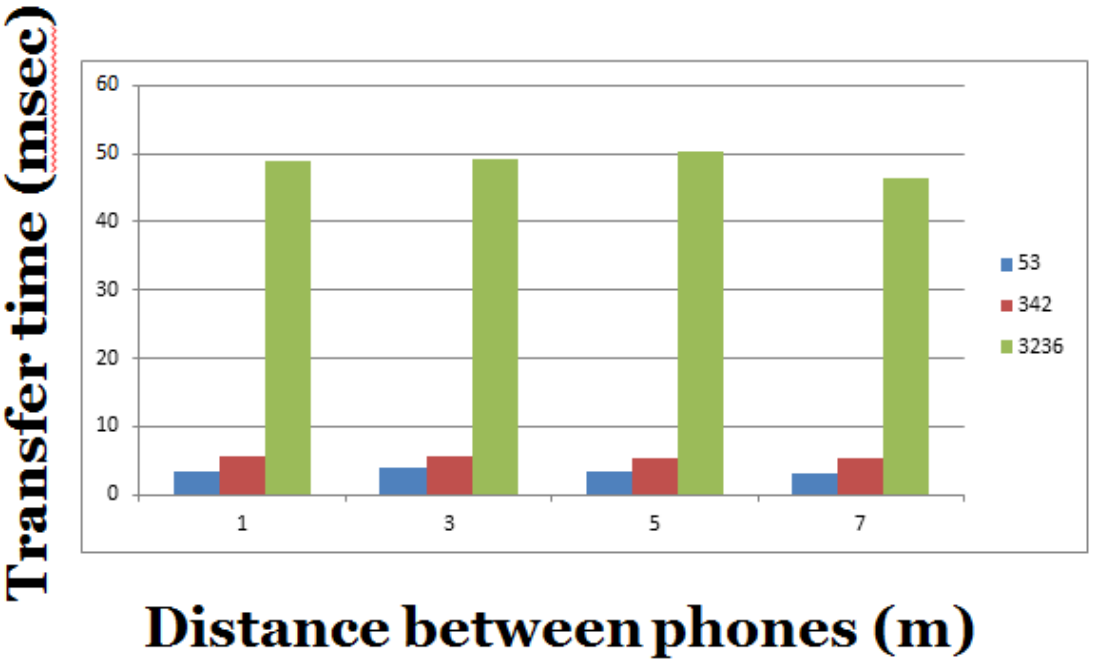

Figure 73 Relationship between Distance and Transfer time

(2) Time Evaluation for Synchronization – Read Configuration File

Synchronization reading time is a period of time from when the file is opened to the time when the contents in the configuration file have been read. Table 28 shows the time of reading the configuration file for synchronization.

Table 28 Reading Configuration Time Evaluation

File Size (KB)	Read Time (ms)
4	1.5
20	10.6
100	32.2

(3) Time Evaluation for Synchronization – Write Configuration File

Synchronization writing time is a period of time when changing the content of the file to the time when the configuration file has been changed based on the modification. Table 29 shows the time of writing the configuration file.

Table 29 Writing Configuration Time Evaluation

File Size (KB)	Write Time (ms)
4	2
20	12.8
100	37.5

The relationship could be described in a histogram as figure 74 shows.

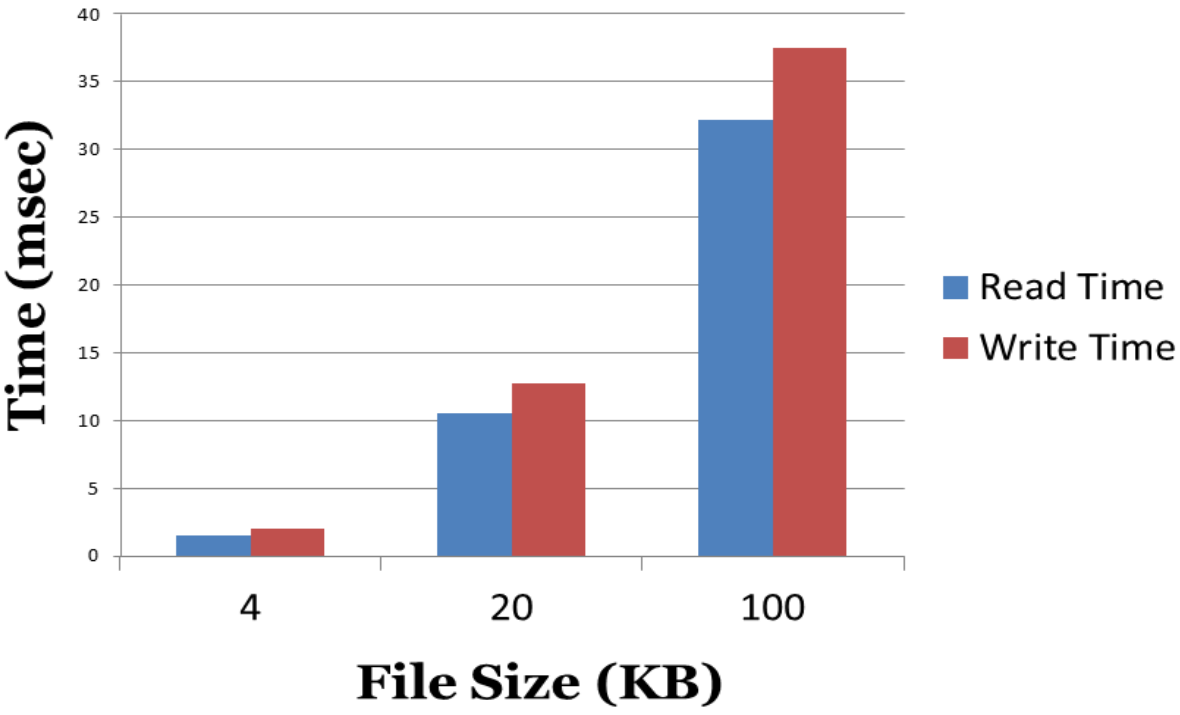

Figure 74 Relationship between R/W Time and File Size

5.3.3 Space Evaluation for Activation Model

We also evaluated the relationship between the Android application installation package size and the real application size after unpacking the installation package. The result is shown in Figure 75. From this result, we can imagine a situation in a long run prospective. For the largest size application, each has a size about 7 MB. If we have 1000 such applications installed in our smart phone, the total size could be almost 7 GB, and that is a very huge number for most low end smart phone devices. Therefore, our approach could provide an efficient way to manage the applications in a smart phone. For example, if a user is out of the situation, then the SAMAF reasoning model will recognize such a condition and remove the unrelated application from the smart phone. Meanwhile, the SAMAF reasoning model can also assemble some light weight applications to users based on the identified situations. In this way, the storage could be saved in smart phone

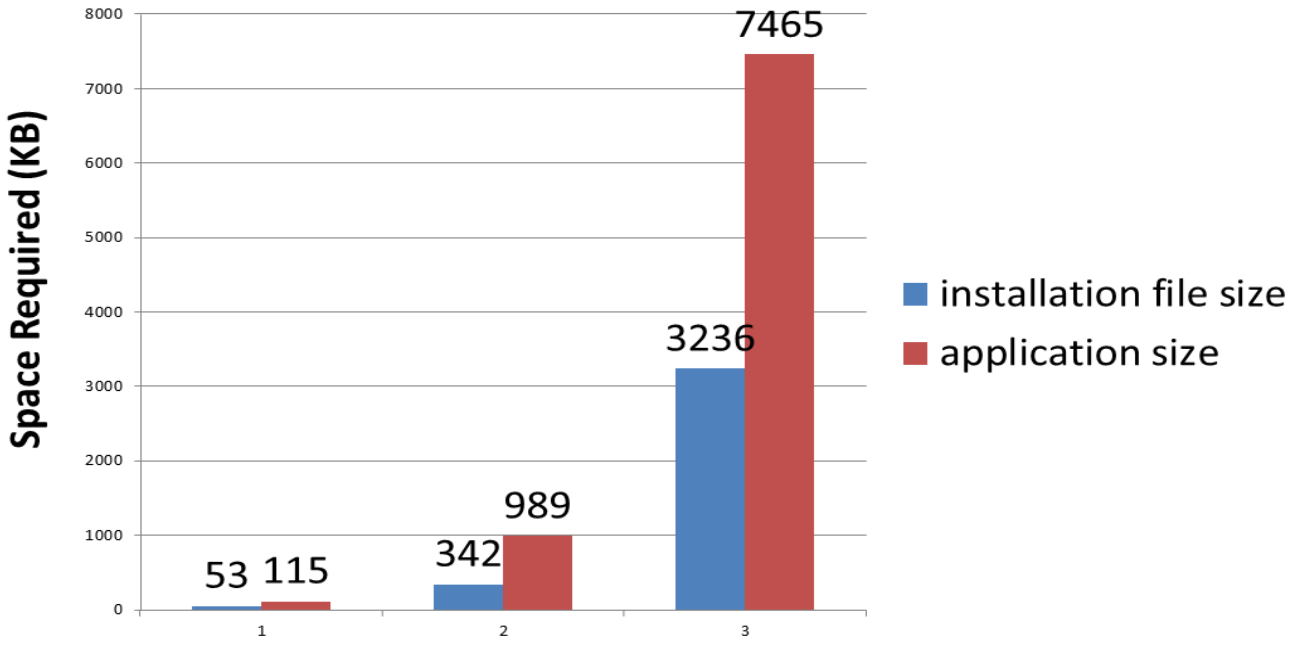

Figure 75 Relationship between Installation Package Size and Unpacked Application Size

5.4 Simulated Multiple Phones Evaluation

5.4.1 JADE Environment Overview

JADE (Java Agent DEvelopment Framework) is a software framework fully implemented in Java language. It simplifies the implementation of multi-agent systems through a middle-ware that complies with the FIPA specifications and through a set of graphical tools that supports the debugging and deployment phases. The agent platform can be distributed across machines (which not even need to share the same OS) and the configuration can be controlled via a remote GUI.(http://jade.tilab.com)

Four kinds of agents are used here to implement the evaluation scenario. Figure 76 shows the relationship between different agents. Up to 600 agents are used to evaluate the SAMAF model.

Three kinds of evaluations will be done: (1) Rule-based approach vs. Non rule-based approach in a distributed environment, (2) Centralized knowledge base approach vs. Distributed knowledge base approach, (3) Four situation aware adaptation cases.

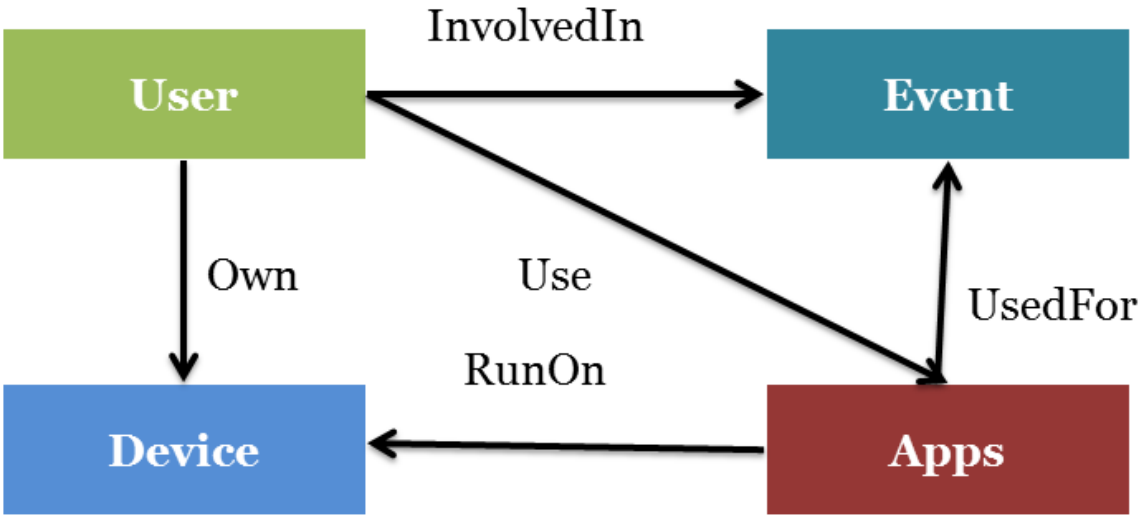

Figure 76 Relationship among Different Agents in JADE

To evaluate a SAMAF model in different manners, two types of evaluation environments have been proposed here. One is a distributed knowledge base environment, as Figure 77 shows. Another is a centralized knowledge base environment, as Figure 78 shows. From these two figures, we can see their basic architecture of them is the same. The only different between those two approaches is the operation on the knowledge base. For the distributed one, each phone agent in the environment has one knowledge base. For the centralized one, all the phone agents in the environment share one knowledge base.

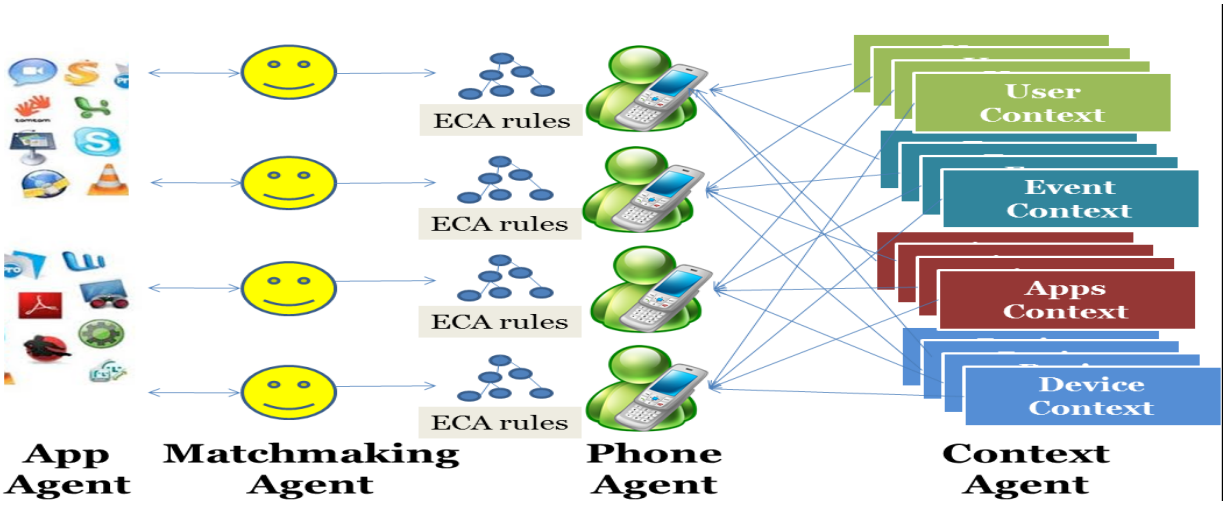

Figure 77 Distributed JADE Environment

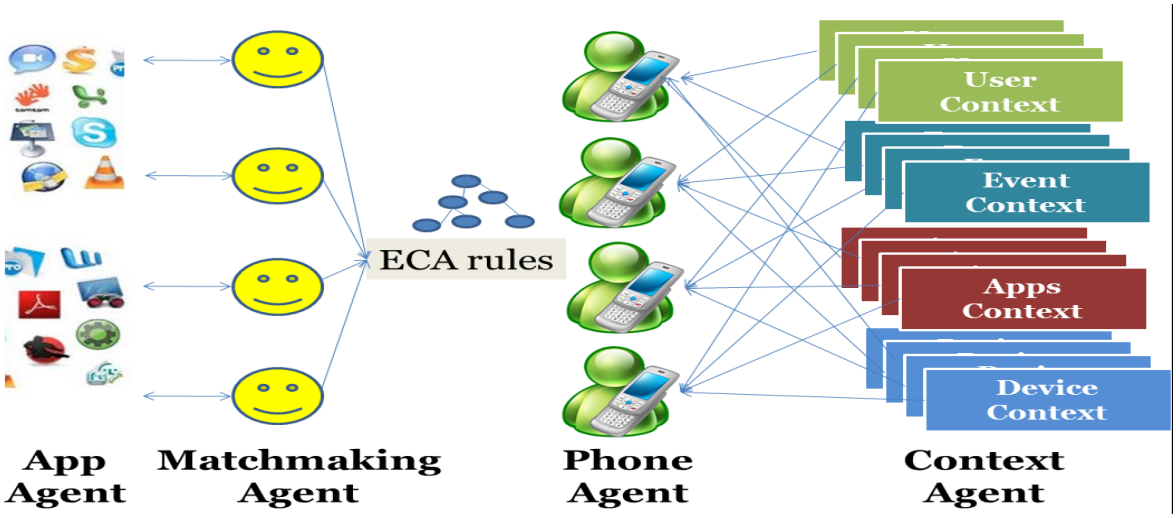

Figure 78 Centralized JADE Environment

In the evaluation, we will evaluate two kinds of time. One is adaptation time (the time for responding to a situation). Another is situation awareness time (time for recognition of an evolving situation). The time line for our evaluation is shown in Figure 79.

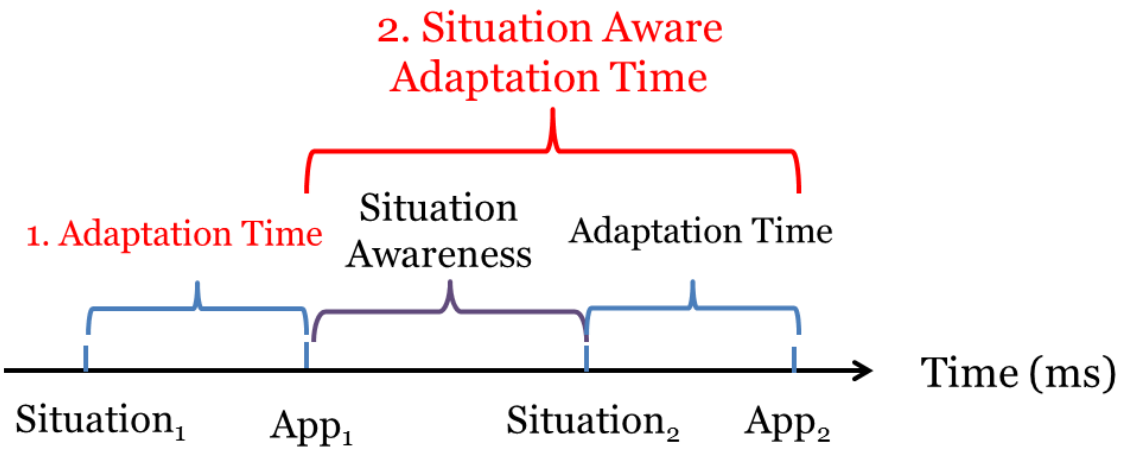

Figure 79 Time Line for JADE Evaluation

Since it is a simulated evaluation environment, we make some simple rules here to do reasoning work for different situations. Here we only consider the reasoning adaptation for the evolving phone locations, and based on different phone locations, SAMAF framework will do corresponding adaptation work for each phone.

We use SWRL language here to design the rule and use "Pellet reasoner" to do reasoning work based on rules. Table 30 shows those rules for the simulated evaluation work.

Table 30 Sample SWRL Rule for Evaluation

Rule1	$HasPhone(U,\ P) \cap PhoneOS(P,O) \cap Event(E)\ \cap$ *¬HasApp(O, Ai)*∩ $RunOn(Ai,O)\ \cap$ *InBoundary(Center(E), P)* ➔ *HasApp(O, Ai)*
Rule2	$HasPhone(U,\ P) \cap PhoneOS(P,O) \cap Event(E)\ \cap$ *¬HasApp(O, Ai)*∩ $RunOn(Ai,O)\ \cap$ *NearBoundary(Center(E), P)* ➔ *HasApp(O, Ai)*
Rule3	$HasPhone(U,\ P) \cap PhoneOS(P,O) \cap Event(E)\ \cap$ *¬HasApp(O, Ai)*∩ $RunOn(Ai,O)\ \cap$ *OutofBoundary(Center(E), P)* ➔ *HasApp(O, Ai)*

5.4.2 Evaluation for Rule-based and Non-Rule-based Approach

This evaluation was conducted in the distributed JADE environment. The number of phones we chose varied from 40 to 200. For each size number of phone, we compared the adaptation time and situation aware adaptation time between two approaches. Table 31 shows the adaptation time compared in this evaluation, Table 32 shows the situation aware adaptation time in this evaluation, Figure 80 and Figure 81show the comparison in an intuitive way.

From our results, we can easily observe that the rule-based approach takes longer than the non-rule-based approach in both adaptation time and situation aware adaptation time. However, the difference between those two approaches is not very large. From this aspect, we can see the SAMAF framework is feasible to solve the problem.

Table 31 Adaptation Time for Rule-base and Non-rule-based Approaches

Number of Phones	Rule-based adaptation time (msec)	Non Rule-based adaptation time (msec)
40	86	15
60	84	21
80	85	28
100	84	31
120	88	36
140	99	48
160	117	52
180	109	63

200	120	84

Table 32 Situation Aware Adaptation Time for Rule-base and Non-rule-based Approaches

Number of Phones	Rule-based adaptation time (msec)	Non Rule-based adaptation time (msec)
40	1046	988
60	1057	1000
80	1088	1008
100	1067	1016
120	1069	1023
140	1071	1052
160	1097	1043
180	1099	1091
200	1208	1145

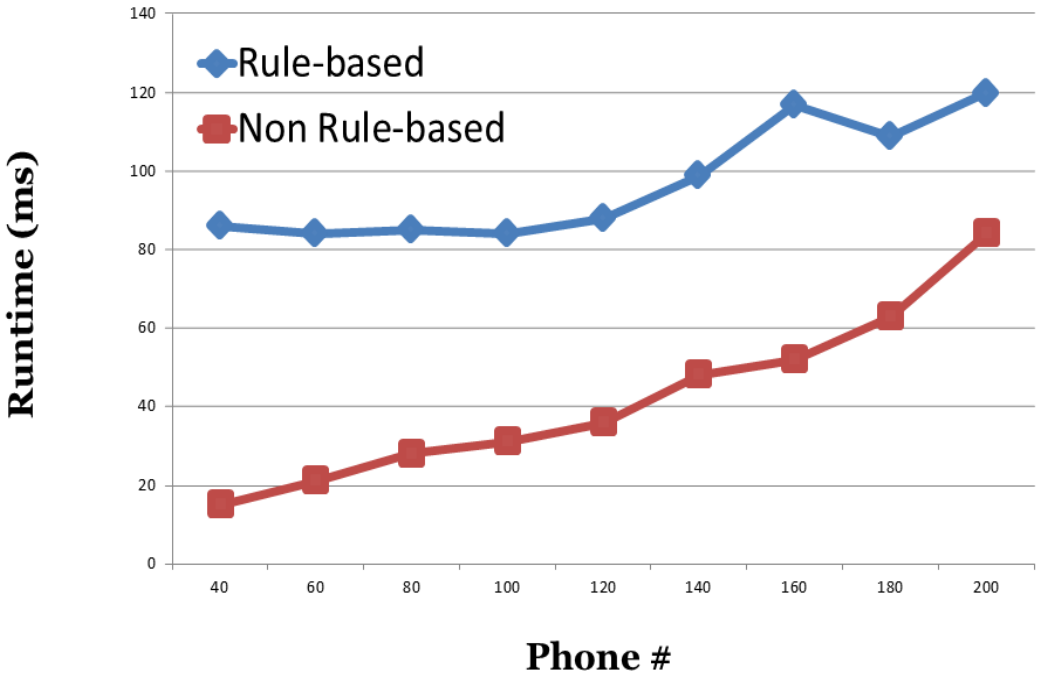

Figure 80 Adaptation Time for Rule-based and Non-rule-based Approach

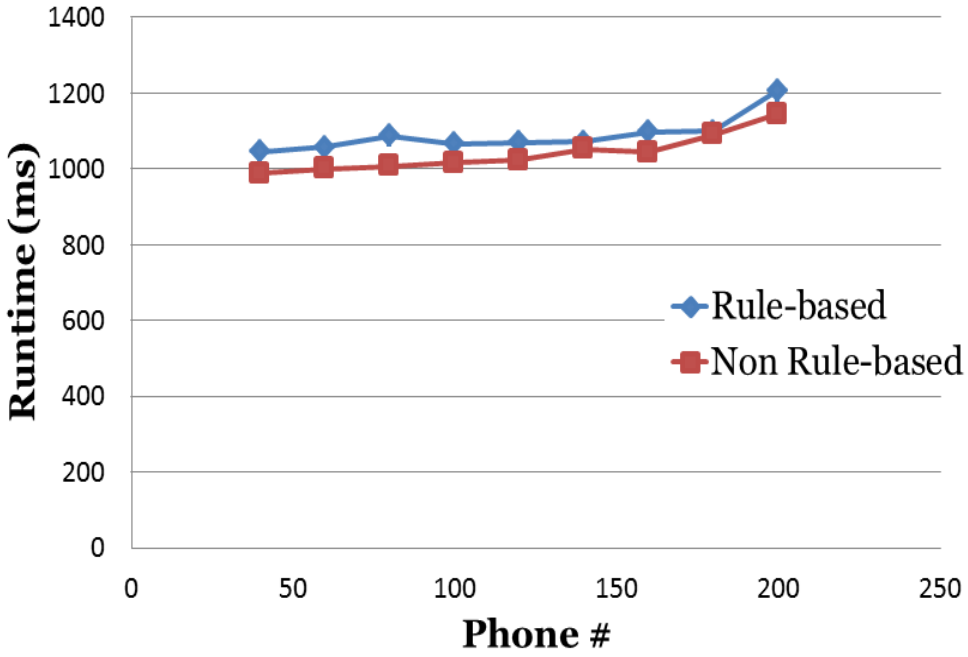

Figure 81 Situation Aware Adaptation Time for Rule-based and Non-rule-based Approach

5.3.3 Evaluation Between a Distributed Approach and a Centralized Approach

This evaluation was conducted on two kinds of system environments: a distributed one and a centralized one. We aimed at comparing adaptation time and situation aware adaptation time for both environments. The number of phones we chose in this case varied from 40 to 200. Table 33 and Table 34 show the results of this evaluation. Figures 82 and 83 show the result in an intuitive way.

From the result, we can observe that for both adaptation time and situation aware adaptation time, the distributed framework cost more time than the centralized one. However, both frameworks have their disadvantages. For the distributed knowledge base framework, each user will hold on to the knowledge base as it means much more time spent for updating the knowledge base for each one. While for the centralized knowledge base framework, because all the agents share one knowledge base, the rest of the agents will wait in a queue until previous ones finish their work; thus, this approach means much more time spent for the waiting time in a queue. Since both approaches have their weaknesses, we will try to combine them together to find a good joint point and make the best performance in our future work.

Table 33 Adaptation Time for Distributed and Centralized Approaches

Number of Phones	Distributed adaptation time (msec)	Centralized adaptation time (msec)
40	86	1470
60	84	1634
80	100	1582

100	84	1771
120	90	1843
140	99	2092
160	117	2133
180	109	2161
200	184	2272

Table 34 Situation Aware Adaptation Time for Distributed and Centralized Approaches

Number of Phones	Distributed situation aware adaptation time (msec)	Centralized situation ware adaptation time (msec)
40	1046	2392
60	1057	2565
80	1088	2425
100	1067	2629
120	1069	2836
140	1071	2998
160	1097	3024
180	1099	3072
200	1208	3174

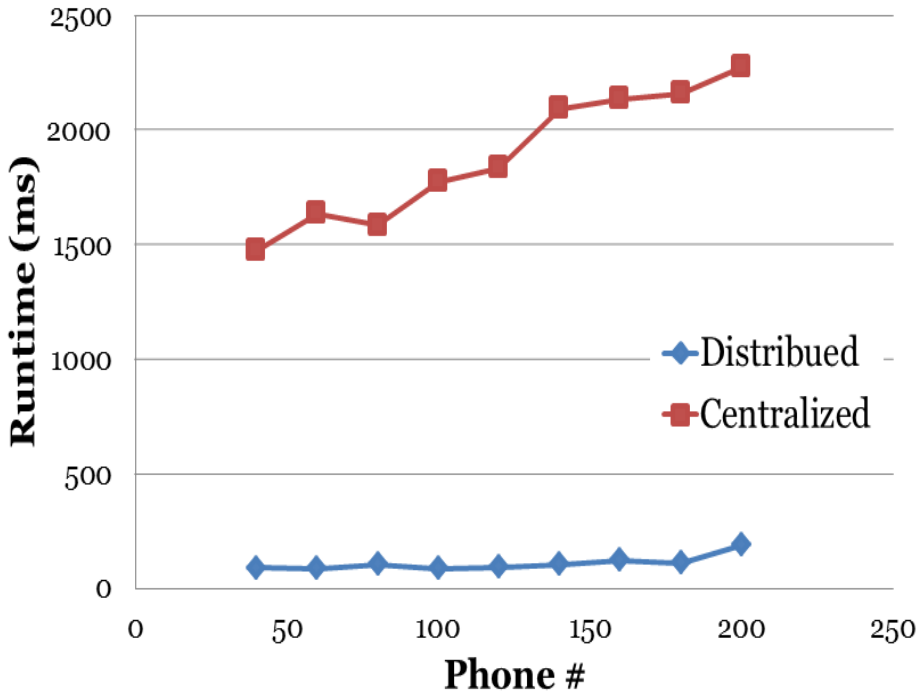

Figure 82 Relationship between Distributed and Centralized Framework on Adaptation Time

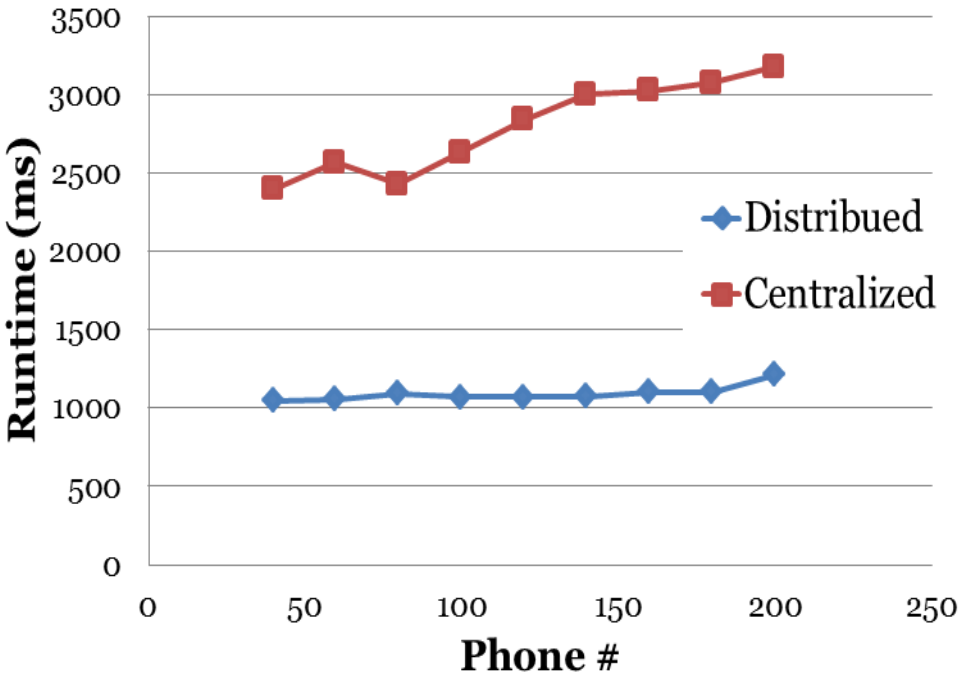

Figure 83 Relationship between Distributed and Centralized Framework on Situation Aware Adaptation Time

5.3.4 Evaluation for Four Situation Aware Adaptation Cases

In this section, we will evaluate the adaptation time and situation aware adaptation time in four different scenarios. We want to find some effect factors to the adaptation time and situation aware adaptation time.

We define the situation boundary we are interested, and such boundary could be divided into three cases: In situation boundary, near to situation boundary and out of situation boundary. Four steps to do the recognition of situation boundary: (1) recognize the event center. (2) recognize the situation boundary. (3) recognize 3 cases (in, near and out). (4) recognize the phones' context and determine their situation.

For Case 1, in initial time T1, there were many phones. Some were within the boundary, some were out of the boundary. For those phones within the boundary, we assigned applications to some of them, and left others without applications. Meanwhile, we held the phones in the boundary without letting them move out or move in. Then, in time T2, all the phones within the boundary will were assigned applications. We hope to evaluate the adaptation time and situation aware adaptation time in this scenario. Figure 84 shows such a scenario and Figure 85 show the result for this case.

We got a conclusion from case 1 that the greater number of phones in boundary in initial time T1, meant that the SAMAF framework assigned applications to less phones in the boundary until time T2, then less adaptation time and situation aware adaptation time cost.

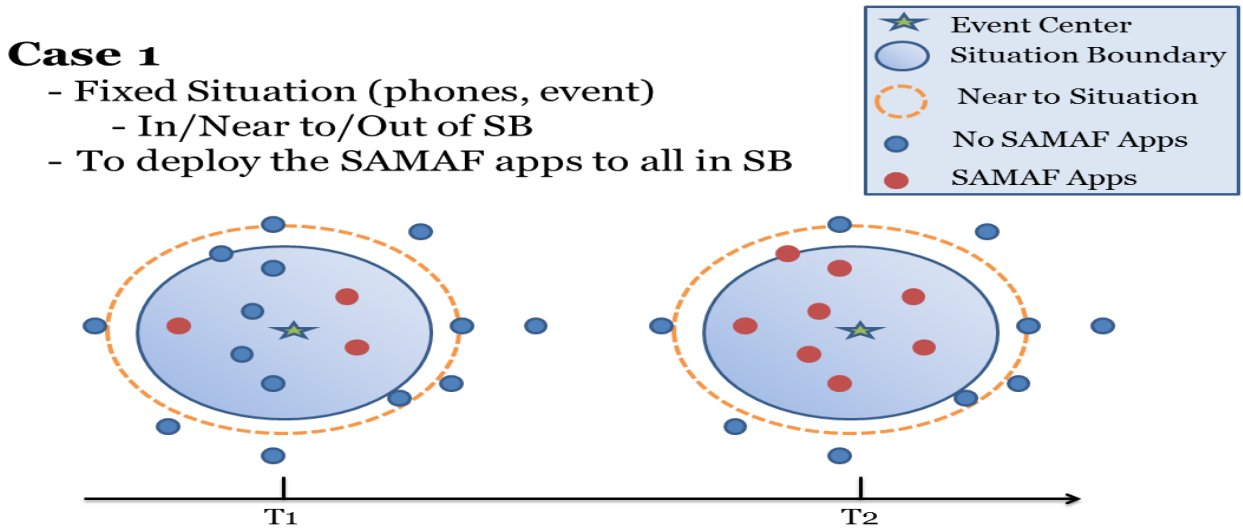

Figure 84 Case1: Fixed Phone and Event

Phone# in SB at T1	Phone# in SB with apps at T1	Phone# in SB with apps at T2	Adaptation Time (msec)
40	30	5	73
60	30	10	76
80	30	15	88
100	30	20	100

Figure 85 Result for Case 1

For Case 2, in initial time T1, there were many phones. Some were within the boundary, some were out of the boundary. For those phones within the boundary, we assigned applications to some of them, and left others without applications. Meanwhile, we let the phones in the boundary freely move in and move out. For those phones from boundary to out of boundary, the SAMAF framework will inactivated their applications, while for those

phones from out of boundary to boundary, the SAMAF framework assigned them applications. Then, in time T2, all the phones within the boundary were assigned applications and all the phones out of the boundary were inactivated applications. We hope to evaluate the adaptation time and situation aware adaptation time in this scenario. Figure 86 shows such a scenario and Figure 87 shows the result for this case.

We can got a conclusion from Case 2 that if more phones moved out of the boundary, more phones remained in boundary, but without applications, and more phones came from out of boundary to boundary then this means that the SAMAF framework did more work to inactivate applications for those out of boundary and assigned applications for both new coming phones and those phones that remained in the boundary, but without applications, then the adaptation time and situation aware adaptation time cost more.

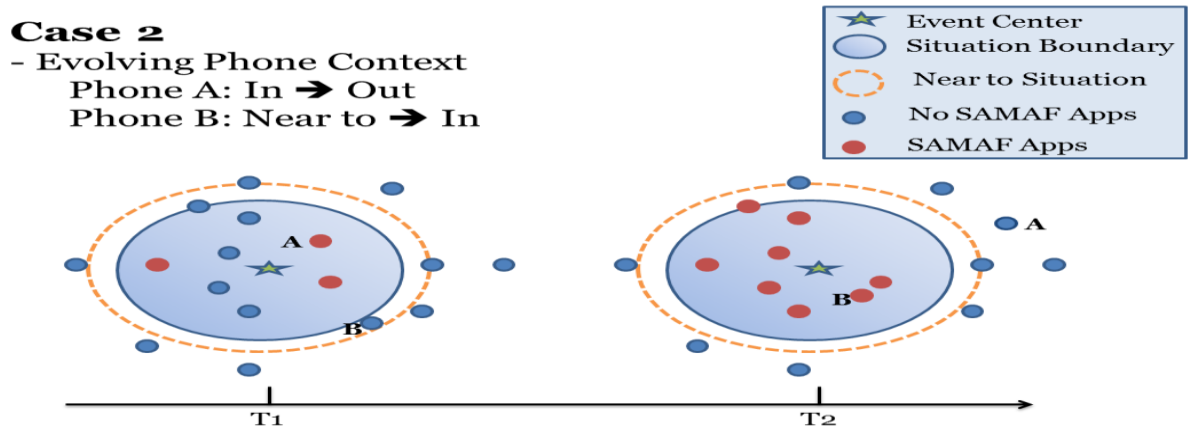

Figure 86 Case2: Evolving Phone Contexts

Phone# in SB at T1	Phone# move from SB to out of SB	Phone# without apps in SB at both T1 and T2	Phone# from outside to SB at T2	Situation aware adaptation Time (msec)
40	20	5	20	1050
60	30	10	30	1092
80	40	15	40	1133
100	50	20	50	1196

Figure 87 Result for Case 2

For Case 3, in initial time T1, there were many phones. Some were within the boundary, some were out of the boundary. For those phones within the boundary, we assigned applications to each of them. Meanwhile, we held the phones in the boundary and shrank the boundary. Then, in time T2, all the phones within the boundary were assigned applications. We hope to evaluate the adaptation time and situation aware adaptation time for inactivate applications for those phones out of the boundary in T2 in this scenario. Figure 88 shows such a scenario and Figure 89 shows the results for this case.

We got a conclusion from Case 3 that the greater number of phones in boundary in time T2, means that the SAMAF framework inactivated applications to less phones out of the boundary in time T2, then the less adaptation time and situation aware adaptation time will be cost. Also, because the boundary was shrank, meant the radius was changed, then it is necessary to change the rule, that is the reason why case 3 costs a little bit longer than others.

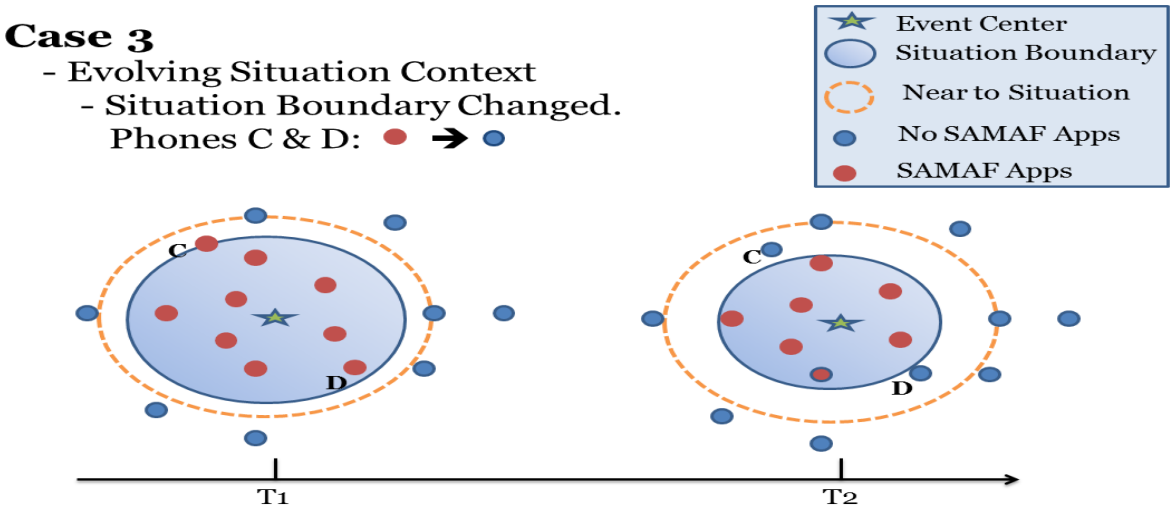

Figure 88 Case 3: Evolving Situation Context

Phone# in SB at T1	Phone# in SB at T2	Situation aware adaptation Time (msec)
40	30	1256
60	20	1386
80	10	1448
100	5	1489

Figure 89 Result for Case 3

For Case 4, in initial time T1, there were many phones. Some were within the boundary, some were out of the boundary. For those phones within the boundary, we assigned applications to each of them. Meanwhile, we held the phones in the boundary and moved the

event center to another place; this means that the boundary coverage area has changed. Then, in time T2, all the phones within the boundary were assigned applications. We hope to evaluate the adaptation time and situation aware adaptation time for inactivated applications for those phones out of the boundary in T2 in this scenario and assign applications to those phones that are coming to the boundary. Figure 90 shows such a scenario and Figure 91 shows the result for this case.

We came to the conclusion from Case 4 that if a greater number of phones go out of the boundary in time T2, and more phones come into the boundary from outside in time T2, it means that the SAMAF framework inactivated applications and assigned applications to more phones. Therefore, for those phones, then the more adaptation time and situation aware adaptation time will cost.

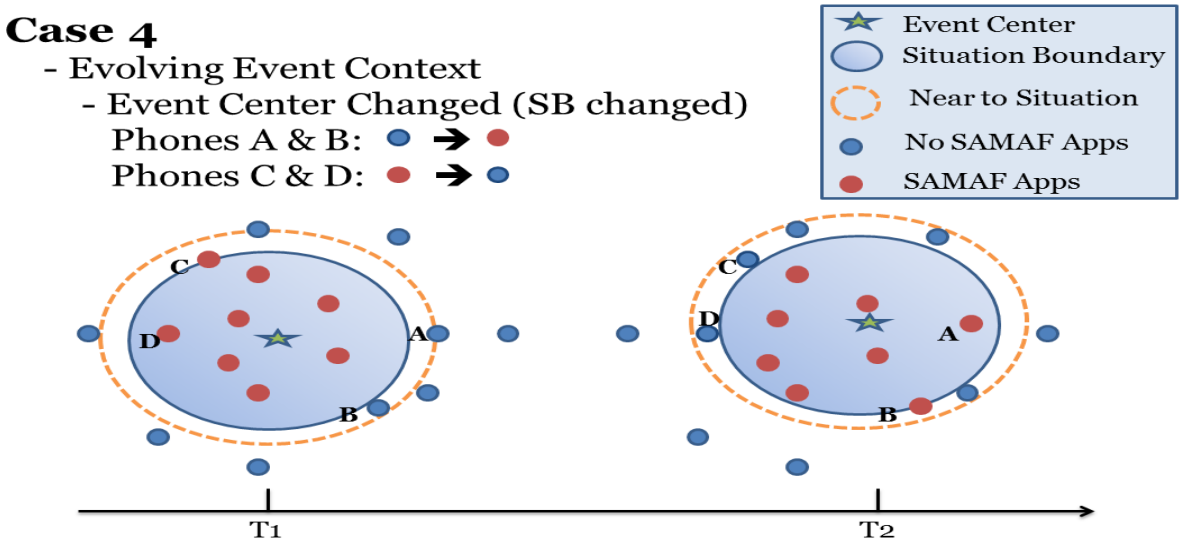

Figure 90 Case 4: Evolving Event Context

phone# in SB at T1	Phone# move from SB to out of SB	Phone# from outside to SB at T2	Situation aware adaptation Time (msec)
40	30	50	1088
60	20	40	1057
80	10	30	1046
100	5	20	958

Figure 91 Result for Case 4

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 Summary

In this thesis, a situation awareness mobile application framework is proposed. The proposed framework could be used as an efficient method for retrieving context from the user side and doing intelligent reasoning based on defined rules. A phone based prototype developed as a main control application works together with a cloud server for managing and coordinating applications to achieve the goal of intelligent action reasoned by rules based on the received users or event context. The experimental results prove that the SAMAF framework can offer an immersive experience to mobile phone users and can help them deal with the emergency situations or daily business in an efficient and effective way.

6.2 Future Work

The SAMAF framework leaves a lot for room for the future, as some of the functions and features that might be extended are

1. Using the various sensors installed in mobile phones to gather the context data from either inside or outside the environment of a mobile phone instead of using string variables to simulate the context generation process.

2. Explore machine learning and optimization approach for mobile adaptations dealing with evolving situations.

3. Study and analyze complexity and comprehensive rules to handle situations of multiple events.

4. Development of a 3D virtual world for visualization and simulation of situations, events, and dynamic adaptations of mobile phone apps.

REFERENCES

[1] Nicholas D. Lane, Emiliano Miluzzo, Hong Lu, Daniel Peebles, Tanzeem Choudhury, and Andrew T. Campbell. *A Survey of Mobile Phone Sensing*. Communications Magazine, IEEE, 2010, 48(9) , pp. 140 – 150

[2] Bill N.Schilit; Marvin M.Theimer. *Disseminating Active Map Information to Mobile Hosts*. IEEE Network, 1994, 8(5), pp 22-32.

[3] Anind K. Dey. *Understanding and Using Context*. Personal Ubiquitous Comput. 5, 2001, pp 4-7.

[4] Xiao Hang Wang, Da Qing Zhang, Tao Gu, Hung Keng Pung. *Ontology based Context Modeling and Reasoning using OWL*. PERCOMW '04 Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops, 2004, pp18

[5] Xiao Hang Wang, Da Qing Zhang, Tao Gu, Hung Keng Pung. *An Ontology-based Context Model in Intelligent Environments*. In proceedings of communication networks and distributed systems modeling and simulation conference

http://www-public.it-sudparis.eu/~zhang_da/pub/Ontology-2004-2.pdf

[6] Karen Henricksen, Steven Livingstone and Jadwiga Indulska. *Towards a Hybrid Approach to Context Modeling, Reasoning and Interoperation.* Excellence in Research Australia (ERA) – Collection

http://henricksen.id.au/publications/UbiCompWorkshop04.pdf

[7] Zakir Laliwala; Vikram Sorathia; Sanjay, Chaudhary. *Semantic and Rule Based Event-driven Services-Oriented Agricultural Recommendation System*. Distributed Computing Systems Workshops, 2006, pp24

[8] Thomas Hornung, Agnes Koschmider, Andreas Oberweis. *Rule-based Auto Completion of Business Process Models*. n CAiSE Forum 2007. Proceedings at the 19th Conference on Advanced Information Systems Engineering (CAiSE)

http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-247/FORUM_13.pdf

[9] Wolfgang May, Jose Julio Alferes, Ricardo Amador. *An Ontology- and Resources-Based Approach to Evolution and Reactivity in the Semantic Web*. OTM'05 Proceedings of the 2005 OTM Confederated international conference on On the Move to Meaningful Internet Systems: CoopIS, COA, and ODBASE - Volume Part II, 2005, pp 1553-1570

[10] Randall Perrey; Mark Lycett. *Service-oriented Architecture*. Applications and the Internet Workshops, 2003. Proceedings. 2003, pp 116-119

[11] Hiroshi Ishiguro. *Android Science Springer Tracts in Advanced Robotics*. Volume 28/2007, 2007, pp 118-127

[12] Tobias Warden, Robert Porzel, Jan D. Gehrke, Otthein Herzog, Hagen Langer, Rainer Malaka. *Towards Ontology-based Multiagent Simulations: The Plasma Approach*. ECMS June1-4, 2010, pp 50-56

[13]Yugyung Lee, Nikhilesh Katakam, Deendayal Dinakarpandian, Dennis Owens, Sachin Mathur, Saranya Krishnamoorthy, & John Wubbenhorst. *An Intelligent Online System for*

Enhanced Recruitment of Patients for Clinical Research. Missouri Regional Life Sciences Summit, 2010, Kansas City, MO.

[14]Saranya Krishnamoorthy, Yugyung Lee & Deendayal Dinakarpandian. *Data Driven Derivation of Canonical Eligibility Criteria for Clinical Trials*. CSHALS--2011 Boston, MA

[15]Yugyung Lee, Deendayal Dinakarpandian, Nikhilesh Katakam, Dennis Owens. *MindTrial: An Intelligent System for Clinical Trials*. American Medical Informatics Association (AMIA). 2010 Annual Symposium

VITA

Feichen Shen was born on Nov 2nd, 1987, in Nanjing, China. He studied as an undergraduate student in Computer Science in Nanjing University Jingling College and received the Bachelor degree grant by Nanjing University, China. After his under graduation he moved to the United States for his higher education. He is currently pursuing his Masters in Computer Science at the University of Missouri – Kansas City. He will continue his PhD study in University of Missouri – Kansas City.

He worked as a Graduate Research Assistant in School of Computing and Engineering of University of Missouri – Kansas City from June 2011. Through the period of research, he worked with Dr.Yugyung Lee on the project titled MindTrial funded by National Institute of Mental Health (NIMH) and the project about Situation Awareness Mobile Application Framework. He also worked as Intern-Biostats (PHD) in Department of Health Sciences Research at Mayo Clinic. Feichen Shen also co-authored one paper "Semantic Search Engine for Clinical Trials" and presented a poster at SHARPn Summit 2012 that is held by Mayo Clinic.

Posters:

1. Yugyung Lee, Saranya Krishnamoorthy, Feichen Shen, Sourav Jana, Deendayal Dinakarpandian, Dennis Owens, Semantic Search Engine for Clinical Trials, SHARPn Summit 2012, Rochester, MN.