NORWEGIAN UNIVERSITY OF SCIENCE AND TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY, MATHEMATICS AND ELECTRICAL ENGINEERING

PROJECT ASSIGNMENT

Student's name:	Urooj Fatima
Course:	TTM4521
Title:	**Intelligent Travel Guide – Service Composition by End-User**
Description:	The goal of this assignment is to investigate possibilities for end-user service composition using a composed service for the intelligent travel guide as case study. This kind of composed service is aimed for the frequent business or leisure traveler in a typical city environment. In this case the city is Trondheim downtown and the communication infrastructure will be the Wireless Trondheim network as well as the mobile telephony networks. Devices that will be used will range from laptop to various kinds of handheld gadgets. The main position or location enabler will be the GPS system. The task will consist of the following subtasks: 1. Work out and describe the composition scenarios of the service possibly with several situations (involved actors, interactions, etc.) and different settings (location, multiplicity, preferences, etc.). It is required to specify these situations in such a way that traveler, travelling-agency, and someone-on-traveler's-behalf will be the composer. 2. Select and define a set of services for the composition. 3. Define the appropriate user interface for executing and composing these services.
Deadline:	11.01.2010
Submission date:	12.01.2010
Department:	**Department of Telematics**
Supervisor:	Mazen Shiaa

Trondheim, ……2010

Rolv Braek
Professor

Abstract

Service composition is an emerging field in the world of ICT. This project work investigates possibilities for end-user service composition using a composed service for the intelligent travel guide as case study.

This project has followed a scenario based approach to identify the possible users and contexts for intelligent travel guide. Usage scenarios are brainstormed that have resulted in the identification of new self-contained services. These services then served as bricks for building composition scenarios. These scenarios are exploited for the design of new services that are to be composed by the end-users. An existing end-user programming environment 'EasyComposer' is upgraded by implementing the new services in it. 'EasyComposer' has served successfully to demonstrate the composition scenarios. XML representation has been worked out both for services and service compositions to realize them.

Besides, 'Doctor's Appointment' scenario is demonstrated in EasyComposer to strengthen the investigation of end-user service composition. A comparison of 'EasyComposer' with 'Arctis' is exercised that helped in the analysis of further improvements in EasyComposer.

A number of possibilities has been investigated for end-user service composition, but still there is a long way to go. Most significantly, the practical issues needs to be addressed specially related to the provision of freedom and control to the end-user and at the same time ensuring the composition to be logical. One solution is proposed in this project work.

Preface

This project report is written at the Department of Telematics in the Norwegian University of Science and Technology (NTNU) during the Autumn Semester, 2009. This specialization project is mandatory for the students of M.Sc. in Telematics, in 3rd Semester.

The theme of this project work was defined by UbiCompForAll - Ubiquitous service Composition For All users. UbiCompForAll is a research project founded by the Norwegian Research Council. The project involves SINTEF, NTNU, Gintel, Tellu and Wireless Trondheim.

A part of this project report is based on the work of former Master student Jens Einar. He created an interface 'EasyComposer' for end-user service composition. After working out for the set of services, I started with the same interface created by him and upgraded it by implementing these new services in it. Jens Einar also defined the services and service compositions using XML representation. I built XML representation of the services and service compositions on the work done by him and made some improvements.

I would like to thank my Supervisor, Mazen Malik Shiaa, and Professor Rolv Braek, for giving me good and comprehensive advices and guiding me through the tricky tasks. Special thanks to them for understanding my stage of illness during the project work.

I am thankful to Jens Einar for his support and cooperation throughout my project work.

I would also thank my Mother, Salma Nasim and Sister, Nida Batool for being there always for me to keep my moral high, regardless of being thousands of miles away from me. Who supported me and encouraged me whenever I felt alone.

Urooj Fatima
January 11, 2010

Contents

List of Figures

List of Tables

Acronyms

BPEL	Business Process Execution Language
FLC	Functional Level Composition
PBE	Programming By Example
UbiCompForAll	Ubiquitous service Composition For All users
VAS	Value Added Services
VPL	Visual Programming Language

Chapter 1: Introduction

This chapter gives an overview of the report and the problem to be solved. It explains the motivation behind it by answering the question; why one should spend time to read it? Outline of the report will guide you how to read this report that will save your time.

1.1 Motivation

Nowadays, every one talks about intelligent environment. We are using computational devices and systems in our daily life. They influence almost all parts of our lives. The Value Added Services (VAS) is one of the most common used terminologies nowadays in the world of Telecom. Every one is familiar with the term 'Services', but what are 'Value Added Services'. In simple words, it is adding value to the services by combining the existing standard services[1]. Next question that comes into mind is; who adds value to the services? The answer should be 'The Service Provider'. But, what if I say, "We can add value to the services by combining the existing standard services with our own preferences". Yes, this is absolutely true and 'End-user service composition' is the answer.

End-user service composition explains similar concept discussed above. The number of end-users of the services is larger than those who develop these services and professionally use them. Therefore, we should focus on what end-user needs. The services, which are already there to assist the end-users, may not fulfill all of their needs. Service providers have solutions with a general set of services. Apart from general requirements, every end-user has his or her personal preferences as well. End-user service composition provides an easy solution for this dilemma. It facilitates you to have your own personal service which can assist in daily life at your home, at your office, in your city and even outside the country, without any special assistance from service providers.

1.2 Problem to be solved

The goal of this assignment is to investigate possibilities for end-user service composition using a composed service for the intelligent travel guide as case study. This goal is to be met

[1] Value Added Services can be standalone services as well.

by describing composition scenarios and defining a set of services for the composition. The tasks also include definition of user interface for executing these services but due to time constraints it was not possible to execute these services. Gintel has an interface (EasyComposer) for composing Telecom related services for end-users. It is decided to define new services, implement some of these in EasyComposer and upgrade it. These new services, together with the existing services of EasyComposer, will be used to describe the composition scenarios not only for intelligent travel guide but also for other scenarios including 'Doctor's Appointment Scenario'.

1.3 Project report outline

The project report is structured as follows:

Chapter 1: Introduction gives an overview of the report and the problem to be solved.
Chapter 2: Background describes the theoretical background regarding the concepts and technologies relevant to this project.
Chapter 3: Methodology narrows down the problem to a set of tasks, explains the approach followed and methodology used from the point of origin to the point of completion in this project work.
Chapter 4: End-user service composition presents the work that has been executed following the methodology.
Chapter 5: Practical Issues discusses the practical issues related to end-user service composition.
Chapter 6: Conclusion concludes the thesis by summing up the results, discussing them and proposing the future work.
Appendix: XML representation tables of services and service compositions are placed here because they are too complex to be written in the main part.

Chapter 2: Background

This chapter introduces the theoretical background regarding the concepts relevant to this project. As said by [Jens 2009], background information directly related to end-user service composition is not easy to be described because the existing service composition technologies are aimed at supporting professional developers. These technologies are too professional to the end-users to easily grasp and apply them.

Concepts related to ubiquitous computing and service composition combined with end-user programming that are mentioned in the goals of UbiCompForAll project, serve a reasonable theoretical background for this project work. Similar format has also been followed by [Jens 2009] [Eirik 2009].

2.1 Ubiquitous Computing

Stefan Poslad defined ubiquitous computing comprehensively as [Poslad 2009]:

"The term 'ubiquitous', meaning appearing or existing everywhere, combined with computing to form the term Ubiquitous Computing (UbiComp) is used to describe ICT (Information and Communication Technology) systems that enable information and tasks to be made available everywhere, and to support intuitive human usage, appearing invisible to the user".

The synonymous term for this concept is 'calm or disappearing computer'. Its understanding can have several directions [Poslad 2009]:

- It can mean that programmable computers as we know them today are replaced by something else, e.g., human brain implants, that they are no longer physically visible.
- It can mean that computers are present but they are hidden, e.g., they are implants or miniature systems.
- The focus of the disappearing computer can mean that computers are not really hidden; they are visible but are not noticeable as they form part of the peripheral senses. They are not noticeable because of the effective use of implicit human–computer interaction.

These alternate perceptions of meaning will depend in part on the target audience. For some groups of people, ubiquitous computing is already here. Applications and technologies, such as mobile phones, email and chat messaging systems, are considered as a necessity by some people in assisting them on daily basis.

The most significant focus of ubiquitous computing is to support intuitive human usage. The complexity is the main obstruction for end-users when using current professional composition technologies [Liu 2007]. The idea of UbiCompForAll is about providing support to ordinary non-IT professional end users so they can easily compose service behaviours in ubiquitous service environments [UbiCompForAll 2010].

2.2 Services and service composition

Services are self describing platform-agnostic computational entities [Liu 2007]. When functionality that cannot be realized by the existing services is required, the existing services can be combined together to fulfill the requirement [Evren 2003]. Hence, Service Composition is the process of creation of new services by combining existing services. Services used in a service composition can come from different providers and domains. Figure 2.1 depicts these concepts.

2.2.1 Service composition approaches

This section is an extract from [Jens 2009]. Five different approaches for service composition are identified:

- Natural language based composition
- Goal-based composition
- Choreography-based composition
- Functional level composition (FLC)
- Aggregation of non-functional properties

The *natural language approach* allows the service consumer to specify its service request in an informal way, i.e. in natural language. A formal specification is then derived from this request and can be used as an input to the composition engine. The core idea is to match

fragments of the natural language request to semantics known to the composition engine, ontology elements and service descriptions.

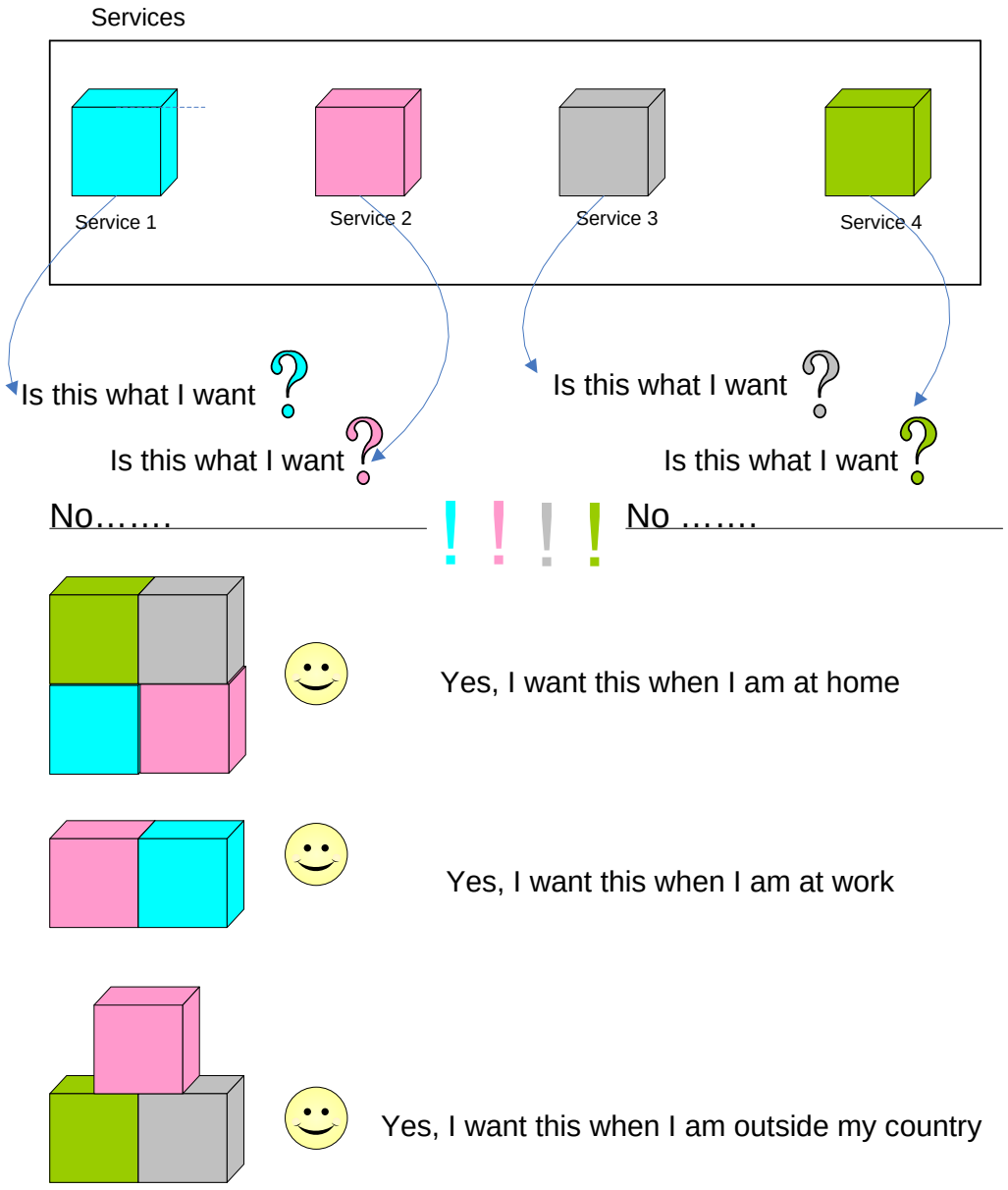

Figure 2.1: An illustration of service composition and why do we need it.

The objective of the *goal-based approach* is to provide a composite service based on a request expressed in a certain format. This request includes a description by goals, i.e. the effects of the service in the "real world". An example for a goal could be "book a flight" or "send a

message". This approach uses the description of the goals in different steps. At the service discovery and selection time, potential services are selected based on the goals they achieve. At the composition step, services are assembled together based on the semantics which includes the goals.

Choreography-based composition evaluates if a composite service matches a standard workflow from a problem domain, a choreography. The workflow can be seen as a composite service because it is a set of relations service provider/consumer. The flow of services can be described in a BPEL flow.

The focus of *FLC* is on selecting a set of web services that, combined in a suitable way, are able to perform a service composition request, i.e. an abstract service described by its Inputs, Outputs, Preconditions and Effects. Since web service discovery and FLC are very close processes, they are usually combined together to find a chain of suitable services. The service composition request defines the overall functionality that the composed service should implement.

The *aggregation of non-functional service* properties checks that a given composite service matches the non-functional properties specified in the service request. Additionally, the analysis of these properties allows to establish a ranking of different composite services. Such properties can be cost, response time or reliability.

The service composition process is triggered by a service need. When a service is needed (at design time or at run time), a service requester specifies a service request containing the desired properties of the service. The service requester sends the service request to a repository that contains descriptions issued by service providers of the services they provide. A search is then executed to discover a service with the desired properties in the repository. If a service with the desired properties could not be discovered, some algorithm is executed that will construct a service composition automatically.

2.3 What is end-user programming and what it is not

End-user programmers are not professional programmers yet they need to program in order to fulfill their daily tasks. Spreadsheets are often touted as the major success story in end-user programming. Majority of them do not realize that they are programming and they do not

consider themselves as programmers [Michael 2005]. For example, a teacher is using spreadsheets for keeping tracks of his grades. But he will not necessarily realize himself as a programmer.

End-user programming does not mean that we are going to train end-users in software engineering; neither it is thought to develop interest in them to become software engineers.

2.3.1 End-user programming approaches

Based on the featured user interface, end-user programming can be divided in to a series of categories [Eirik 2009] [Stav 2006].

1. General purpose programming languages

These are generally suitable for professional programmers and developers. By featuring an interface builder like Microsoft's Visual Basic, some parts of the process are simplified. However, the underlying logic must still be managed by the end-user using traditional programming which makes this method unsuitable as end-user programming tool.

2. Scripting

Scripting languages like JavaScript are generally easier to learn and flexible to use than traditional programming languages. Due to their simpler formation they have less adaptability and usefulness compared to general purpose programming languages.

3. Macro Recording

User actions are recorded for playback later in future. It allows the user to do complex tasks much faster. This is very simple, but may not be appropriate because the actions are specific and are recorded as script, it's difficult to modify a prerecorded macro without knowing the language.

4. Programming by example (PBE)

This approach is similar to macro recording but it is more advantageous for general tasks as compared to macro recording. AI techniques are used by this system to learn a problem from the examples created by the user.

5. Grid based

In grid based programming, the application is basically a grid with a number of “actors” in the cells that interact in different ways. The actors will follow a specified number of rules and everything is laid out clearly with no hidden logic.

6. Visual programming

The Wikipedia defines; “*A **visual programming language** (**VPL**) is any programming language that lets users create programs by manipulating program elements graphically rather than by specifying them textually (also known as **diagrammatic programming**)*”. VPLs can be further categorized, according to the type and extent of visual expression used, into icon-based languages, form-based languages, and diagram languages. This also includes dataflow programming, wherein a program can be modeled as a graph of the data flowing between the operations.

Chapter 3: Methodology

This chapter discusses the ‘problem description’ and explains the approach followed and methodology used from the point of origin to the point of completion in this project work to address the problem.

3.1 Problem Description

The goal of this assignment is to investigate possibilities for end-user service composition using a composed service for the intelligent travel guide as case study. This kind of composed service is aimed for the frequent business or leisure traveler in a typical city environment. In this case the city is Trondheim downtown and the communication infrastructure will be the Wireless Trondheim network as well as the mobile telephony networks. Devices that will be used will range from laptop to various kinds of handheld gadgets. The main position or location enabler will be the GPS system.

The task will consist of the following subtasks:
1. Work out and describe the composition scenarios of the service possibly with several situations (involved actors, interactions, etc.) and different settings (location, multiplicity, preferences, etc.). It is required to specify these situations in such a way that traveler, travelling-agency, and someone-on-traveler’s-behalf will be the composer.

2. Select and define a set of services for the composition.

3. Define the appropriate user interface for executing and composing these services.

3.2 Approach followed

In his Master thesis Jens Einar combined one of the service composition approaches (discussed in section 2.2.1) with correct end-user programming approach to develop the infrastructure for end-user service composition [Jens 2009]. Choreography based approach combined with the concept of visual programming resulted in an interesting and easy to learn interface (EasyComposer).

Scenario-based approach is followed to fulfill the requirements of this project. Scenario building is a widely accepted approach to generate design ideas for new systems and products and to identify the possible users and contexts of use for these systems and products. Scenarios describe individual users in individual usage situations. The value of scenarios is that they concretise something for the purpose of analysis and communication. These scenarios will be exploited both for the design of new services that will be composed by the end-users and for the design of the composition approach itself. In this way, it is aimed to establish a clear understanding of what end-user service composition is, and illustrates the potentials of an end-user service composition approach [UbiCompForAll 2010].

3.3 Methodology Used

Work starts with the study of existing end-user service composition infrastructure for composing Telecom services; the EasyComposer. The next step is to identify which service components to use. This step goes in parallel with brainstorming the usage scenarios that helps in the identification of services for composition. Then the next step is to define the attributes of these services and develop some of these services in EasyComposer. The last step is to describe composition scenarios using the developed services, and to some extent the non-developed services, in EasyComposer. Finally, XML representation for the services and service compositions is worked out to realize them.

Chapter 4: End-user service composition

In this chapter, I explain step by step how the possibilities for end-user service composition are investigated to meet the project requirements. I begin by describing usage scenarios that helps in identifying the service components. Then I follow the same methodology as already described in section 3.3. I have explicitly mentioned the services which I implement in EasyComposer and the ones which already exist in EasyComposer. In addition to this, composition of 'Doctor's Appointment Scenario' [DR-AP] is worked out using EasyComposer. Then, EasyComposer is compared with 'Arctis' to look at how service components are modeled in 'Arctis' and to analyze which areas of EasyComposer can be improved.

4.1 Usage Scenarios

Official Tour

It is Monday 2nd November, a group of three officers from different sectors (one officer from each sector) of Telenor in India are asked to arrange a tour for Norway. Visit is to be scheduled on 16th November for a duration of three days.

Sarah, one of the officers of the team, has been assigned the responsibility of arranging the tour and to inform others. Sarah turned her mobile device on. She selects "Seat Reservation" service to make seat reservations for the tour. As a result, she received a reference number which can be used to buy the tickets. She used "SMS/Message Multicast" service to send the reference number along with other relevant information to her team. Sarah wants to send an e-mail to her friends a week before leaving her country. She composes the service "Send Later" on her mobile device informing her friends about her absence in the country for the period of the tour.

Sarah and her team arrive Norway on 15th November. They are received and escorted to the Hotel. For the upcoming day, Sarah does not want to be disturbed during the meeting hours. She uses "Call Filtering" and "Message Block Filter" service to reject all the anonymous calls and advertisements messages during the meeting. After the meeting, Sarah wants to take tour of the city. She selects "Find Location" service to find markets where she can reach within 15 minutes by walk. She selects some items from a shop, but before buying them she wants to

know whether she can take these items out of Norway. She checked *ban items list* by clicking on "General Information" service. Unluckily she finds some items, in the *ban items list*, she was going to buy. So she has to drop some of them.

While shopping, she gets a "Chat alert". It is his colleague, she opts to chat. Her colleague has found Sarah in the same shopping centre by looking at Sarah's *Current Location.* Her colleague tells Sarah about some special offers. Sarah wants to have a look on those items. Her colleague takes a photo of the items and sends them as an attachment. Sarah "receives MMS" and she is interested in buying them. On her way to the shop, a "Weather Alert" prompted on her mobile device with the prediction of storm in half an hour. Sarah's Hotel is on 20 minutes walk from the shopping centre. But she wants to avail offer at the shop. She selects "Find Transport" service to find a Taxi to get back to Hotel before storm comes. She orders the Taxi that can reach the shopping centre in 10 minutes.

Alternate Scenarios

1. By using "Message-call filtering", Sarah can select other criteria of filtering the messages and calls e.g. filtration by country dialing code.
2. Sarah can select next service as a result of filtering the calls e.g. instead of merely rejecting the calls, Sarah can select other behaviors like:
   a) Receives Call Alerts in the form of SMS.
   b) Sends an SMS to the callee informing about her current location and situation.
   c) SMS to be sent automatically can be manually composed or can contain the contents extracted from the Calendar (that contains the data regarding the current situation).
   d) Forwards the SMS to an E-mail address or any other cell number.
3. Sarah can choose for shops using "Find Location" by filtering the list of the shops with good offers instead of going to every shop first and learning afterwards.
4. Sarah can see the current location of other officers in the Instant Messaging friend list. Their location is always updated and visible with their names in the list.
5. Instead of receiving media files through MMS, Sarah and her colleagues can start a Conference chat sharing their experiences and files about the city and allowing others to join the conference.
6. Sarah can choose the span of time when the "Weather Forecast" should prompt, according to the severity of the weather.

Holiday Tour

John is an employee at an organization and also runs a small business at his home. He is tired of his routine of daily work at office. He wants to have holidays for two weeks and visit Trondheim, Norway.

John turned on his mobile device. By selecting the “General Information” service, he chooses the option to have a list of Holidays in Trondheim. Luckily he found no National Holiday in the weeks he is interested to plan his visit.

He arrived in Trondheim by Train on Saturday. He wants to search for hotels within 10 km distance from the Central Station. He uses his mobile device, selects “Find Location” service and asks the service to display hotel list within 10 km of distance in Kr. 500-2000 range. He finds a room in the hotel of his choice. Now the question is how to get there. He uses “Find Transport” service. The service gives two options (bus/taxi) depending upon his current location. He chooses for bus. But for the bus credit-cards option was disabled. He selects “Current exchange rate” service and finds reasonable rates. He takes the bus and reaches the hotel.

For the upcoming days, he wants to visit the city, see the architectures and some shopping. “Find Location” service tells him that 95% of the shops remain close on Sunday. So he chooses to look around the architecture of the city. On Sunday, he uses “Find Location” service to select some architectures to visit in vicinity of the hotel. He selects the option to *go on foot*. The service guides him the route through all the architectures he has selected. While visiting the architectures, *“Frequently Asked Questions”* prompts on his device. These *“Frequently Asked Questions”* guide him through some general information regarding the architecture he is visiting.

Its Monday, John wants to rent a car. He selects “Find Transportation” service and orders for renting a car. Now he has a car and he wants to look for Traffic rules. He selects “General Information” service and looks for traffic rules in the *Rules Category*. John is now visiting random places in the city on his car.

While driving the car in the city, he gets several alerts regarding some special festivals going on in the vicinity of his location. He reads the details of these special events in the “Special Occasions” service. He finds one event very interesting. He chooses that event and the “Find

Location" service guides him to that event. "Find Parking" service guides him to the vacant parking slot.

Alternate Scenarios

1. Instead of searching for the Hotels after arriving the city, John can make the booking beforehand.
2. Instead of looking for a parking slot after reaching the destination, John can decide prior to go to the event, depending upon the availability of the parking slots.
3. "Find Location" service can be intelligent enough to give options to the user to choose the route depending upon the traffic level on different routes leading to the same destination.
4. John can select from a list of "Special Festivals" for which he wants to receive the alerts.

4.2 Definition and Description of Services

This section defines a set of self-contained atomic services or service components that will be used in composing the scenarios. Description of the services also includes the design of dialogue boxes with settings and preferences.

Receiving Email

This service receives an E-mail. This triggers rest of the behaviour of service composition.

Send E-mail

This service sends an E-mail to a cell number. The e-mail can contain any other file.

Settings and Preferences

1. Select or enter the address to which the e-mail is to be sent.
2. Enter addresses in CC & BC fields if any.
3. Enter the subject of e-mail if any.
4. Enter the message to be sent.
5. Attach a file if any.
6. Select the next service to be executed.

Receiving SMS

This service receives an incoming SMS. This triggers rest of the behaviour of the service composition.

Send SMS

This service sends an SMS to a cell number.

Settings and Preferences

1. Select or enter the contact to which the message is to be sent.
2. Enter the message to be sent.
3. Select the next service to be executed.

Receiving MMS

This service receives an incoming MMS. This triggers rest of the behaviour of the service composition.

Send MMS

This service sends an MMS to a cell number. The message can contain media files as an attachment.

Settings and Preferences

1. Select or enter the contact to which the message is to be sent.
2. Enter the message to be sent.
3. Attach the video, audio or picture file.
4. Select the next service to be executed.

SMS/MMS Multicast

This service is similar to “Send SMS” and “Send MMS” services but the difference is that it is used if user wants to send the same SMS/MMS to a group of friends/contacts.

Settings and Preferences

1. Select or enter two or more contact numbers to which the message is to be sent.
2. Select the option to send “All friends”, “All family members” and “All coworkers”.

3. Enter the message to be sent.
4. Attach a file if any.

Message Block Filter

This service filters the incoming SMS/MMS/E-mail. The filtering criteria are selected by the user. If the criteria match, the message is blocked. Rest of the messages undergoes the service defined to be executed next.

Settings and Preferences

1. Select the filtering criteria that can be one or more of the following:
    a) One or more phone numbers.
    b) One or more e-mail addresses.
    c) Message containing <this> text.
    d) Anonymous phone number.
    e) Advertisements.
    f) Country Dialing Code.
    g) Domain name (for email addresses).
2. Select next service to be executed as a result of filtration.

Message Auto-Forward

This service forwards the SMS/MMS/Email to another cell number or e-mail address automatically.

Settings and Preferences

Select one or more e-mail addresses or cell numbers to which the SMS/MMS/Email has to be forwarded.

Instant Message

By using this service user can start a chat session with friends available/online in his friend list. This service is helpful for short discussion and scheduling of tasks.

Settings and Preferences

1. Maintain a number of profiles with different settings, like Chat alert tones, according to location/situation.

2. Enable or disable the option of “Let others see my status”. By disabling the option, it is chosen to hide the status of displaying current location to others. If it is enabled, other friends can see his current status (location) along with user name in their friend list.
3. Maintain a list of friends which can be e-mail addresses or contact numbers.
4. User can choose to start conference chat with two or more friends.
5. Select next service to be executed if any.

Find Location

This service when selected, guides the user to a destination from his current location. The service is intelligent and can sense the current location of the user. The user interface of this service contains ‘tabs’. These tabs contain the following sub-services:

a) Find Restaurant
b) Find Hotel
c) Find Shopping Centres
d) Find Architectures
e) Miscellaneous

Settings and Preferences for “Find Restaurant”

1. User can select a restaurant from a list of restaurants or User can search for a restaurant by either one or all of the following choices:
   a) Distance range (from x km to y km) within which he wants to find a restaurant.
   b) Time to reach; If he selects this option he has to specify the “means of transportation” he is using to reach the restaurant i.e. bus, car, on foot.
   c) Money range (from Kr. x to Kr. y)
   d) Food (vegetarian or non-vegetarian)
2. According to the entries he make in the above choices, a list of restaurants will be displayed within the same tab window. The properties of the restaurants i.e. their distance from user’s current location; time to reach by (bus/car/on foot); Money range; Food - will be displayed in the same list.
3. User selects one of the restaurants of his choice and click on the MAP button. MAP will be displayed in a separate window. He can see his location and location of the restaurant on the MAP. On the left side of MAP there are options of “means of transport” he is going to use to reach the restaurant. Depending upon the “means of transport”, MAP guides the user from his current location to the restaurant.

Settings and Preferences for "Find Hotel"

1. User can select a Hotel from a list of Hotels or User can search for a Hotel by either one or all of the following choices:
    a) Distance range (from x km to y km) within which he wants to find a Hotel.
    b) Time to reach. If he selects this option he has to specify the "means of transportation" he is using to reach the Hotel i.e. bus, car, on foot.
    c) Money range (from Kr. x to Kr. y)
2. Accordingly, a list of Hotels having vacant rooms will be displayed within the same tab window. The properties of the Hotels i.e. their distance from user's current location; time to reach by; Money range; will be displayed in the same list.
3. User selects one of the Hotels of his choice and click on the MAP button. MAP will be displayed in a separate window. He can see his location and location of the Hotel on the MAP. On the left side of MAP there are options of "means of transport" (i.e. bus, car, on foot) he is going to use to reach the Hotel. Depending upon the "means of transport", MAP guides the user from his current location to the Hotel.

Settings and Preferences for "Find Shopping Centres"

1. User can select a Shopping Centre from a list or User can search by either one or all of the following choices:
    a) Distance range (from x km to y km) within which he wants to find a Shopping Centre.
    b) Time to reach. If he selects this option he has to specify the "means of transportation" he is using to reach the Shopping Centre i.e. bus, car, on foot.
    c) Kind of Shopping Centre(i.e. Garments; Furniture; Cosmetics; Crockery; Electronics; Miscellaneous)
2. Accordingly, a list of Shopping Centres will be displayed within the same tab window. The properties of the Shopping Centres i.e. their distance from user's current location; Time to reach by; kind of Shopping Centre; Opening hours; will be displayed in the same list.
3. User selects one of the Shopping Centres of his choice and can see "Special Offers" by clicking on the button. This will open a website where the user can find updated offers. Then by clicking MAP button, MAP will be displayed in a separate window. He can see his location and location of the Shopping Centre on the MAP. The MAP window has the same properties as described earlier.

Settings and Preferences for "Find Architectures"

1. User can select Architectures from a list or User can search by either one or all of the following choices:
    a) Distance range (from x km to y km) within which he wants to find a Shopping centre.
    b) Time to reach. If he selects this option he has to specify the "means of transportation" he is using to reach the Shopping Centre i.e. bus, car, on foot.
2. Accordingly, a list will be displayed within the same tab window. The properties of the Architectures i.e. their distance from user's current location; time to reach by (bus/Car/on foot); will be displayed in the same list.
3. User selects several architectures and has the option to set their sequence of visit (1, 2…..) of his choice. Then by clicking on MAP button, MAP will be displayed in a separate window. He can see his current location and the route of the architectures to visit according to his sequence if any. The MAP window has the same properties as described earlier.

Settings and Preferences for "Miscellaneous"

1. User enters an already known address explicitly.
2. Click on the MAP button to see the route from his current location to the address specified.

Find Transportation

This service helps the user to find transport with respect to his current location.

Settings and Preferences

1. User can select between the options
    a) Bus

       The information related to the nearest bus stops will be displayed along with the timings and routes of the bus.
    b) Taxi

       The user can order a nearest taxi by selecting the timings on which he wants to get in the taxi.
    c) Rent a car

       This will display how and from where to rent the car.

d) Rent a bike

This will display how and from where to rent the bike.

2. User can choose to enter the distance range within which he needs to find a transport.

Weather Forecast

This service helps the user to know about the weather forecast. Weather Alerts will be sent to the user mobile device depending upon the preferences the user selects.

Settings and Preferences

1. User enters the duration for which he wants to know the weather forecast. Forecast will be displayed on a web-site.
2. For Weather Alerts to be sent on the mobile device , user select weather severity level from the options
    a) Snowfall
    b) Rain
    c) Storm
3. Along with all the above options, user has to set the time duration before which the weather alert has to be sent on his mobile device. This time duration is the time to reach his home/hotel from his current location. For example for *Rain* he wants an alert 30 minutes earlier. For *Storm* he wants an hour earlier. This time duration also depends upon the 'means of transport' user is likely to avail.
4. User has the option to enter the 'means of transport' he is availing so that alert can be sent in more accurate timings.

General Information of the City

This service helps the user to have some basic knowledge about the city.

Settings and Preferences

Select one of the following options to have some basic knowledge

a) Currency Exchange rate
b) General Rules
c) Traffic Rules
d) National Holidays
e) Special Occasions (This option also allows the user to subscribe for alerts)
f) Ban Items (cannot be taken out of the country)

g) Emergency numbers

4.3 Services Implemented in EasyComposer

Section 4.2 describes the set of services which will be used in composition scenarios. I implemented some of these services in EasyComposer as well. These services already defined in section 4.2 are:

- Receive E-mail
- Send E-mail
- Receive SMS
- Send SMS
- Receive MMS
- Send MMS
- SMS/MMS Multicast
- Message Block Filter
- Message Auto-Forward
- Instant Message

4.4 Existing Services in EasyComposer used in Composition Scenarios

A number of self-contained services already exist in EasyComposer for end-user service composition developed by [Jens 2009]. In this section these services are described which will be used together with the services defined in section 4.2 to describe the composition scenarios in the next section.

Time Manager

This service component is basically like a scheduler or calendar. User defines one or more time durations. If time match is found, next service defined by the user is activated.

Settings and Preferences

1. Define the time of day, days of week, months of the year for when the succeeding behaviour is active.
2. Define the next service if none of the time filters match.

White list

This is a list of important contact numbers. A number entered in this list is thought to be an important person, authority, relatives or other persons that the user wants to talk to or does not mind being disturbed by at any given time. The white-list service is not that useful by it self. To create a useful composition one needs to combine white-list with at least one other service providing some sort of blocking and the white-list functions as a bypass of this blocking.

Settings and Preferences

1. Add, Edit and Remove numbers in the white-list.
2. Next service if white-listed.
3. Next service if not white-listed.

4.5 Description of Composition Scenarios

4.5.1 Precautious Weather Alerts

Summary
This scenario describes how to compose a service to receive weather alerts depending upon the user location. The end-user composing this scenario is a traveler.

Actors
User, Location enabler.

Usage Scenario
A traveler is in the Trondheim city. This city has a very unpredictable weather. He wants to return to his hotel in time depending upon the severity level of the weather for example in case of storm or heavy rain. But he does not want to receive alerts every now and then. In other words, he needs to receive weather alerts within right timings with respect to the distance from his current location to his hotel.

Service Composition
First user enables his positioning system. Select the ‘Time Manager’ service. Enter the timings in which the user wants to receive the weather alerts. Now the user has to select ‘Weather Forecast’ service as next service to be executed in the timings defined in ‘Time

Manager'. In the 'Weather Forecast' service, user selects the severity levels (Heavy Rain, Storm, and Snowfall) in which he wants to receive the weather alerts. For every severity level, user defines the forecast time prior to which he wants to receive the alerts. An illustration of this composition is shown in figure 4.1.

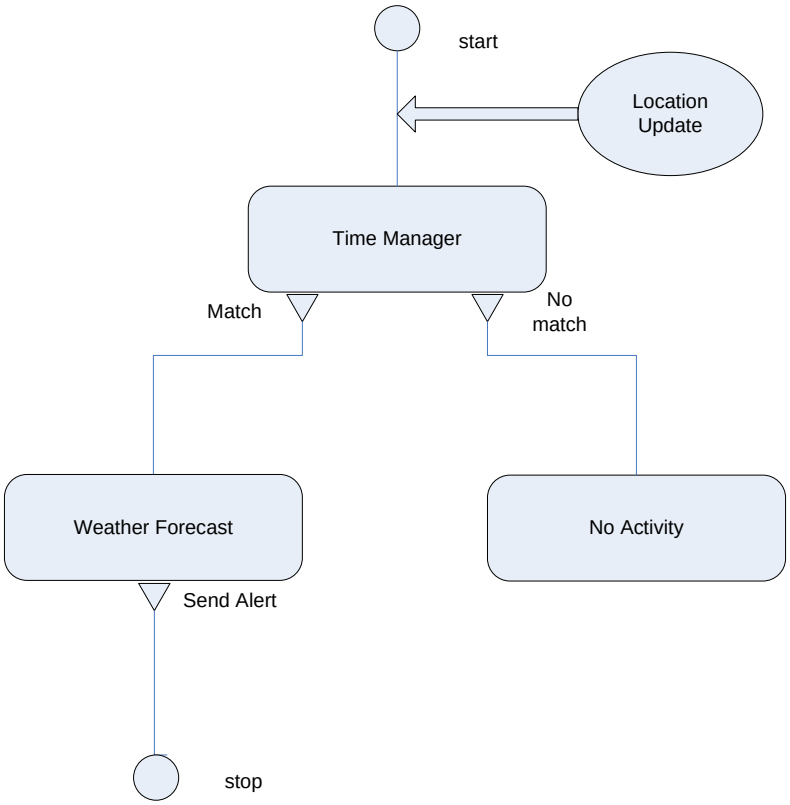

Figure 4.1: Precautious Weather Alerts

4.5.2 Intelligent E-mail Manager

Summary

This scenario describes how to compose a service that can manage user's incoming e-mails by filtering the e-mails, sending custom auto-replies and forwarding the e-mails to selected e-mail address.

Actors

User, Email sender.

Usage Scenario

A Business person wants to manage his incoming business e-mails. He cannot check his emails for most of the time due to his busy schedule. He does not want to delay replies to e-mails from his business contacts. Sometimes he is out of the country with tight busy schedule. He wants his business contacts to know his current status when he is not there to reply their emails. When he is on holidays, he wants to forward his emails to his business colleagues so that they can reply the important emails if any. During holidays he thinks that during some hours he is free enough to check emails and reply to them. But for this, he wants to receive alerts as SMS on his mobile to notify him if there is any email to check.

Service Composition

Select the 'Message Block Filter' service. User has to select the filtering criteria of the incoming email. The user selects to block all 'Anonymous contacts', 'Advertisements' and 'Specific Domain'. Then select next service as 'Time Manager' towards which all the e-mails, that passed the filtering criteria, are directed. In the 'Time Manager' he enters the details regarding the days of week and the timings during which he wants to manage his e-mails in an intelligent way. Moreover, he needs to define the next service which will be executed in those particular timings. For example, when he is out of the country, he defines the next service as 'Send E-mail' to the sender describing his status. When he is on holidays, he defines the next service as 'Message Auto Forward', where he specifies the e-mail address of the colleague to whom he wants to forward the e-mail. During holidays when he thinks he will be free enough to check the emails, he defines the next service as 'Send SMS'. In this service he selects his cell number where the SMS will be sent that notifies him if there is any e-mail to check. This composition is depicted in figure 4.2.

4.5.3 Feel Like Home / Be there always for ALL

Summary

When special contacts make a call or send an SMS, this service composition starts Instant Messaging. User can also start Instant Messaging conversation when he has two or more callers.

Actors

User, Callers and SMS/MMS sender

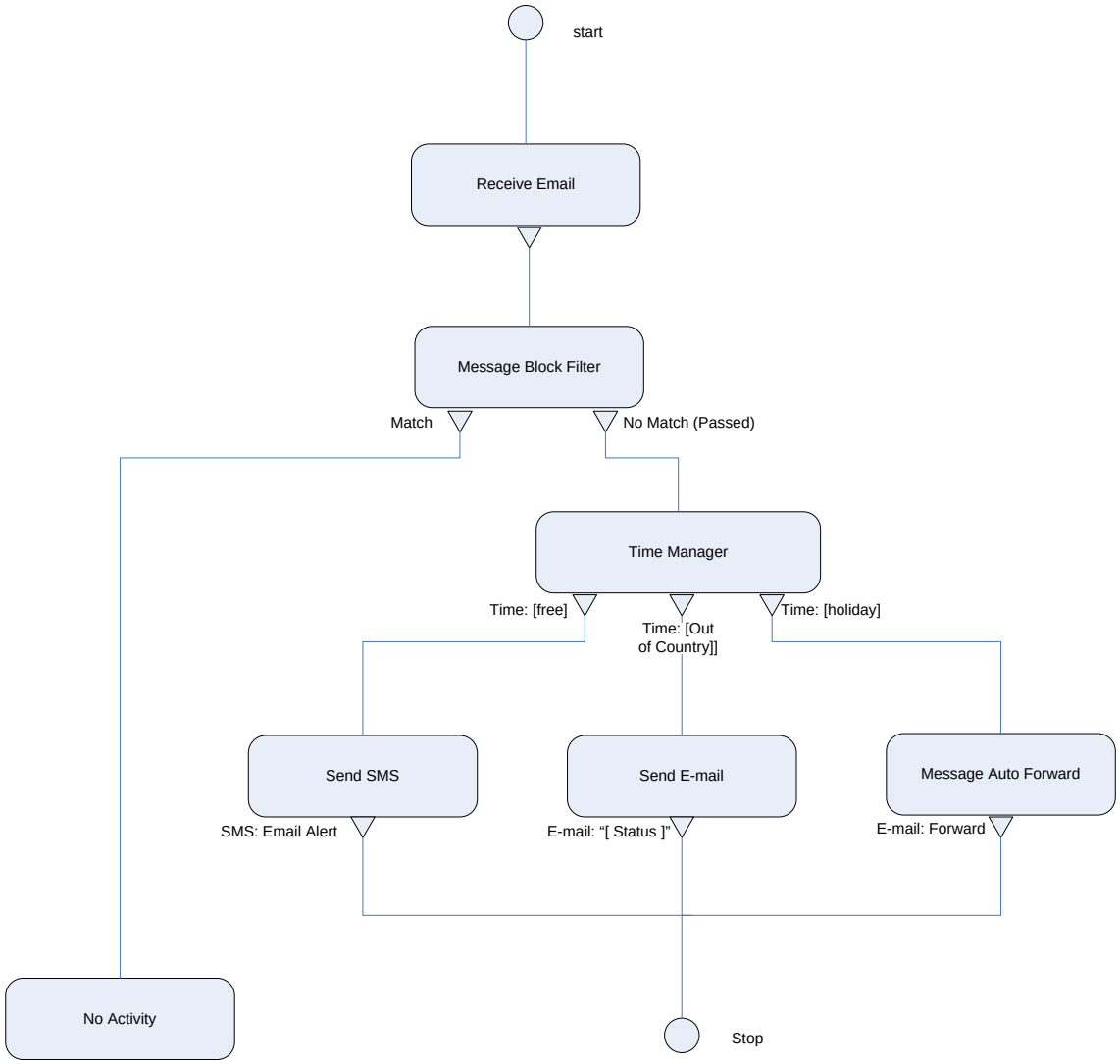

Figure 4.2: Intelligent E-mail Manager

Usage Scenario

A business traveler is in a city for two weeks without his family and friends. He misses his home and friends. When he is on a scheduled trip on a bus for the city, he wants to be in continuous conversation with all his family members and friends who try to contact him by making a phone call or sending an SMS. He does not want to delay or have busy status for any of them. He wants to be in contact with all of them simultaneously. Moreover, he wants to share his city trip with them by sharing photographs he takes on his way. He does not want to receive calls from contacts other than his family and friends. In short, he wants to 'Feel Like Home'.

Service composition

User selects the service ‘White list’. The friends and family contacts have to be added in this list. Rest of the contacts will be automatically blocked. Then user defines the next service as ‘Instant Messaging’. So that caller and user come into contact by ‘Instant Messaging’. The next service to be defined after this is ‘Disconnect Call’. User selects the option of ‘Start Conversation’ from the ‘Instant Messaging’ dialogue. This has to be done so that the next callers can be added to the ‘Conversation’. Now, user selects ‘Message Block Filter’ service and selects the criteria so that only family and friends SMS/MMS pass the filter. The senders are then added to the ‘Conversation’ by defining the next service as ‘Instant Messaging’. An illustration of this composition is shown in figure 4.3.

Alternate Stories

Instead of just disconnecting the calls of the contacts not in family and friends list, user can ‘Send SMS’ with some message and then disconnect the call. So that the caller does not keeps on calling again and again.

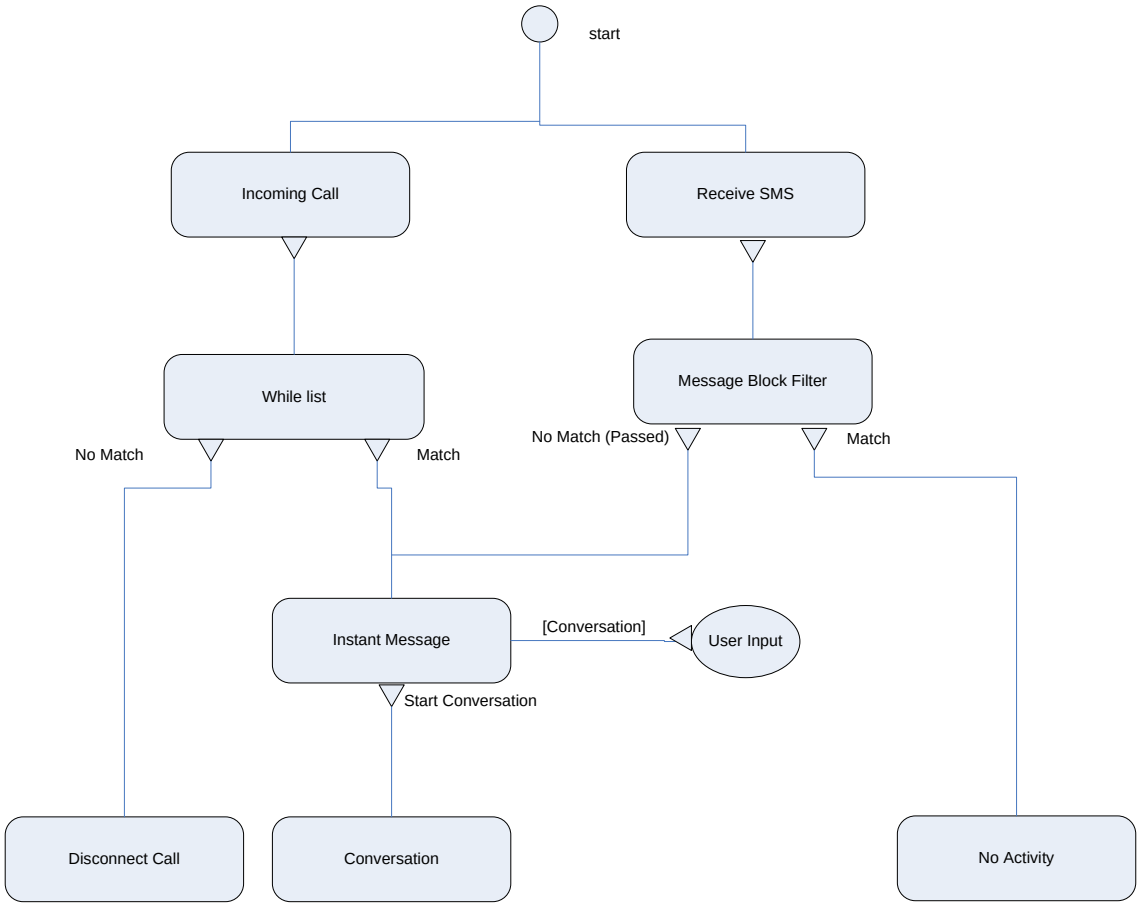

Figure 4.3: Feel Like Home / Be there always for ALL

4.5.4 Automatic SMS/MMS Multicast

Summary

To compose a service that filters a specific SMS/MMS and forwards it to a group of contacts.

Actors

User, Coworkers

Usage Scenario

A person is responsible to inform all the colleagues about an official meeting in time i.e. as soon as the Boss sends him an SMS regarding the meeting schedule. At the same time, he is expecting his personal business partner to meet him. He is unsure whether he would be able to inform his colleagues in time. Therefore, he wants a service that may forward the SMS automatically to all the participants of that meeting.

Service Composition

User selects the 'Message Block Filter' service so that only the Boss number is passed through the filter. The user defines the next service as 'SMS/MMS multicast'. In this service dialogue he adds phone numbers of all the participants to whom the SMS is to be sent. He then chooses the message 'from previous service' from the dialogue that is to be sent to the meeting participants. User has to select this option because he wants the same message containing the meeting details to be forwarded to the participants. This composition is illustrated in figure 4.4.

Alternate Stories

If user is unsure about the Boss SMS, then he may forward it to any other colleague so that he can read the message and multicast accordingly.

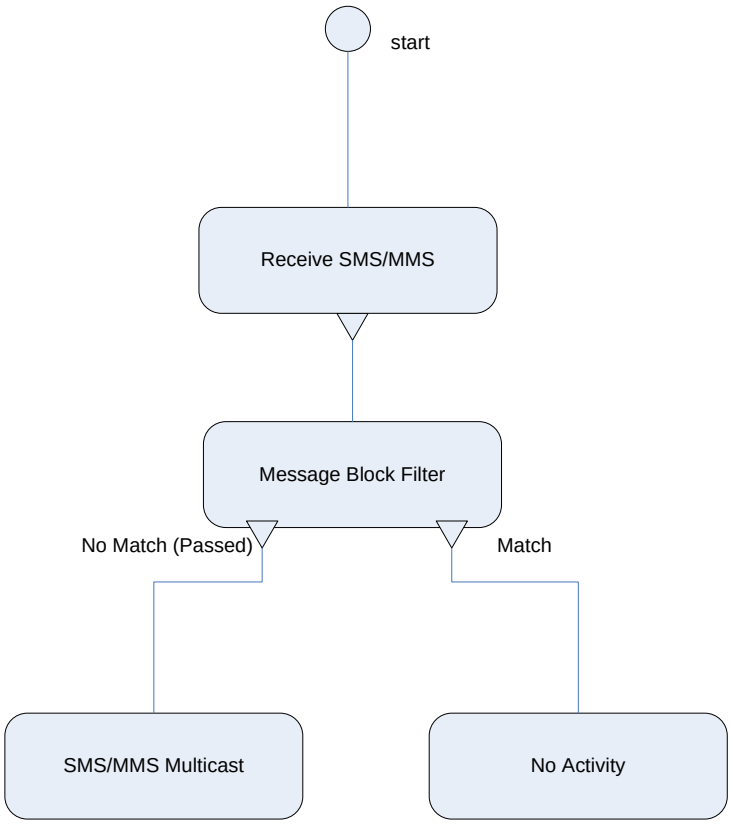

Figure 4.4: Automatic SMS/MMS Multicast

4.5.5 Find Location and Transport

Summary

User wants to find a location in a city and select a means of transport to reach there.

Actors

User, Location enabler, Map provider.

Usage Scenario

A person arrives in a city. This is his first ever visit to this city. He wants to reach to the Hotel where he has reserved a room in advance. He has the address of the Hotel but does not know the direction. He needs a map that is intelligent enough to tell him the direction from his current location and displays all the possible ways to reach that Hotel. He wants to know all the means of transport available as well. So that he can pick the one of his own choice.

Service Composition

User selects ‘Find Location’ service. Here he selects the *Miscellaneous* tab where he can enter the address of the Hotel. He then clicks the *Map* button which displays the route to the Hotel from his current location. On the *Map* window he selects the means of transport he is going to use to attain his destination. He selects the *Bus* option. The *Map* displays the bus route (highlighting the Bus Stops) from his current location to the Hotel. Next he selects the ‘Find Transport’ service. The service gives him the default options of means of transport available at his location. He selects the *Bus* option. The service displays the information regarding the timings and bus routes of the nearest Bus Stops. This composition is depicted in figure 4.5.

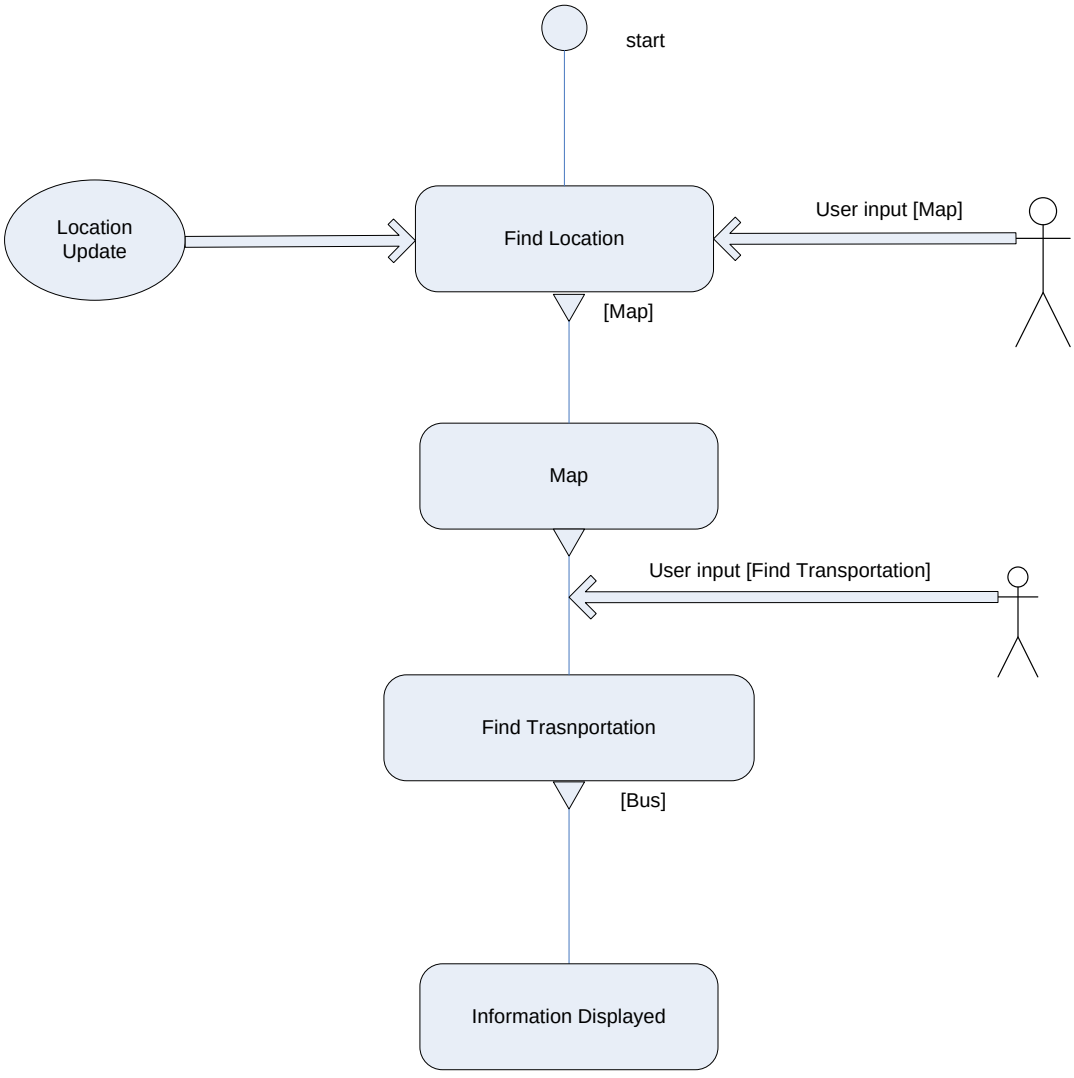

Figure 4.5: Find Location and Transport

4.6 Composition of Doctor’s Appointment Scenario in EasyComposer

‘Doctor's appointment’ was specified as part of the UbiCompForAll project in order to identify potential users of the technology that will be developed in the project and to generate design ideas for the services these users might compose [DR-AP].

In this section, summary and problem description of the Doctor’s Appointment are taken as extracts from [DR-AP]. Then these sub-scenarios of ‘Doctor’s appointment’ are defined using the EasyComposer i.e. some of the service components described in section 4.2 and section 4.4 are used to cover some functional parts of the Doctor's appointment sub-scenarios.

Summary of Doctor’s Appointment

Ove creates a service to help his ageing mother to get to her doctor's appointments. This service keeps track of the appointments, reminding her when its time to go, it helps her find her way to the doctor's office, and it lets him keep track of her progress and alerts him of problems.

Problem description

Oda is 75 years old, and starting to get forgetful. She is finding it harder to manage on her own as she gets older and starts to forget things. This is something that will happen to virtually all people who live to an old age while physically healthy enough to manage on their own. She has trouble remembering her doctor's appointments and keeping track of the bus schedule.

Her son Ove is concerned for his mother but also very busy with his work. He wants to assist her, but he also wants her to manage on her own as much as possible. For Oda to keep her cognitive abilities and independence as long as possible she should exercise those cognitive abilities. If she is provided with the right assistance she can keep track of her appointments and feel safe on her own, helping her feel confident and independent, which will in turn help her stay mentally fit longer.

Iterative approach is followed in the description of the scenario. Four versions are presented starting from the simple version to the complete one. In this scenario, two elements that will take part in the composition are described:

Enabler: a service or software that provides functionality to the composed service. It may be available online. These are not used directly in the composition.
Component: Building block that is used in the composition.

According to these definitions, EasyComposer constitutes the building blocks which are used directly in the composition i.e. components.

4.6.1 Version 1 – Calendar with alerts

Usage Scenario

This most basic version entails connecting a calendar storing appointments to the sending of alerts to Oda's phone. Ove also wants the alerts sent to him by email, and he wants the system to keep a log of what it does. Composition is shown in figure 4.6.

The system must support the following tasks:

- *Enter appointment*: Ove will enter Oda's appointments using his home computer.
- *View appointments*: Ove also wants to be able to check the calendar.
- *Send alerts*: Messages should be sent to Oda's phone at specific times before a coming appointment, and Ove wants to have a reminder by email.
- *View event log*: Ove wants to be able to log on to the service from his computer and view a list of what the service has done. He thinks this will be useful for testing, and for keeping track of the system as it gets more complex.

Composing Version 1 in EasyComposer

Amongst the four tasks described above, *Enter appointment* and *view appointments* is handled by *CalendarAlert* shown in figure 4.6. *CalendarAlert* is an enabler which is available online through a web browser. *View Event Log* is also assumed to be a sort of online enabler with web interface. Hence, only one task is left to be composed i.e. *Send Alerts.* Composition is described next.

In EasyComposer, select the services ‘Send SMS’ and ‘Send E-mail’. In ‘Send SMS’ service, Ove enters Oda’s *Phone Number* and a *Message* that he wants to send Oda. Or, for *Message,* he can select the option of “*from previous service*” which means that the message that will be

sent to Oda, is from the 'CalendarAlert'. In 'Send E-mail' service, Ove enters his *Email Address* and a *Message* he wants to receive as reminder in his e-mail. Similar to the 'Send SMS' service, Ove can also choose *"from previous service"* option for the *Message.* The composition is shown in figure 4.7. For the sake of clarity, enablers are depicted as ellipses and components as rectangles.

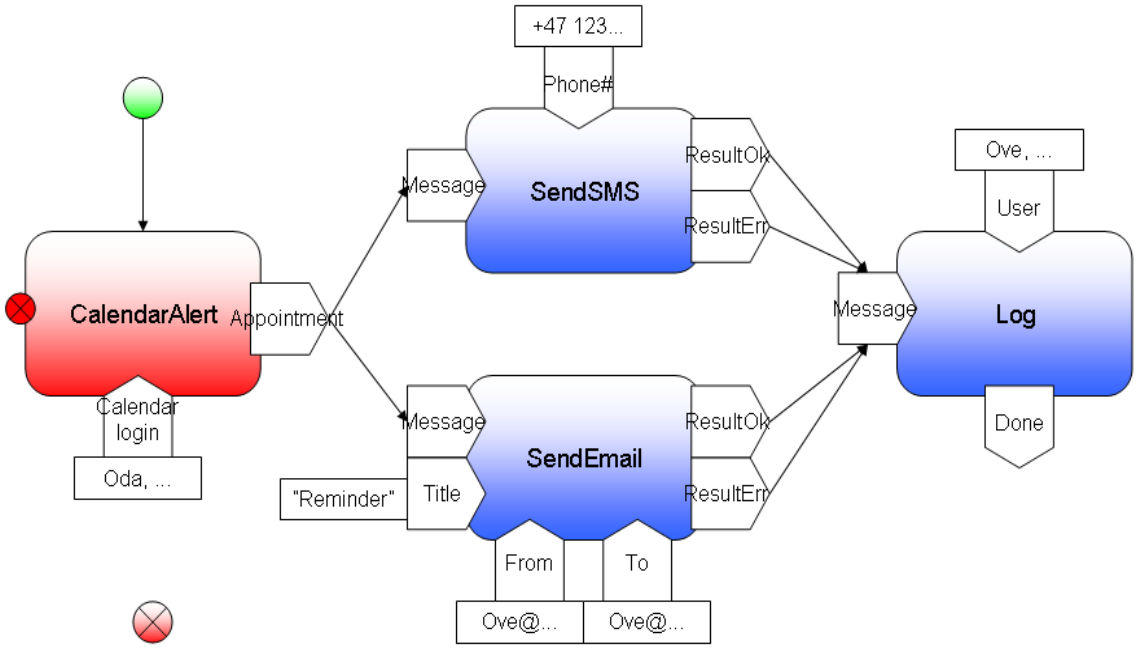

Figure 4.6: Composition of Version 1: Calendar with Alerts [DR-AP]

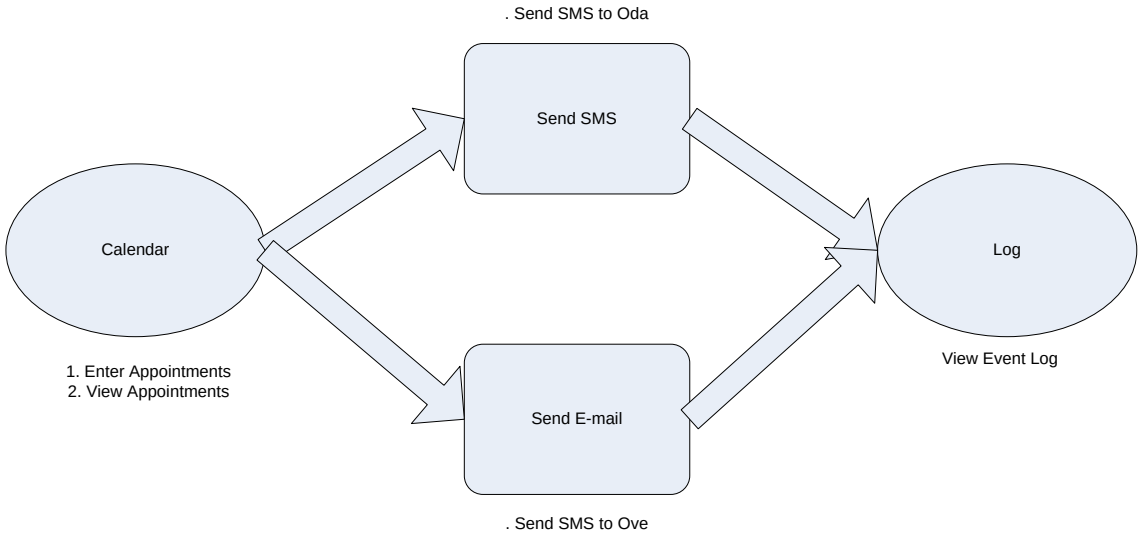

Figure 4.7: Composition of Version 1 in EasyComposer. Enablers are depicted as ellipses and EasyComposer components are depicted as rectangles.

4.6.2 Version 2- Schedule with position check

Usage Scenario

The service should check the bus schedule, finding a bus to get Oda to her appointment on time and informing her of this in a message. And it should check her position shortly before the appointment, using her phone or an RFID tag, and alert Ove if she is still at home. Send message to Ove, alerting him of Oda's upcoming appointment.

In the first version Ove relied on the calendar for scheduling, as it can send alerts at specific times before an appointment. But that will no longer be satisfactory as the service gets more complex. Ove wants to be able to specify many tasks at different times before an appointment. There must be some way to setup a schedule for different tasks.

This system must support the following tasks; composition is shown in figure 4.8:

- *Scheduling*: Ove wants to be able to create a task list with time offsets.
- *Bus Information*: This provides access to bus time tables.
- *Location*: Keeps track of Oda's current position.

Composing Version 2 in EasyComposer

Bus Information is an enabler which is available online. *Location* is tracked through some means. The components, *GetBusTime* and *IsInArea*, are representing the functionality of *Bus Information* and *Location* enablers respectively. In figure 4.8, some Calculation Components can be seen. These components along with other components, like *GetCurrentTime* and *GetBusTime*, are used to find the time Oda needs to be at the destination. This calculated time will be sent to Oda as SMS. Total functionality is more than the service to be composed. In short, *Scheduling* is the task needed to be composed in addition to version 1. Composition is described next.

In EasyComposer, select the service 'Time Manager'. Time Manager provides the functionality that Ove wants for *Scheduling.* He can create a task list by entering the days of the week and the time range during which he wants to execute his tasks. After entering the day and time duration for every task, he has to define each task he wants to execute during that time. For example, to remind Oda about the appointment one day earlier, Ove enters the day and time in 'Time Manager' at which he wants to send SMS to Oda. Then he selects the service 'SendSMS' as the task he wants to execute during that time. Likewise, he can create

the whole list of tasks. The composition of version 2 in EasyComposer is shown in figure 4.9. As said before, enablers are depicted as ellipses and EasyComposer Components are depicted as rectangles in figure 4.9.

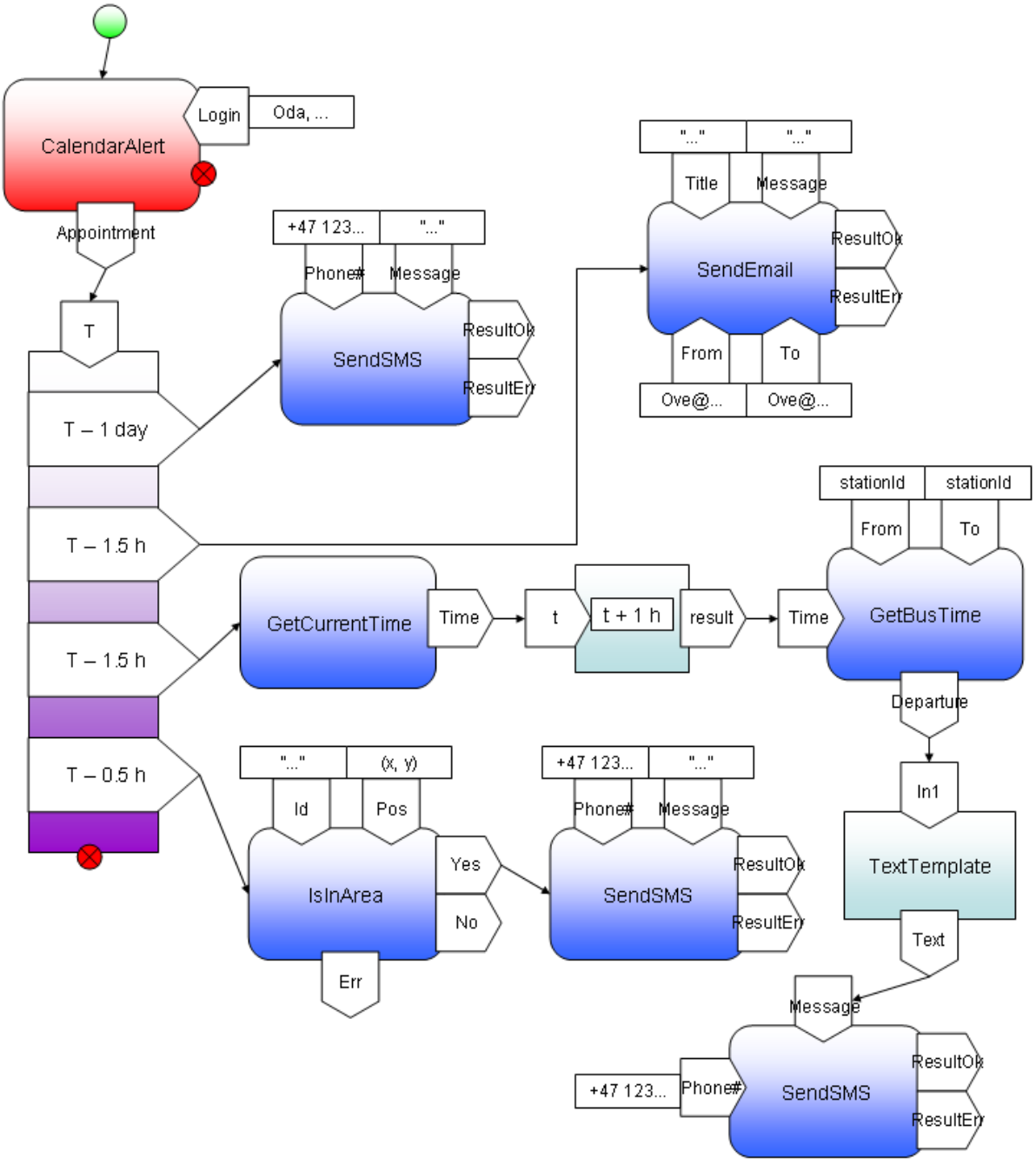

Figure 4.8: Composition of Version 2: Schedule with position check [DR-AP]

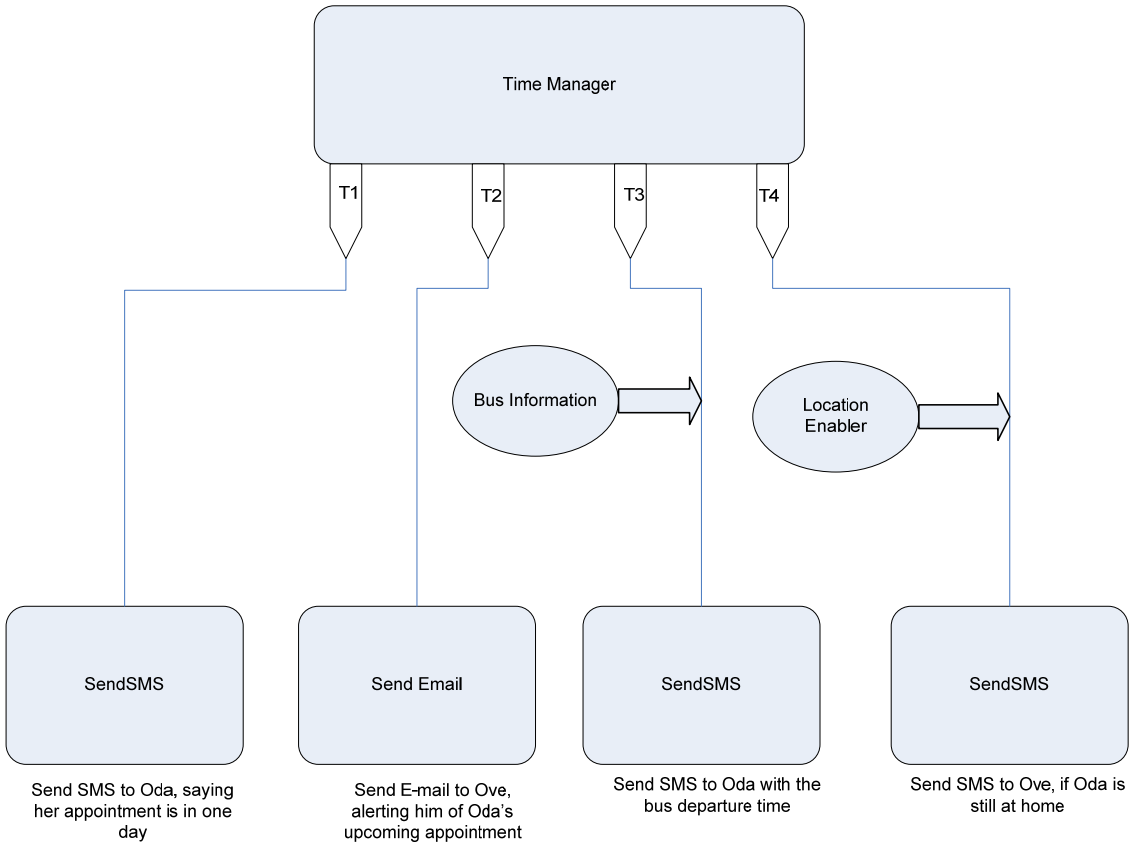

Figure 4.9: Composition of Version 2 in EasyComposer. Enablers are depicted as ellipses and EasyComposer components are depicted as rectangles. T1, T2, T3 and T4 are the time durations in which the tasks specified by Ove will be executed.

Versions 3 and 4

Version 4 improves the scheduling information. It introduces the enablers to track Oda's position. However, the basic functionality of versions 3 and 4 remains the same as discussed in version 1 and 2.

4.7 Comparison of EasyComposer with Arctis - A UML-based tool for Service Engineering

In this section comparison between EasyComposer and Arctis is described. This comparison is worked out to look at how services or service components are modeled in Arctis and to analyze which areas of EasyComposer can be improved.

4.7.1 Service Logic

Arctis

Tool support implemented by the Arctis plug-ins is shown in the figure 4.10.

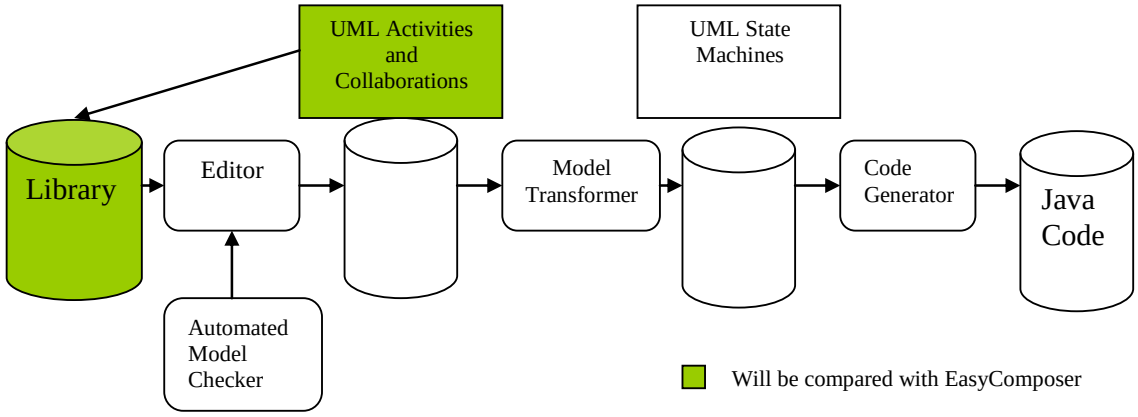

Figure 4.10: Overview of the Arctis tool support [Frank 2009]

Service building blocks in Arctis are defined by UML collaborations, UML activities and External State Machines (ESM) [Frank 2009] [Surya 2009] [Rolv 2009] [Arctis 2010]:

UML Collaborations: UML collaborations give an overview of the system structure and demonstrate the roles played by the system components as shown in figure 4.11. These system components or collaboration roles are presented as activity partitions in the UML activities.

UML Activities: They complement the UML collaborations and explain the internal behaviour of the building blocks as shown in figure 4.12. At the frame of each activity there are pins. There are three different types of pins:

- Initial pin: *start* following the initial node.
- Streaming pin: This pin is drawn in black and can pass tokens (an activity is understood by token flow semantics while an activity is active).
- Mutually Exclusive pins: These pins have additional box around it. If an activity can be terminated in alternative ways we use these pins.
- Simple Terminating pin: The activity is terminated when token is passed through this pin.

These pins refer to the events that are visible external to the activity. These externally visible events are used to plug the building blocks together by using control logic between them.

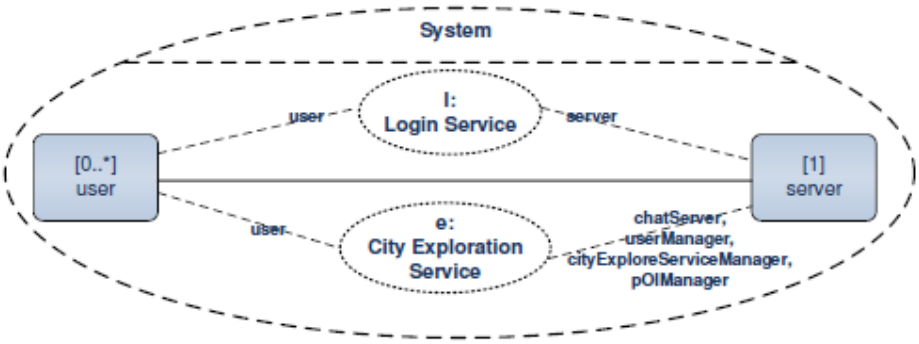

Figure 4.11: UML Collaboration; Example taken from "Exploring the City"; 'user' and 'server' are collaboration roles where server is partitioned into four collaboration roles. 'Login Service' and 'City Exploration Service' are collaboration uses [Surya 2009].

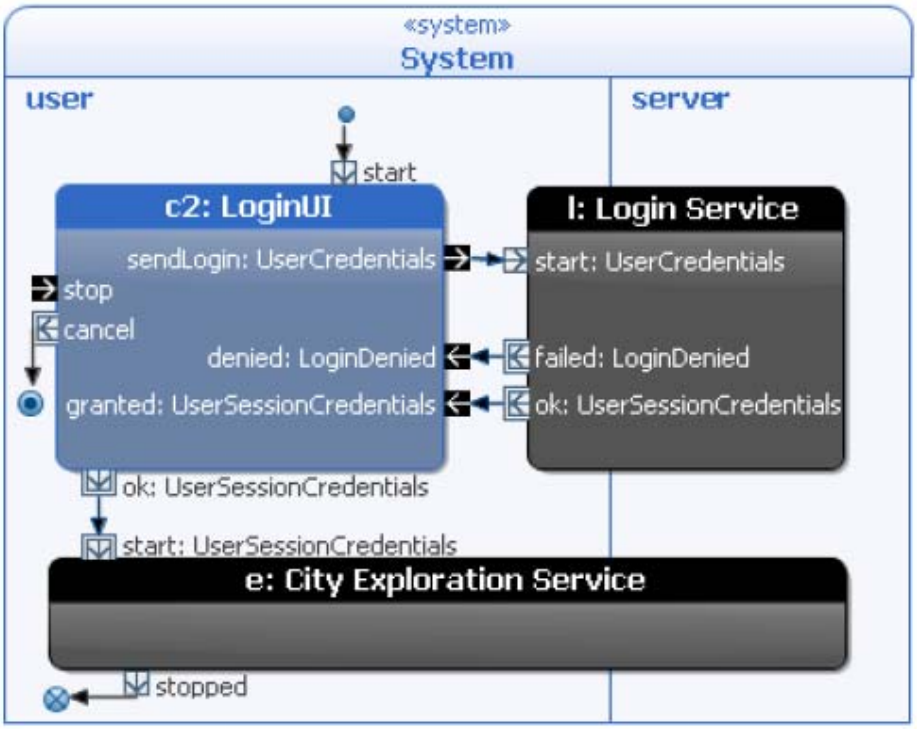

Figure 4.12: UML Activity; It contains partitions namely 'user' and 'server' corresponding to the collaboration roles in UML collaboration. 'Login Service' and 'City Exploration Service' are presented by the rounded boxes in figure 4.11. They cross the partition borders since they are executed by both user and server (cross cutting service behaviour) [Surya 2009].

External State Machines (ESM): This is not an SDL State Machine. This is a special state machine that encapsulates the building blocks and provides an external view hence can also

be thought as building block interface. ESM defines the allowed sequences of tokens passing through pins. As depicted in figure 4.13, slash ("/") distinguishes pins triggered from the outside of the building block from those triggered as a reaction to an external trigger. ESM allows the engineers to compose services without looking into the internal details of the building blocks.

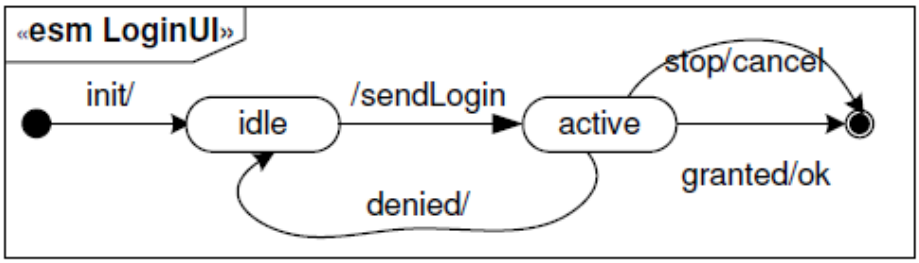

Figure 4.13: ESM; This diagram is the continuation of the example explained in figure 4.11 and 4.12. It is an external view of 'LoginUI' (login user interface) [Surya 2009].

Separation between two kinds of building blocks is introduced in [Surya 2009];

- **Collaborative Service Blocks:** These blocks model the cross cutting service behaviour among several collaboration roles as shown in the UML activity in figure 4.12.
- **User Interface Blocks:** These blocks represent the interaction from and to the users as well as detailed operations on user interface elements. For example, the building block *LoginUI* in figure 4.12.

Activity diagrams can be used to define the behaviour of user interface building blocks as depicted in figure 4.14. According to [Surya 2009], user interface blocks are local building blocks which are only represented in the activity perspective.

EasyComposer Service Building Blocks

In contrast with Arctis where building blocks take into account the behaviour of server as well, EasyComposer service building blocks are built from the perspective of user interface. EasyComposer is more intuitive and easy to use. The objective of EasyComposer is to provide

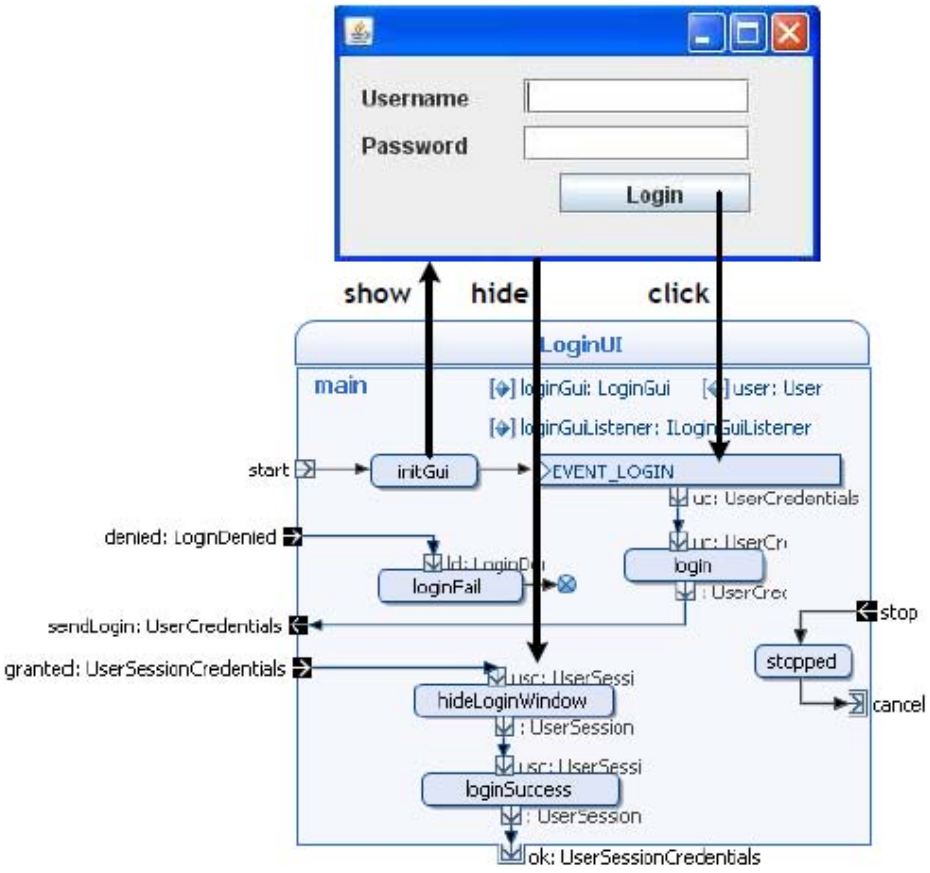

Figure 4.14: Building block for User Interface; Showing the internal behaviour of Login User Interface 'LoginUI' of figure 4.12 [Surya 2009].

graphical composition tool targeted to non-IT professionals as end-users. Whereas, Arctis requires expertise in Service Engineering. Therefore, in my opinion improvements cannot be suggested in EasyComposer when compared with 'Arctis' in this regard.

4.7.2 Control Logic

In this section, comparison between EasyComposer and Arctis is made with respect to control logic. As far as control logic is concerned, the following four areas have the probable room for improvement in EasyComposer.

Initial node: In EasyComposer the composition can be started only with "incoming call", "receiving SMS" or "receiving MMS". Hence, no service can be composed bypassing these

components. However in Arctis, the user can select any of the services as the starting node/initial node in the service composition.

Fork node: In Arctis the control/object flow can be duplicated external to the service components. Same data can reach more than one service components from a single output pin by the help of fork nodes. In EasyComposer, to achieve the same results we need to have as many output pins as number of service components to be served. Therefore, in EasyComposer the data flow is more static.

Decisions on data: In Arctis we make decisions external to the service components. These decisions are based on the contents of the data. In EasyComposer decision is made inside the service components.

Timer: In Arctis, there is a 'Timer' which can be used if repeated behaviour of an activity is required. EasyComposer does not have this functionality.

4.8 XML representation

The objective of this task is the platform-agnostic generic representation of the services and service compositions that will be used to realize them. This section is based on the work done by former student Jens Einar in his Master's thesis [Jens 2009]. It was decided to define the services with XML representation to make them more general. I built this representation on Jens Einar's work and tried to improve it.

4.8.1 Service Component XML representation

First, the existing model is described as extract from work done by [Jens 2009] and then the improvements made are explained in upgraded model.

Existing Model [Jens 2009]

Component is used to represent each service used. A component has the following mandatory attributes: *Id* of the component, *Type* of the component. To be able to connect the services to each other every component also needs an attribute telling which component is the next in this behaviour: *nextId*. This attribute can be omitted if this service is the last in the composition

behaviour branch. Some services depend on external services and therefore could be prone to errors. These services need a *nextIdIfError* attribute as well. *Type* is the type of the service, telling which of the services described in 4.2 is being used here. Most of the services have one or more settings that need to be described. These settings can be any thing from a link to a sound-file or a telephone number or phone id. Hence the component would look like figure 4.15.

```
<Component id="unique_id" type="example_service" [component specific attributes]>
Service Specific Content
</Component>
```

Figure 4.15: Component XML representation [Jens 2009]

Each service has its own component type. The type of the component dictates which attributes should be used. For many of the services it is sufficient with this element. But in some cases, there is a need for list of either numbers, times or locations, etc. These cannot be described by attributes in a proper manner. 'Entry' can be used for this. An entry has the following mandatory attributes: *Type* and *Next Component Id*. The *Type* tells how to interpret the entry and also dictates which attributes are needed. Figure 4.16 shows a basic entry common for all types of entries.

```
<Entry type="setting_type" [entry specific attributes] nextId="next_component_id"/>
```

Figure 4.16: Component Entry [Jens 2009]

Upgraded Model[2]

As discussed in section 4.7, the composition behaviour could be started only with some specific service components in existing model. Hence, no service could be composed bypassing these components. To make the service compositions more flexible and to improve the generality of the service components, another mandatory attribute *initialNodeOption* is added to the component. This mandatory attribute introduces the option to start service compositions from any of the service components. It is achieved by selecting the component as *initialNode*. This component will then be the first in the behaviour branch. Hence, the new XML representation of component which is common for all types of components will look like figure 4.17.

[2] XML representation tables of all the services in section 4.2 are explained in Appendix A.

```
<Component id="unique_id" type="example_service"
  initialNodeOption=”first_node_option” [component specific attributes]>
  < ! -- Service Specific Content -- >
</Component>
```

Figure 4.17: Upgraded ‘Component’ XML representation

In the existing model, entry has *type* and *nextId* as mandatory attributes. In the upgraded model, *nextId* is replaced by *nextIdOption.* The difference is; by using *nextIdOption* attribute, the component can be selected either as terminating node or as continuation node in the behaviour branch. If *nextId* is chosen in *nextIdOption,* id of the next component is to be specified in the *nextId* attribute. If the component is selected as the terminating node in the behaviour then *nextId* attribute does not need to be specified. The modified XML representation of basic entry which is common for all types of entries is presented in figure 4.18.

```
<Entry type="setting_type" [entry specific attributes]
  nextIdOption=”next_component_id_option”/>
```

Figure 4.18: Upgraded ‘Component Entry’ XML representation

XML example of a service

Example of ‘Send MMS’ is explained. Two versions of the ‘Send MMS’ service are described. In the first version shown in figure 4.19, ‘Send MMS’ service is the last component in the behaviour branch as specified by the attribute *nextIdOption*=“none”. It takes *text* string as the text to be sent along the MMS. In the second version shown in figure 4.20 ‘Send MMS’ is not the last component in the behaviour branch as it is specifying the *nextId*=“3” which is the component *id* of the next service after MMS is sent. This version is taking *textParam*[3] as *textSource* from the preceding service whose component *id* is 1.

```
<Component id="2" type="sendMMS" initialNodeOption=”none” number=” 45770186”
  textSource=”text” text=”Happy Birthday” fileURL=”Location of file”
  nextIdOption=”none”>
</Component>
```

Figure 4.19: 'Send MMS' XML example as last component in the behaviour branch

[3] Explanation of this attribute and other attributes of ‘Send MMS” are given in Appendix A

```
<Component id="2" type="sendMMS" initialNodeOption=”none” number=” 45770186”
  textSource=”textParam” textParam=”1” fileURL=”Location of file”
  nextIdOption=”nextId” nextId=”3">
</Component>
  …….
```

Figure 4.20: 'Send MMS' XML example as one of the middle components in the behaviour branch

4.8.2 Service Composition XML representation

This section describes an example of service composition using the XML representation of service components and entries discussed in 4.8.1. Figure 4.21 shows an XML representation of service composition previously described in section 4.5.4.

```
<Service Name=“AutomaticSmsMmsMulticast”>
   <Component id="0" type="receiveSMS" initialNodeOption=“initialNode”
    nextIdOption=“nextId ” nextId= “5”>
   </Component>
   <Component id="5" type="messageBlockFilter" initialNodeOption=“none”
    nextIdOption=“nextId” nextId= “4”>
      <Entry type= “ messageFilterEntry” phoneToMatch= “1234567” textToMatch= “abc”
       anonymousPhoneNumber=“True” advertisements=“True” dialCode= “123”
       nextIdOption=“none”>
      </Entry>
   </Component>
   <Component id="4" type="multicastSmsMms" textSource=“textParam”
    textParam=“0” initialNodeOption=“none” nextIdOption=“none”>
     <Entry type=“multicastEntry” contactGroup=“none” sendToAllFriends=“False”
      sendToAllFamily=“False” sendToAllCoworkers=“True” nextIdOption=“none”>
     <\Entry>
   <\Component>
<\Service>
```

Figure 4.21: 'Automatic SMS/MMS Multicast' Composition XML example

Chapter 5: Practical Issues

This chapter discusses some of the major practical issues related to the area of end-user service composition.

5.1 Turning the tables?

We have looked into the possibilities for end-user service composition. It can be undoubtedly inferred that the subject matter of end-user service composition has turned the tables. Now, end-users can create what they want without taking assistance from service operator. This is a major breakthrough.

The next question that promptly strikes the brain is; what is the future of service operator and providers? This is one of the most significant practical issues that needs to be addressed. A service subscriber is the main revenue generating unit for a company. To remain competitive, a company always needs to increase its revenue generating units over time. On the other hand, the real essence of end-user service composition lies in giving full control to the end-user to compose services so that their compositions remain logical and can fulfill their requirements. In short, end-user service composition can cause a setback to these business companies.

5.2 Solution

Giving full control to the end-user is not pragmatic enough. One direction to solve this issue is to think that what level of control should be given to the end-user to compose a service so that the composition remains logical to end-user and at the same time end-user can feel the essence of service composition.

In my opinion, defining a time limit for which the user can have the full control over his composed service, can serve the purpose. This time limit can vary as per end-user's needs and requirements. For example, after composing the service, user may need to specify the time duration for which he is going to use his composed service. According to this time duration,

the end-user can be charged a nominal fee which is the source of revenue for the service operators as well.

Besides, [Judith 2009] addresses the practical issue of rapid development and dynamic deployment of services in the distributed service environment. It presents a policy-driven methodology and approach for modeling dynamic composition of services. This concept is demonstrated for dynamic access control.

Chapter 6: Conclusion

This chapter describes the results achieved in this project work. It discusses whether the obtained results were expected or not. The last section of this chapter proposes some future work.

6.1 Achievements

Possibilities of end-user service composition have been investigated successfully. A set of self contained services is defined. These services are then used to create a number of composition scenarios for intelligent travel guide with several situations and different settings.

All the services are not merely defined on paper. Services related to Telecom are then implemented in EasyComposer as well. These services, together with the existing services of EasyComposer, are then used to create not only the composition scenarios for intelligent travel guide but are also successfully utilized to demonstrate Doctor's Appointment Scenario. Demonstration of Doctor's Appointment Scenario in EasyComposer further strengthens the investigation of end-user service composition and proved the modularity and atomic nature of the services defined. XML representation of services and service composition served well to realize them. Moreover, an effort made to compare EasyComposer with 'Arctis' helped in the analysis of further improvements in EasyComposer.

6.2 Discussion

End-user service composition has been investigated by using scenario driven approach. Scenarios have been created keeping the focus on Intelligent Travel Guide, but the services used to create them can be used for creation of scenarios in other contexts with different situations.

Scenario driven approach did not only prove to be helpful to investigate the possibilities of end-user service composition but it also facilitated to explain and discuss these scenarios with friends and family, who are the end-users. Moreover, intuitiveness of EasyComposer made this task further easier for me. End-users' curiosity, their inputs to identify the services and

response to the scenarios, revealed that end-users really need similar service compositions which can fulfill their needs and adapt to their personal interests of daily life. Although they seem to be quite interested but the most important issue which I came across with is, that end-users lack the self confidence to experiment new service compositions. I did not work on this aspect but in my opinion, it would have been a good exercise to look for the stratagems to remove the information gap, to convince end-users, to build their self confidence and to get them interested in end-user service composition.

While working out for scenarios in EasyComposer, it has been learnt that this interface has great potential to be improved specifically in its control logic. Moreover, the comparison of this tool with other tools available for service composition helped in further analysis of its enhancement.

Finally, an effort has been made to define the services in a generic way using XML representation. I built on the work already done by Jens Einar in his Master thesis. I tried and improved it to some extent by comparing our service definitions with other service composition tools like Arctis. Nevertheless, more work can be done in this area by exploring other possible ways to improve the generic definitions of the services.

Due to the time limitation, it was not possible to experiment the scenarios in real environment but this project can progress further in this respect. In my opinion, the effort made in this project to meet the challenges, is one step towards further development.

6.3 Future Work

Based on the results achieved in this project work future development can be done to experiment the composition in real environment.

The control logic of EasyComposer, interface used for demonstrating the service compositions, can be improved. Addition of more optional inputs and outputs in service definitions can further improve the generality of service definitions in EasyComposer.

Several interfaces can be thought and developed depending upon the handheld devices. This will address the issue of user interfaces for small devices as well.

Incorporation of the concept of context-awareness in future work can enhance the versatility of this project. Our service compositions will then be able to sense their physical environment, i.e., their context of use, and adapt their behavior accordingly. Such systems are a component of a ubiquitous computing.

End-users are not trained in securing the service compositions they created. It can lead to serious security issues if their personal confidential information passes to the wrong brains. Future work can address this issue specifically if context-awareness is incorporated.

Further possibilities and solution should be worked out to address the practical issues, related to the freedom of control given to the end-user, mentioned in this project work.

Last but not the least; investigation of the strategies to attract end-users towards service composition could be done. It should be explored that how can we build the self-confidence of the end-users to compose services for themselves. This exercise together with end-user survey will lead to better service definitions.

References

[Evren 2003] Evren Sirin, James Hendler, and Bijan Parsia; *Semi-automatic Composition of Web Services using Semantic Descriptions: modeling, architecture and infrastructure workshop in International Conference on Enterprise Information Systems (ICEIS 2003)*, April 2003.

[Jens 2009] Jens Einar; *Composing end-user services.* Master thesis, Norwegian University of Science and Technology, June 2009.

[Eirik 2009] Eirik Blakstad; *City Guide – Corporate Group Scenario.* Master thesis, Norwegian University of Science and Technology, June 2009.

[Poslad 2009] Stefan Poslad; *Ubiquitous Computing: Smart Devices, Environments and Interactions.* John Wiley & Sons Ltd, ISBN: 978-0-470-03560-33, 2009.

[Liu 2007] Xuanzhe Liu, Gang Huang, Hong Mei; *Towards end user service composition, in Proceedings of 31st International Computer Software and Application Conference (COMPSAC 2007*, IEEE Computer Society, 2007.

[UbiCompForAll 2010] The UbiCompForAll research project, *UbiCompForAll – Ubiquitous service Composition For All users*, http://www.ubicompforall.org, *accessed January 10, 2010.*

[Michael 2005] Michael Bolin; *End-user programming for the web.* Master thesis, Massachusetts Institute of Technology, May 2005.

[Stav 2006] E. Stav; *An Approach to Developing Extensible Application Composition Environments for End Users.* Doctoral thesis, Norwegian University of Science and Technology, 2006.

[DR-AP] The UbiCompForAll research project; *Doctor's appointment, UbiCompForAll – Ubiquitous service Composition For All users / Results / Scenarios,* http://www.sintef.no/Projectweb/UbiCompForAll/Results/Scenarios, accessed January 10, 2010.

[Judith 2009] Judith E. Y. Rossebo; *Dynamic composition of services – a model-based approach.* Doctoral thesis, Norwegian University of Science and Technology, September 2009.

[Frank 2009] Frank Alexander Kraemer, Vidar Slatten and Peter Herrmann; *Tool Support for the rapid composition, analysis and implementation of reactive services*, in *The Journal of Systems and Software,* 2009.

[Surya 2009] Frank Alexander Kraemer, Surya Bahadur Kathayat and Rolv Braek; *Unified Modeling of User Interfaces and Service Logic, in Proceedings of the First International Workshop on Model Driven Service Engineering and Data Quality and Security (MoSE+ DQS'09)*, Hong Kong, China, 2009, ACM.

[Rolv 2009] Frank Alexander Kraemer, Rolv Braek and Peter Herrmann; *Compositional Service Engineering with Arctis: Telektronikk 1*, 2009.

[Arctis 2010] Arctis Reference; http://arctis.item.ntnu.no/doc/building_blocks, accessed January 10, 2010.

Appendix A: XML Representation Tables

A.1 XML Services

ComponentType	sendEmail
Entries	None
initialNodeOption	[none, initialNode]
textSource	[none,textParam,text]
subjectSource	[subjectParam, subjectText]
file	[none, fileURL]
cCopy	name@server.com
bCopy	name@server.com
emailAddress	name@server.com
initialNode	Start the composition from this service
textParam	Service Id of the preceding service providing the text string
text	Text String
subjectParam	Service Id of the preceding service providing the subject text string
subjectText	Subject text string
fileURL	Location of the file to be attached
nextIdOption	[none, nextId]
nextId	Service Id of the next service after the E-mail is sent

Table A.1: 'Send Email' Service Attributes

ComponentType	sendSMS
Entries	None
initialNodeOption	[none, initialNode]
number	Contact number to which SMS is to be sent
textSource	[textParam,text]
initialNode	Start the composition from this service
textParam	Service Id of the preceding service providing the text string
text	Text string
nextIdOption	[none, nextId]
nextId	Service Id of the next service after the SMS is sent

Table A.2: 'Send SMS' Service Attributes

ComponentType	sendMMS
Entries	None
initialNodeOption	[none, initialNode]
number	Contact number to which MMS is to be sent
textSource	[none,textParam,text]
fileURL	Location of the file to be attached
initialNode	Start the composition from this service
textParam	Service Id of the preceding service providing the text string
text	Text String
nextIdOption	[none, nextId]
nextId	Service Id of the next service after the MMS is sent

Table A.3: 'Send MMS' Service Attributes

ComponentType	multicastSmsMms
Entries	multicastEntry
initialNodeOption	[none, initialNode]
textSource	[none,textParam,text]
file	[none, fileURL]
initialNode	Start the composition from this service
fileURL	Location of the file to be attached
textParam	Service Id of the preceding service providing the text string
text	Text String
nextIdOption	[none, nextId]
nextId	Service Id of the next service after the MMS or SMS is sent

Table A.4: 'SMS/MMS Multicast' Service Attributes

ComponentType	messageBlockFilter
Entries	messageFilterEntry
initialNodeOption	[none, initialNode]
initialNode	Start the composition from this service
nextIdOption	[none, nextId]
nextId	Service Id of the next service when no match is found

Table A.5: 'Message Block Filter" Service Attributes

ComponentType	messageAutoForward
Entries	no
initialNodeOption	[none, initialNode]
phoneNumber	[none, number]
emailAddress	[none, emailAddress]
initialNode	Start the composition from this service
number	Contact number to which the message is to be forwarded
emailAddress	name@server.com
nextId	Service Id of the next service after the message is forwarded

Table A.6: 'Message Auto-Forward' Service Attributes

ComponentType	instantMessage
Entries	contactEntry
initialNodeOption	[none, initialNode]
initialNode	Start the composition from this service
nextIdOption	[none, nextId]
nextId	Service Id of the next service after the chat is initiated

Table A.7: 'Instant Message' Service Attributes

ComponentType	findLocation
Entries	locationEntry
initialNodeOption	[none, initialNode]
initialNode	Start the composition from this service
map	Map guide of the selected location
nextIdOption	[none, nextId]
nextId	Service Id of the next service after the location is found

Table A.8: 'Find Location' Service Attributes

ComponentType	findTransportation
Entries	transportEntry
initialNodeOption	[none, initialNode]
initialNode	Start the composition from this service
nextIdOption	[none, nextId]
nextId	Service Id of the next service after the transport is found

Table A.9: 'Find Transportation' Service Attributes

ComponentType	weatherForecast
Entries	weatherEntry
initialNodeOption	[none, initialNode]
initialNode	Start the composition from this service
nextIdOption	[none, nextId]
nextId	Service Id of the next service after the weather alert is sent

Table A.10: 'Weather Forecast' Service Attributes

ComponentType	generalInformationOfTheCity
Entries	generalInfoEntry
initialNodeOption	[none, initialNode]
initialNode	Start the composition from this service
nextIdOption	[none, nextId]
nextId	Service Id of the next service after the information is found

Table A.11: 'General Information of the City" Service Attributes

A.2 XML Service Entries

contactGroup	[none, contactNumbers]
sendToAllFriends	True/false
sendToAllFamily	True/false
sendToAllCoworkers	True/false
contactNumbers	Contact numbers to which message is to be sent

Table A.12: ' multicastEntry' Attributes

phoneToMatch	The contact numbers to match
emailToMatch	E-mail addresses to match
textToMatch	Text string to match
anonymousPhoneNumber	True/false
advertisements	True/false
dialCode	Country or city dialing code to match
domainName	Domain name to match
nextIdOption	[none, nextId]
nextId	Service Id of the next service when match is found

Table A.13: 'messageFilterEntry' Attributes

contacts	[emailAddress, phoneNumber]
message	Instant message to send
startConversation	Initiate conversation between contacts
phoneNumber	Phone numbers to be added in the contact list/friend list
emailAddress	Email address to be added in the contact list/friend list

Table A.14: 'contactEntry' Attributes

locationToFind	[restaurant, hotel, shoppingCentre, architecture, addressToMatch]
distanceRange	Distance within which location has to be situated
timeToReach	Time limitations to reach a place
moneyRange	Limits of amount of money preferred
foodToMatch	Preferred food to be matched with restaurants
favouriteShopping	Preferred kind of shopping to be matched

Table A.15: 'locationEntry' Attributes

transportChoice	[bus, taxi, rentACar, rentABike]
distanceRange	Distance limits within which transport is to be found

Table A.16: 'transportEntry' Attributes

forecastDuration	Duration for which weather forecast is needed
alertSeverityLevel	[snowfall, rain, storm]
timeToReach	Time preference to reach destination
meansOfTransport	[onFoot, byBus, byCar]

Table A.17: 'weatherEntry' Attributes

generalInfoRequired	[currencyRate, generalRules, trafficRules, nationalHolidays, specialOccasions, banItems, emergencyNumbers]
occasionAlert	Identifier of the special occasion for which alert is to be received

Table A.18: 'generalInfoEntry' Attributes