INF-3981

Master Thesis in Computer Science

Design and implementation of an Encourager and Motivator Application of Physical Activity based on Bluetooth devices and Argos Middleware Platform

Harald S. Hanssen

27th of January, 2008

Faculty of Science

Department of Computer Science

University of Tromsø, N-9037 Tromsø

Abstract

Argos is a middleware platform developed at the ArticBeans lab at the University of Tromsø. The purpose of Argos is to provide a personal application platform for custom services. In this thesis we look at how Bluetooth devices along with Argos can be combined to create an application for encouraging and motivating physical activity.

Physical activity is steadily decreasing among the population. People are getting more and more unhealthy due to lack of exercise. Playing computer games or watching television is rather normal today, going outside for a walk is something people seldom do.

If a person wants to break the habit of sitting still, he will struggle with breaking the habit and committing to the new lifestyle. A person might have a higher chance of success, if he has something to remind him to exercise at a certain time of the day. The most common reason for failing to break a habit, can be due to lack of reinforcement of the new lifestyle. Having a personal application which allows the person to control his exercise, and also lets him see how much he has achieved, might prove both motivating and encouraging to continue to be more active.

Acknowledgements

I would like to thank my supervisors Arne Munch-Ellingsen and Gunnar Hartvigsen for the their inputs and discussions.

The people back at the NST lab at Telenor, for their input and help. Telenor for lending me the units I required. Also thanks to Gunnvald Bendix Svendsen at Telenor for taking time to have a discussion with me.

Also my family, Torgunn-Bente Sømnes Hanssen, Inge Borgan Hanssen, Valborg Borgan Hanssen, Brynmor Evans and Nikolai-Johan Borgan Evans for all of their love, support and also letting me borrow the car while I worked on the thesis.

Contents

Contents **1**

List of Figures **3**

List of Tables **5**

1 Introduction **7**
1.1 Background 7
1.2 Problem Definition 8
1.3 Interpretation 8
1.4 Limitations 9
1.5 Method and Approach 9

2 Related work **11**
2.1 Behavior Change Mechanism (BCM) 11
2.2 Discussion of related work 12

3 Requirements **15**
3.1 Server side - Argos 16
3.2 Client side - mobile unit 16
3.3 Requirement tables 17
3.3.1 Naming policy 17
3.3.2 Argos and Participants (AP-x) 17
3.3.3 Argos and Missions (AM-x) 19
3.3.4 Argos and Bluetooth Beacons (AB-x) 20
3.3.5 Client and Argos (CA-x) 22
3.3.6 Client and Missions (CM-x) 22
3.3.7 Client and Beacons (CB-x) 24
3.4 Non-functional requirements 24

4 Technology **27**
4.1 Argos 27
4.1.1 Core Components 27
4.1.1.1 Derby 27
4.1.1.2 Jetty 28

4.1.1.3 JMX 29
4.1.1.4 Hibernate 30
4.2 Argus 30
4.3 Bluetooth 30
4.3.1 Layers and Protocols 31
4.3.2 Languages with Bluetooth support 32
4.4 RFID 33
4.5 JME 34
4.6 Microsoft .Net Compact Framwork 34
4.7 Discussion 35
4.7.1 RFID versus Bluetooth 35
4.7.2 JME versus .Net Compact Frameworks 35

5 Design and Implementation 37
5.1 Platform choices 37
5.2 System Overview 38
5.3 Argos 38
5.3.1 The Argos component 38
5.3.1.1 Overview of the Participant class 39
5.3.1.2 Overview of the Mission class 39
5.3.1.3 Overview of the DBComponent class 39
5.4 The Mobile Unit 39
5.5 Bluetooth Beacon 41

6 Evaluation 43
6.1 Method 43
6.2 Functional requirements 45
6.3 Non-functional requirements 45

7 Conclusion 47
7.1 Achievements 47
7.2 Future work 48
7.2.1 Fully functional hibernate service 48
7.2.2 Argos and the mobile unit 48
7.2.3 Web interface 48
7.3 Conclusion 48

Bibliography 49

List of Figures

3.1 Overview of the system in use 15
3.2 Basic view of the interaction between the Mobile Unit and Argos . . 16
3.3 Interaction between the participant and mobile unit 17
3.4 Argos operations on the Participant 18
3.5 Argos operations on Missions 19
3.6 Beacon 21
3.7 Client and Mission requirements 22

4.1 Overview of Argos[6] 28
4.2 XML file describing the custom component 29
4.3 Argus, the JMX browser 29
4.4 Controller of a Bluetooth Device 31
4.5 The Bluetooth protocol hierarchy 31

5.1 System Overview 38
5.2 The Tag menu for the mobile client 40
5.3 The current mission details downloaded from Argos 41

List of Tables

3.1 Requirement AP-1: Add Participant 18
3.2 Requirement AP-2: Remove Participant 18
3.3 Requirement AM-1: Add Mission 19
3.4 Requirement AM-2: Send Mission 20
3.5 Requirement AM-3: Receive Completed Mission 20
3.6 Requirement AB-1: Add Bluetooth Beacon 21
3.7 Requirement AB-2: Remove Bluetooth Beacon 21
3.8 Requirement CA-1: Receive Data from Argos 22
3.9 Requirement CA-2: Send Data to Argos 22
3.10 Requirement CM-1: Receive Mission 23
3.11 Requirement CM-2: Send Completed Mission 23
3.12 Requirement CM-3: Receive Notification of Mission Completed . . . 23
3.13 Requirement CB-1: Detect Beacon associated with Mission 24
3.14 Requirement NG-1: Graphical User Inteface 24
3.15 Requirement NFR-2: Webservice 25
3.16 Requirement NFR-3: Motivate Participant 25

6.1 Evaluation of the Argos component 45
6.2 Evaluation of the client device 45
6.3 Non-functional requirements 46

Chapter 1

Introduction

1.1 Background

Technology surrounds people at all times, it is used to communicate, aid, increase effiency as well as entertain. Technology has its bad sides too, technology can make people inactive and too dependant on it. Once technology is a part of someones life, the struggle between technology as a tool or as a dependency has already started. Trivial things as surfing the web, checking email or even texting on the phone suddenly can become a necessesity or even a compulsion. People feels useless and even unsafe if they do not have their gadgets with them.

To change a habbit is difficult, if it was easy anyone could do it anytime. Habbits can comes over time. However to quit a habit, takes considerable longer time and effort. For example a person who tries to quit smoking, will most likely fail, due to either motivation, effort or the environment. One theory looking at the problem of changing habits or lifestyle, is called Behavioral Change Mechanism (BCM). BCM tries to identify the steps for any decision, as well as locating the best means to accomplish a lifestyle change.

BCM are in many cases applied for a medical purpose, the goal is often to change unhealthy lifestyles. The current lifestyle of a person might be life threatning, because of inactivity, smoking, alcohol addiction and so on. The purpose of the BCM is to both push the user into a better lifestyle as well as motivating along the way. The motivation to change varies alot from person to person, but in most cases the person undergoing a lifestyle change will struggle. If technology can contribute to keep the motivation stabile for any person, it will be easier to start or even maintain a new lifestyle for anyone that undergoes a similar adjustment.

The software created in this thesis, is based on the concept of BCM. The software introduces an environment, where the user engages in an activity to become more active. The software reminds the user when it is time to be more physical by giving him tasks or missions. The main goal of the software is to give the user a active life.

1.2 Problem Definition

The goal of this thesis is to create a software system that will promote physical activity to its users. The software purpose is both to get the users in activty as well as monitoring their progress. The user is from this point referred as an participant.

The software system will give the participant missions, which can be given daily, weekly or at some other rate. The missions are created by the participant himself. The missions must be completed within a certain amount of time. A mission consists of a waypoint, which the participant has to go to and be registered at. The mission is completed when the participant has registered the correct waypoint.

Once the participant engages in the physical activity, the participant will receive a mission along with a description of where the location is, and a timelimit to complete the assignment, where everything is defined by the participant himself. As soon as the participant reaches a waypoint, the waypoint is registered and the mission is completed. The next mission is given after the specification of the participant. The participant can setup a mission, which he will receive at some point during the week. The participant will not get another mission right after he has completed the previous one, unless he has specified so. The amount of missions available during the week is set by the participant.

Once a participant has completed a mission, the data that has been collected will be processed. The data received from the participant, will be processed to find out wheter the goals of the mission were completed. Since this project is purely "proof of concept", the amount of data gathered is limited to waypoints that have been visited.

1.3 Interpretation

The problem definition introduces a software system which consists of several parts, both mobile and stationary. Due to the timeline, the system will be developed as an prototype.

The outline of the software:

- There is a central server handing out missions defined by the participant, and mobile units are used to complete the assignments.

- The stationary unit or the server side, prosesses and stores the data received from the mobile units whenever a participant has completed a mission. The server also contains services to create and view missions.

The server side application can be based on a middleware platform, where the middleware provides services, such as persistence, communication and user interface. Argos is an personal middleware platform to support all of the services as mentioned. Argos has been implemented with several services by default. Argos can use custom made components for a specified service.

1.4 Limitations

Due to the timelimit, the service created will be limited to basic features. The idea is to have a basic platform, where more data can be collected. The basic data that will be collected, is limited to the amount of waypoints a user has reached.

1.5 Method and Approach

This project will use the agile method called Scrum during the software development. Scrum is a lightweight agile method which emphasizes on small crossfunctional teams. The reason why small teams are preferred, is due to their bigger chance of achieving success, compared to bigger and more complex teams. Scrum was created in an effort to cope with evergrowing changes in the requirements, during development. The waterfall method is the biggest contrast of Scrum, because the watefall method does not allow any changes, while Scrum embrazes all changes coming from user demands or developer needs.

Scrum prefers to have an face-to-face based communication over written documents. There are 3 types of participants in a scrum project; the scrum team, scrum master and the users / stakeholders. The scrum team creates the product, the scrum master manages it and the users envisions it. One good example of the face-to-face communiction is the daily scrum, instead of sending documents to the manager of the project; he can know everything he needs to know directly:

- What have you done since last daily scrum?
- What will you do between now and the next scrum?
- What got in your way of doing work?

Any impendiments are mentioned during the meeting, rather than writing a document containing the problem, it is discussed directly. Discussing a problem is more productive, and in the end reduces the time spent on any problems that has been uncovered during the development.

As with any other agile method, there are phases. The phases consists of the planning, staging, development and release phase. Each phase, in this case called a sprint, is not complete untill the cycle is done. During each sprint, the requirements are quite likely change. This is either because the user requests new features or there are impendiments. Impendiments could be technology issues or something else the Scrumteam are hindered by. Impendiments are reckoned as something the Scrumteam cannot do anything with. Once a sprint is started, its cannot be changed. New requests and features will be added to a backlog. The backlog entry can be filled by anyone, however, the features may or may not be implemented in the final product.

The backlog contains list of work that needs to be done. The backlog will grow during the project lifecycle, when the backlog is empty, the project is regarded as finished.

Scrum can easily be combined with Extreme Programming (XP). XP can be used for projects where there are dynamic requirements. XP is similar to Scrum since it thrives on customer involvement and teamwork. The difference between XP and Scrum, is the system Scrum promotes. Scrum introduces order while XP does not. In XP all features can be implemented at any stage, in many cases this will hinder the development due to a very spread focus and too much work. XP does introduce release, will occurs in this case after a sprint.

The Scrum timeframe lasts 2 weeks, and there is one meeting each week. The Scrum Master is Arne Munch Ellingsen.

Chapter 2

Related work

In this chapter, the themes relevant to this thesis are described. The thesis particulary aims at BCM related topics.

2.1 Behavior Change Mechanism (BCM)

Behavioral change due to any reason, is difficult to acomplish for most people. If change was easy, then the self help industry[1][13] would be out of business. The self help industry aims at those who wants to change their lifestyle but do not know how. Most self help products contains statements which claims will improve the customers current lifestyle.

The self help industry gives the customer, a image of easy solutions to whatever his problem might be. Eventhough the customer has a problem which is both complex and personal, the self help industry can still claim they can solve it. In most cases the customer will not succeed with the self help product.

Health centers experiences the beginning of the new year, a flow of customers wanting to improve their health. However, two or three weeks later the motivation the customer had is gone, and the customer sits with a monthly fee for a service that is never going to be used.

Breaking a habit is just simply hard to do. For example a person might say he will start walking to work instead of taking the car. Intially the he will walk to work, then after a short period of time, relapse to driving instead. This scenario is rather the rule than the exception.

The problem is often located in keeping the motivation. Any setbacks while trying to excersise the new and improved habit, will surely kill any motivation. Shelle Rose Charvet, in an article at selfgrowth.com[3] mentions most pitfalls at behavioral change. She mentions crisis as a behavioral change mechanism, where there is no other option but to actually go through with the change. Crisis occurs in all kinds of levels in the society, either on the personal, professional or on a business level. Whenever a business is experiencing a crisis, it has to change and adjust to the new reality, it is make or break. A company might have to do budget cuts, fire staff and perhaps even sell of parts of the business in order to make it out alive. A person with a lifestyle crisis, will have to do much the same. The person will have to do cuts such as the consumption of unhealthy food, smoking and drinking, aswell as increase the amount of activity.

A crisis is not a good way of solving a lifestyle problem. The reason why the crisis intially occured, is due to the perception of the problem being far far away in the future. When the problem is knocking on the doorstep, the so called "Toward Motivation Trigger" kicks in. The trigger works as an kickstart motivator, but it only works for a short period of time.

The article from the U.S. Department of Agriculture[2] has a deeper and more theoretical approach to the models of behavioral change. The article stresses out that it is important to look at the steps the person does before acting. There are a lot of factors that makes a huge impact on decisions that are made during the day. The environment, location of services (such as parks, store, bars etc), weather and so on, impacts the decisions someone makes. If a person takes the car to work instead of walking or cycling, it does not instantly indicate he is lazy. Perhaps the location of the workplace is so far away, that any other means of transportation is not feasible. Choices like foodintake and exercise is somewhat easier to do something with, however they are harder to change for a longer period of time. Vulnerability is a major motivator for lifestyle change.

The only way of maintaning the new habit, according to Shelle Rose Charvet, is to reinforce the new reality. The reinforcement can be done by making the habit visual instead of hiding it. A system that can constantly remind what is at stake, keeps the lifestyle change more constant and less dependant on whatever whims the person has at the moment.

2.2 Discussion of related work

The BCM section reveals that consistency is required, in order to keep the motivation among the participants. The challenge is to keep the same pace at the participant, making sure the he does his weekly missions. It is important the participant has something that convinces him to continue being active, especially after the "Toward Motivation Trigger" has faded. If the particpiant manages to see the system as a new habbit, an everyday thing he does much like drinking coffe in the morning, it will be easier for him to make the change. To reinforce the new reality, in the system created as part of this thesis, the day for a participant can start with setting up the mission(s) while surfing the web in the morning or the night before. The mission does not have to be completed the same day or the next day, but after the presets of the participant. By letting the participant being involved in the settings of the missions, he can already start taking the charge of his new lifestyle instead of someone else. If the participant can maintain the activity without external help, the better it is. Of course, the participant is not likely to be self reliant in the beginning. The system encourages a independency from the meddling of a supervisor or a physician over time. Having someone hanging over the participants shoulder will prove discouraging over time. It is important for the participant, to be the one to take charge of his lifestyle.

The environment the participant lives in, can prohibit the lifestyle change. The environment contains everything from family, friends, nearby services[1] as well as the living conditions. If the participant lives in the city, he will have a challenge that relates to dealing with the offers a city can provide with. A participant who lives in a village, far away from the city, will have other challenges, such as the likelyhood of being alone in dealing with the lifestyle change. He might also be traveling alot, just to participate in the project. Struggling alone can be discouraging.

Eventhough the participant over time will be on his own, it does not mean he is alone. If the system is incorporated into a bigger service to encourage physical activity, the service can get additional information regarding the activtiy level of each participant. If the service is expanded with a external activity monitor, the service can figure out from the information from the system, wheter the participant is as active he should be, or if he has reverted to his old habits. This allows the staff directly associated with the service, to determine wheter someone needs more help than others. Everyone are different, change is harder for some than others. Though the system is not meant to work as an "Big Brother"[2] towards the participant, but rather to to identify problems along the way.

It is important to know that a lifestyle change, happens in every step or decision a participant makes. If the participant has something to remind him to be active, and perhaps reward him when he reaches a goal, the better. A reward that is relevant for the participant might be the key in the long run. For example; a participant who is a football fan, might want to go Manchester United to watch a Premiere League game if he reaches a certain milestone. Another example; the participant sets a goal where he sees himself doing something he cannot do now, due to his physical condition. A system that can provide the participant with feedback, will help him figure out if he reached his goal. It is important to have something to look forward to while trying to change habits, not only the improvement of health.

[1]Foodstores, fastfood chains, computer games, internet

[2]Refers to George Orwells novel **1984**[9], where the goverment had almost complete controll over the population

Chapter 3

Requirements

In this chapter the functional requirements of the system will be presented, the non-functional requirements will be listed last. The system will consist of a server and client side, where each side has their added requirements. The server consists of Argos and the Argos system services provides. Argos is a precondition for the system.

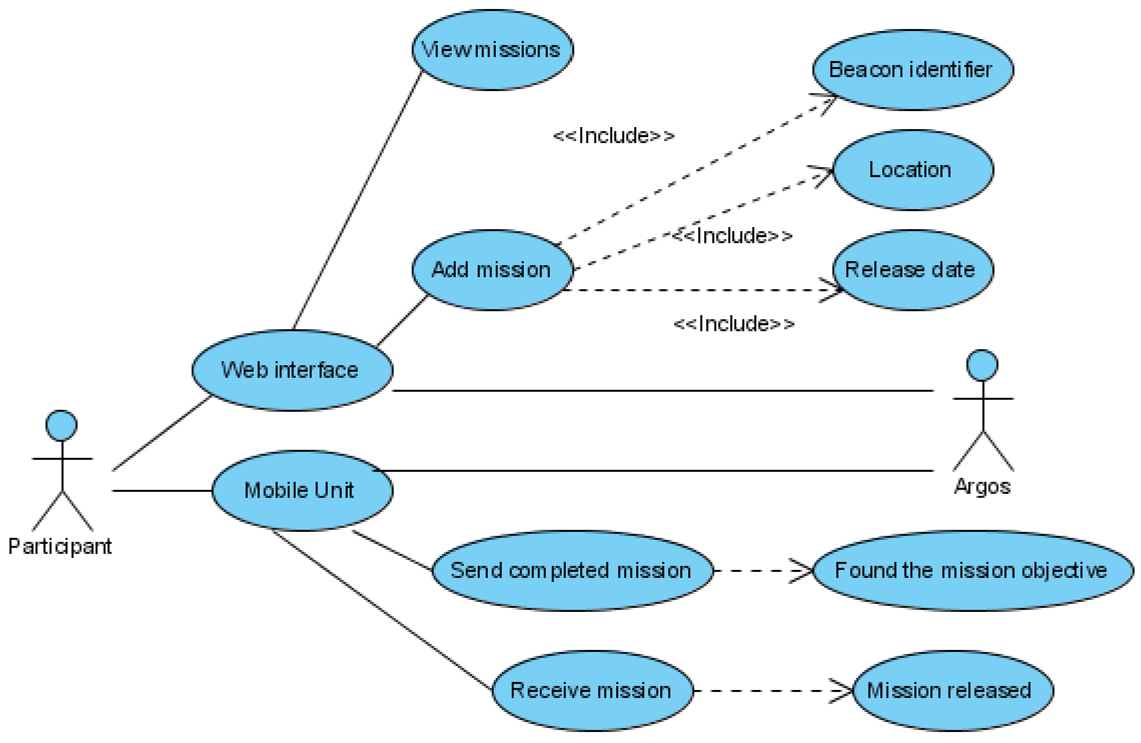


Figure 3.1: Overview of the system in use

The client side consists of the mobile device interacting with Argos as well as the Bluetooth Beacons. A Bluetooth Beacon is a device which is placed at a certain location, and can be identified by its hardware address. The purpose with a beacon, is purely to make the participant physicly move himself to where it is placed. When the participant has reached the location of where the beacon is at, he can register that he has been there.

The mobile device has to have bluetooth available and also General Packet Radio Service(GPRS) or Third Generation Cell-Phone Technoloy(3G) technology is required to reach Argos through internet access.

3.1 Server side - Argos

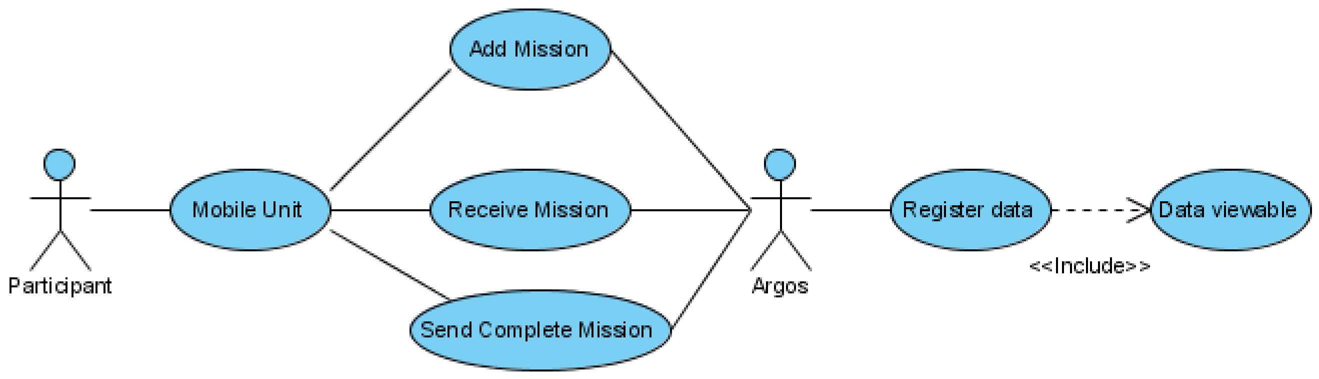


Figure 3.2: Basic view of the interaction between the Mobile Unit and Argos

Argos main task is to be a fully working server, where the it has the required services needed by the system. Argos and its components are described further in the section 4.1.

3.2 Client side - mobile unit

The client or the mobile unit, is operated by the user. The unit has to incorporate a simple Graphical User Interface (GUI). The GUI must provide the Participant with an overview of the Mission goal, and wheter the goal has been achieved.

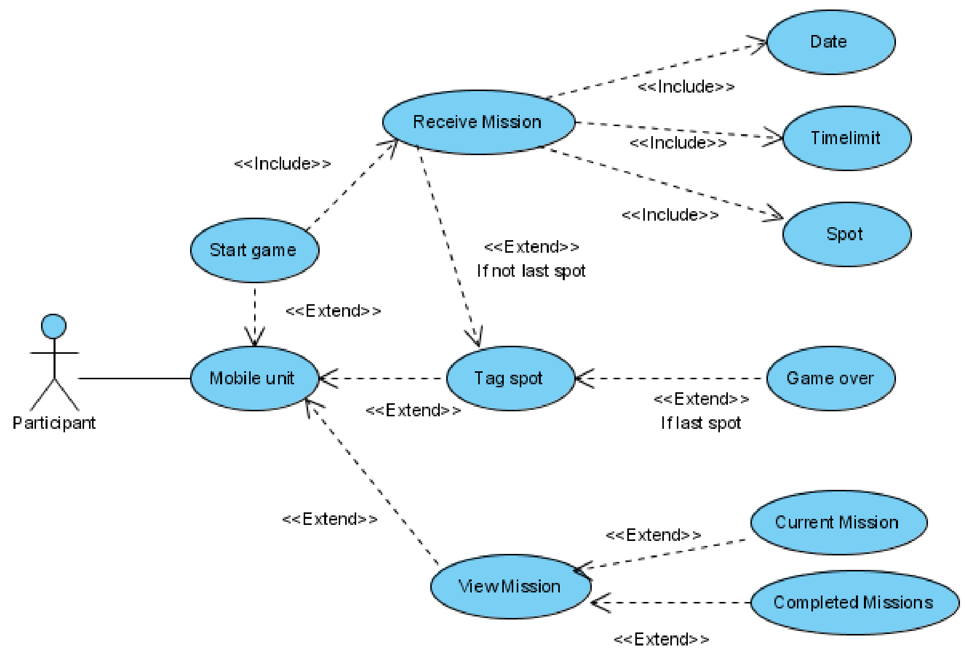


Figure 3.3: Interaction between the participant and mobile unit

The figure 3.3 shows a "game" has started whenever a participant has engaged on a mission. The game starts when the mission was specified to be sent to the Particpant, meaning the mission timer had counted down. The mobile unit does not start the mission by itself.

3.3 Requirement tables

3.3.1 Naming policy

The requirements are all given a unique identifier. The identifier starts with two or three letters indicating where the requirement belongs. For example; AP-1 refers to a requirement shared by Argos and the Participant. The numeration is used to number the requirement, the number does not indicate a priority level.

There are Argos (AP-i, AM-i, AB-i), client (CM-i and CB-i) and nonfunctional (NFR-i) requirement tables, where i goes from 1 to x.

3.3.2 Argos and Participants (AP-x)

Argos must be able to register participants details and activities. Argos stores intially the name and mobile details for the participant. The mobile details is used to send missions.

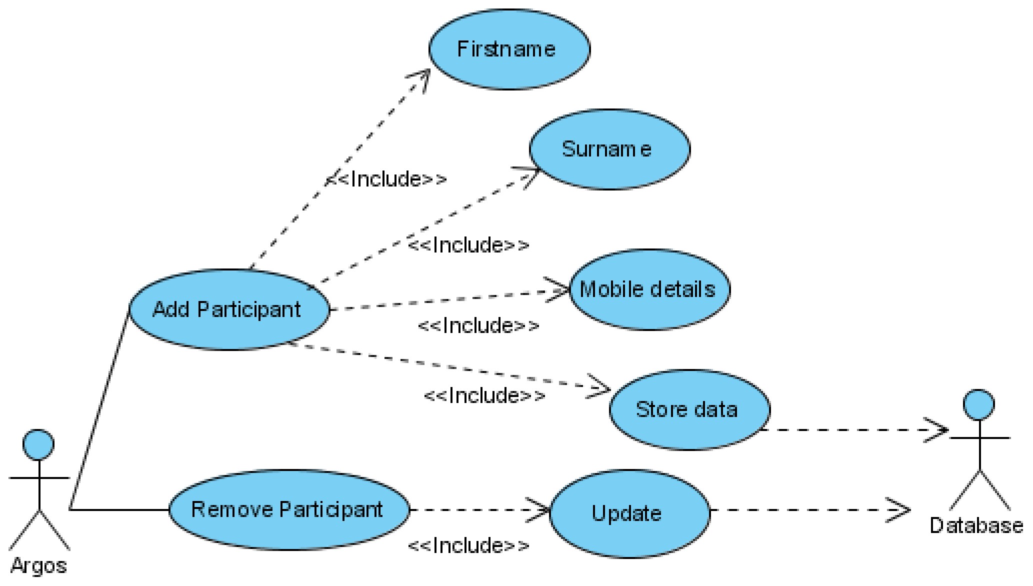


Figure 3.4: Argos operations on the Participant

Requirement ID:	$AP - 1$
Requirement Name:	Add Participant
Priorty:	High
Goal:	Add participant to database
Testing:	1. Insert partipant to database 2. Check to see that the participant is correctly stored
Description:	The database needs to be able to add participants with details. The details includes: *Firstname, Surname and Cellphone number / Mobile details*

Table 3.1: Requirement AP-1: Add Participant

Requirement ID:	$AP - 2$
Requirement Name:	Remove Participant
Priorty:	Low
Goal:	Remove participant and update database
Testing:	1. Remove the participant from database 2. Check to see that the participant was correctly removed
Description:	Removing unnecessary or redundant participants.

Table 3.2: Requirement AP-2: Remove Participant

3.3.3 Argos and Missions (AM-x)

Argos must let the participant be able to add missions, aswell as receiving them at the set times. Missions that are confirmed complete, will be stored in the database.

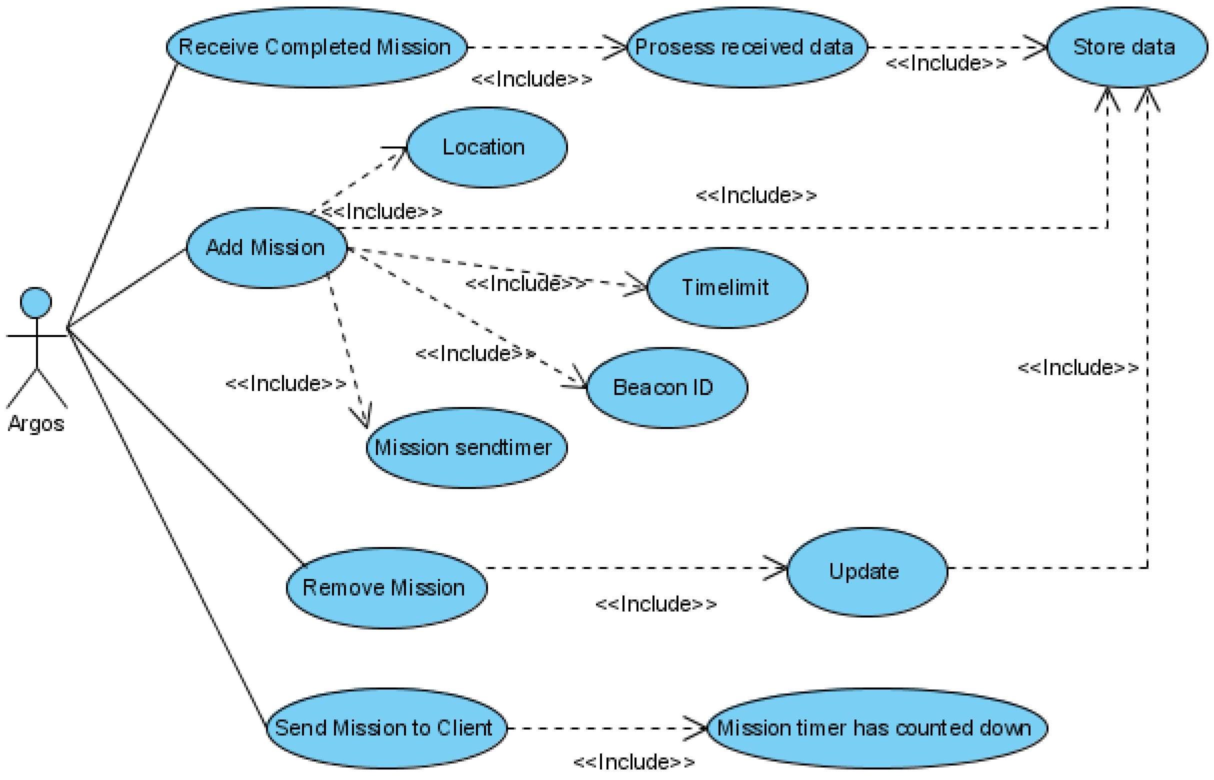


Figure 3.5: Argos operations on Missions

Requirement ID:	$AM-1$
Requirement Name:	Add Mission
Priorty:	High
Goal:	Argos must add a Mission on the request from a Participant
Testing:	1. Add mission to database 2. Check to see that the mission was correctly stored
Description:	Add mission with details regarding the goal, aswell as filling in other details to complete the mission. The mission details are specificed by the Participant (see table 3.1)

Table 3.3: Requirement AM-1: Add Mission

Requirement ID:	$AM - 2$
Requirement Name:	Send Mission
Priorty:	High
Goal:	Send mission to Participant
Testing:	1. Do a query for a mission 2. Send SMS with download details to a client 3. Wait for a link from the client 4. Transfer the mission to the client
Description:	Send mission to the mobile unit belonging to the Participant, when the mission timer has counted down. The mission timer is set by the Participant when he creates the mission.

Table 3.4: Requirement AM-2: Send Mission

Requirement ID:	$AM - 3$
Requirement Name:	Receive Completed Missssion
Priorty:	High
Goal:	Store completed mission from Participant
Testing:	1. Receive data from client 2. Check received data
Description:	Store the details from the completed mission, details can contain details about the mission requirements, such as time of completion. The completed mission is received from a mobile unit as soon as the Participant has found and taged a specific Beacon.

Table 3.5: Requirement AM-3: Receive Completed Mission

3.3.4 Argos and Bluetooth Beacons (AB-x)

To make it easier to know which beacons have been visited during a mission, the hardware adddress aswell as the location is stored in Argos. Argos can add beacons by demand. Having a list of beacons to select from, will make it easier to create missions with the selected beacon as the waypoint. There is no interaction between Argos and the beacons, other than that Argos knows they exsist.

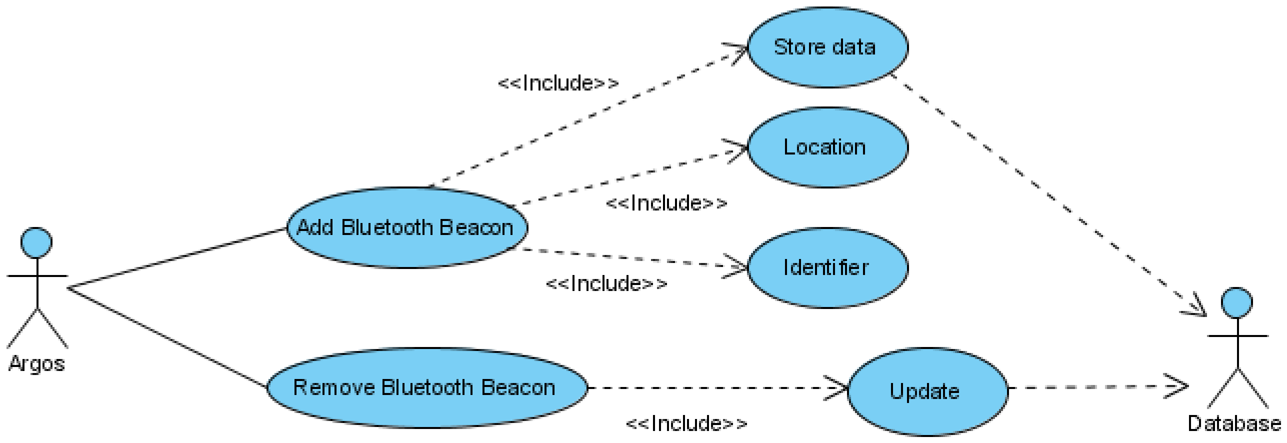


Figure 3.6: Beacon

Requirement ID:	$AB-1$
Requirement Name:	Add Bluetooth Beacon
Priority:	Low
Goal:	Creating a list of available beacons for the participant
Testing:	1. Add Bluetooth Beacon details 2. Check to see that the beacon was correctly stored
Description:	Add a Bluetooth Beacon to the know list of available spots. The Beacon is used as an mission objective.

Table 3.6: Requirement AB-1: Add Bluetooth Beacon

Requirement ID:	$AB-2$
Requirement Name:	Remove Bluetooth Beacon
Priority:	Low
Goal:	Removing unnecessary beacons
Testing:	1. Remove a specific beacon 2. Check to see if the beacon was correctly removed
Description:	Remove a Bluetooth Beacon from the know list of available spots. The Beacon is used as an mission objective.

Table 3.7: Requirement AB-2: Remove Bluetooth Beacon

3.3.5 Client and Argos (CA-x)

The Client must be able to establish contact with Argos in order to receive and send data. As mentioned the client unit must have GPRS, EDGE or 3G incorporated in order to fullfill the requirements from CA-i.

Requirement ID:	$CA-1$
Requirement Name:	Receive data from Argos
Priority:	High
Goal:	Receive details and missions
Testing:	1. Receive SMS from Argos 2. Connect to internet 3. Download data with the specific details from the SMS sent from Argos
Description:	The mobile unit is required to have internet access besides SMS in order to receive data from Argos.

Table 3.8: Requirement CA-1: Receive Data from Argos

Requirement ID:	$CA-2$
Requirement Name:	Send data to Argos
Goal:	Receive missions as well as sending completed missions to Argos
Priority:	High
Testing:	1. Connect to internet 2. Send data to Argos
Description:	The mobile unit is required to have internet access in order to send data to Argos

Table 3.9: Requirement CA-2: Send Data to Argos

3.3.6 Client and Missions (CM-x)

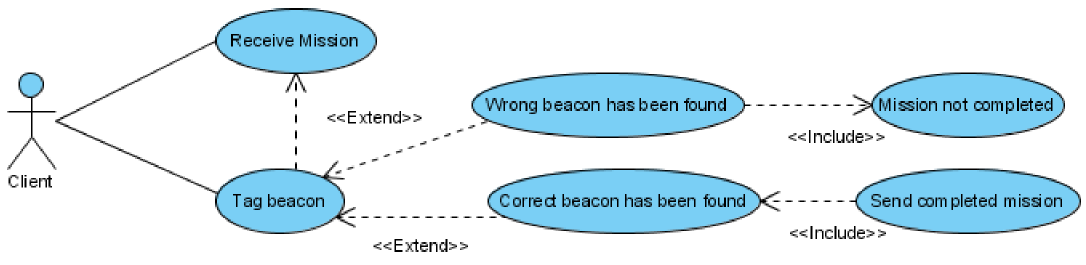


Figure 3.7: Client and Mission requirements

The client must be able to receive missions, as well as returning completed missions to Argos. The requirements in the tables 3.8 and 3.9, must be fullfilled.

Requirement ID:	$CM-1$
Requirement Name:	Receive Mission
Priority:	High
Goal:	Activating the participant
Testing:	1. Receive SMS from Argos 2. Connect to Argos with the SMS details 3. Receive mission from Argos by using the data in the SMS
Description:	The mobile unit receives a notification that there is a new serie of missions available. The unit will then automaticly download the mission based on the settings in the notification.

Table 3.10: Requirement CM-1: Receive Mission

Requirement ID:	$CM-2$
Requirement Name:	Send Completed Mission
Priority:	High
Goal:	Notify Argos that a mission has been completed
Testing:	1. Send the completed misssion by using the details in the received SMS. 2. Check Argos database to see if the completed mission has been received
Description:	When the client has completed a mission, the client has to send the details back to the sender, in this case the Argos service. The data details will be stored persistently.

Table 3.11: Requirement CM-2: Send Completed Mission

Requirement ID:	$CM-3$
Requirement Name:	Receive Notification of Mission Completed
Priority:	Low
Goal:	Ensuring the participant that the mission has been received by Argos
Testing:	
Description:	Argos has received the completed mission.

Table 3.12: Requirement CM-3: Receive Notification of Mission Completed

3.3.7 Client and Beacons (CB-x)

The client must be able to locate the beacons by doing an SDP query near the beacon. The beacon has to be named in a specific manner in order to be detected, since the client does not search for a specific service. The name of the beacon is set by default and is not something the Participant has to deal with. The reason why the client is searching for a name, is for the sake of simplicity.

Requirement ID:	$CB-1$
Requirement Name:	Detect Beacons associated with Mission
Priority:	High
Goal:	Completing mission
Testing:	1. Do a SDP search near a beacon 2. Compare mission beacon with the present beacon 3. Send mission complete to Argos if the beacon is equal to the beacon in the mission
Description:	The Mission (table 3.4) received from Argos contains details regarding the Beacon which has to be spotted. Unless the beacon has been spotted, the Mission will stay incomplete.

Table 3.13: Requirement CB-1: Detect Beacon associated with Mission

3.4 Non-functional requirements

Requirement ID:	$NFR-1$
Requirement Name:	Graphical User Interface
Priority:	High
Goal:	Making it intutive for the participant to interact with the system
Testing:	
Description:	The mobile unit needs to have an easy to use graphical user interface.

Table 3.14: Requirement NG-1: Graphical User Inteface

The requirement 3.14 is vital for the participant, since it must not be difficult nor a hassle to operate the Bluetooth / mobile unit. It also incorporates in the section 2.2 where it is mentioned that a participant needs to see the lifestyle like a new habit. Any problems may work discouraging, especially if the technology seemingly is against the participant.

Requirement ID:	$NFR-2$
Requirement Name:	Webservice
Priority:	Low
Goal:	Making it easy for the Participant to use the service
Testing:	
Description:	The webservice is needed for the Participant to create missions and view completed missions.

Table 3.15: Requirement NFR-2: Webservice

A webservice that can create a track record of missions that has been completed and, also allows the participant to create new missions.

Requirement ID:	$NFR-3$
Requirement Name:	Motivate Participant
Priority:	Low
Goal:	Encourage the Partipant to continue physical activities
Testing:	
Description:	Encourage physical activity at the Participant, based on his terms.

Table 3.16: Requirement NFR-3: Motivate Participant

One way of motivating the participant, is to make sure the system works with the participant. Instead of making the pariticpant frustrated, the system should relay messages that can be understood, especially during a mission.

Chapter 4

Technology

In this chapter, the technology used in the thesis are being looked at. Also technology that can be a substitute or an alternative are also looked into. Argos with its standard components is listed first, then alternative hardware platforms for the beacon, and then the possible software platform for the mobile unit. The chapter will conclude with a discussion in regards of platform choices.

4.1 Argos

Argos is an middleware kernel with added functionality. Argos is used as an platform to develop a custom made service oriented end user applications, where some of the basic functionality already is provided by Argos and Argos system services. Services such as database persistence, web application, service distribution are basic system services in Argos.

4.1.1 Core Components

Argos comes originally with standard components, also customized components can be added by the service provider. As shown in figure 4.1, Argos consist of serveral components, the next few section will describe the components necessary for the custom component.

4.1.1.1 Derby

Derby[10][1] is an embedded relational database management system entirely written in Java. Derby is a powerfull and compact tool to support Java ap-

[1]More known as Apache Derby[11]

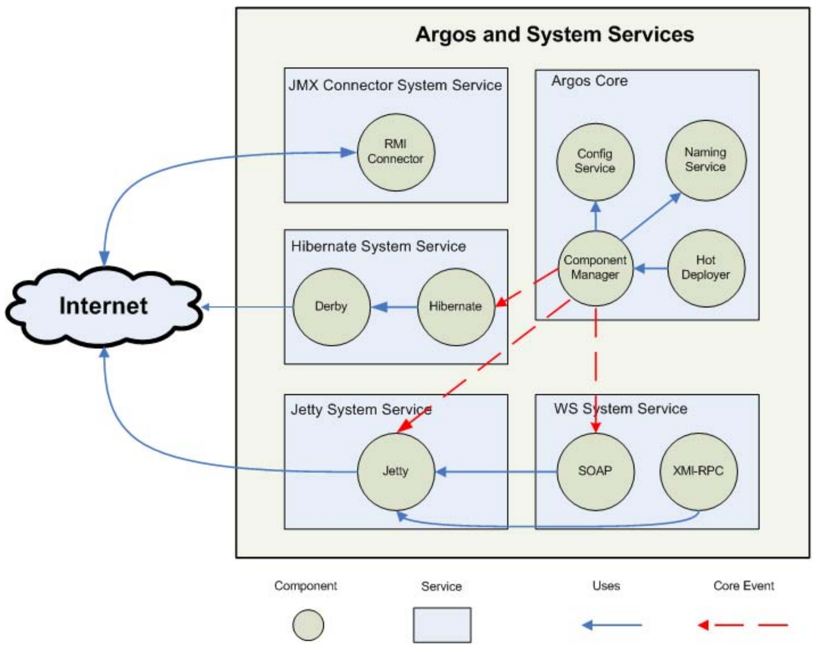


Figure 4.1: Overview of Argos[6]

plications with persistency. Derby has also a small footprint.

4.1.1.2 Jetty

Jetty[4] is a webserver implemented in java. Jetty can be used for a multitude of services such as:

- webserver for static and dynamic content
- embedded component within a java application
- traditional stand alone web server for static and dynamic content

Jetty is easy to program due to a minimal amount of lines to embed it to an instance. Jetty is configured by using an Extended Metalanguage(XML) file or Application Progamming Interface(API). When a component is added to Argos, the XML file that comes with the component, tells how the component is structured, dependencies and how it is engaged.

The figure 4.2 shows what the component contains and what dependecies it has. The component can viewed through an interface called Argus (see 4.2.

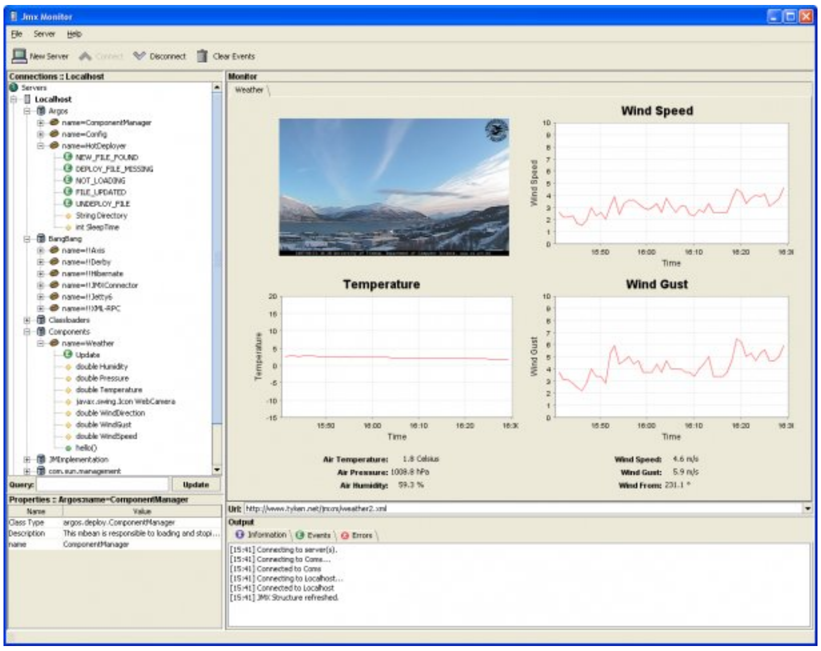
```
<?xml version = "1.0" encoding="UTF-8" standalone = "yes"?>

<service name="Encourager" version="1.0">
    <depend on="!!Hibernate"/>
<deploy>
<component name="DBComponent" class="encourager.db.DBComponent"/>
</deploy>
</service>
```

Figure 4.2: XML file describing the custom component

4.1.1.3 JMX

Java Management Extensions(JMX)[12] is used as an interface for the middleware platform Argos. JMX contributes with monitoring and management in Argos, which in turn gives Argos remote and local accessibilty to any component or resource connected with Argos.



```
Servers
  Localhost
    Argos
      name=ComponentManager
      name=Config
      name=HotDeployer
        NEW_FILE_FOUND
        DEPLOY_FILE_MISSING
        NOT_LOADING
        FILE_UPDATED
        UNDEPLOY_FILE
        String Directory
        int SleepTime
    BangBang
      name=!!Axis
      name=!!Derby
      name=!!Hibernate
      name=!!JMXConnector
      name=!!Jetty6
      name=!!XML-RPC
    Classloaders
    Components
      name=Weather
        Update
        double Humidity
        double Pressure
        double Temperature
        javax.swing.Icon WebCamera
        double WindDirection
        double WindGust
        double WindSpeed
        hello()
    JMImplementation
    com.sun.management
```

Properties :: Argos:name=ComponentManager

Name	Value
Class Type	argos.deploy.ComponentManager
Description	This mbean is responsible to loading and stopi...
name	ComponentManager

```
Air Temperature:  1.8 Celsius        Wind Speed:  4.6 m/s
Air Pressure:  1008.8 hPa            Wind Gust:   5.9 m/s
Air Humidity:  59.3 %                Wind From:   231.1 °
```

Url: http://www.tyken.net/jmxs/weather2.xml

```
[15:41] Connecting to server(s).
[15:41] Connecting to Coms....
[15:41] Connected to Coms
[15:41] Connecting to Localhost...
[15:41] Connected to Localhost
[15:41] JMX Structure refreshed.
```

Figure 4.3: Argus, the JMX browser

A graphical JMX interface can allow an admin or user to view the components running in Argos. There is a customized JMX graphical interface called Argus (see section 4.3) JMX browser[2] meant for monitoring the components in Argos.

[2]http://sourceforge.net/projects/argusjmx/

4.1.1.4 Hibernate

Hibernate[3] is an object-relational mapping library for Java. Hibernate makes the database seem like it contains java objects instead of queries, which makes the development for object-oriented developers[5] easier.

4.2 Argus

Argus is an GUI where it allows the developer to perform operations on components available through the JMX. The components contains methods, where type values can be sent to the method. This allows the developer to test wheter a methods in the component works as intended. However, Argus requires the type values to be object types, such as a Integer instead of a int, otherwise Argus will be refuse to use the selected method.

4.3 Bluetooth

This section will look at some of the physical layers and protocols of Bluetooth, but will focus on the software abit more.

Bluetooth[4] is a wireless network standard for a multitude of devices. Bluetooth is used to connect and exchange data between devices, using short-range radio frequency. Bluetooth acts as an interface to exchange data between different types of devices, for example PC and cellphone. The range for the bluetooth transmitter depends on the type, the transmitter ranges from 1, 10 to 100 meter, based on the power class. Bluetooth is continuously monitored and updated by the Bluetooth Special Interest Group (SIG)[5].

[3]http://hibernate.org
[4]http://en.wikipedia.org/wiki/Bluetooth
[5]http://bluetooth.com

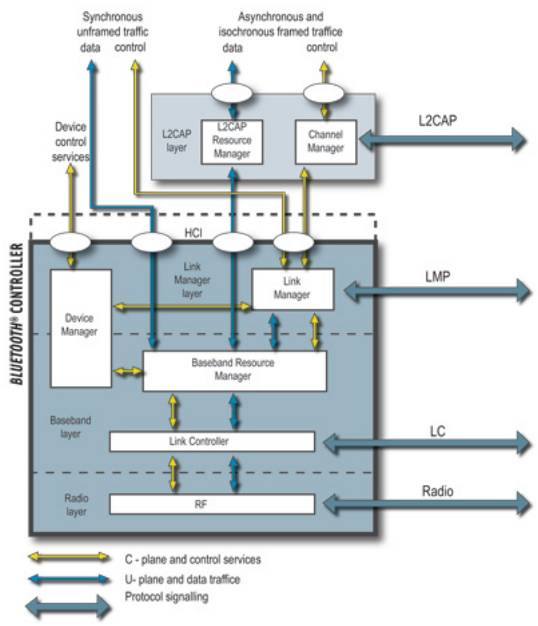


Figure 4.4: Controller of a Bluetooth Device

4.3.1 Layers and Protocols

The figure 4.4 shows the architecture of the Bluetooth core. The Bluetooth Core which covers the four lowest layers as well as the protocols defined by the Bluetooth specifications. There is a service layer protocol referred to as Service Discovery Protocol(SDP), SDP is used to identify the services other Bluetooth devices closeby provides.

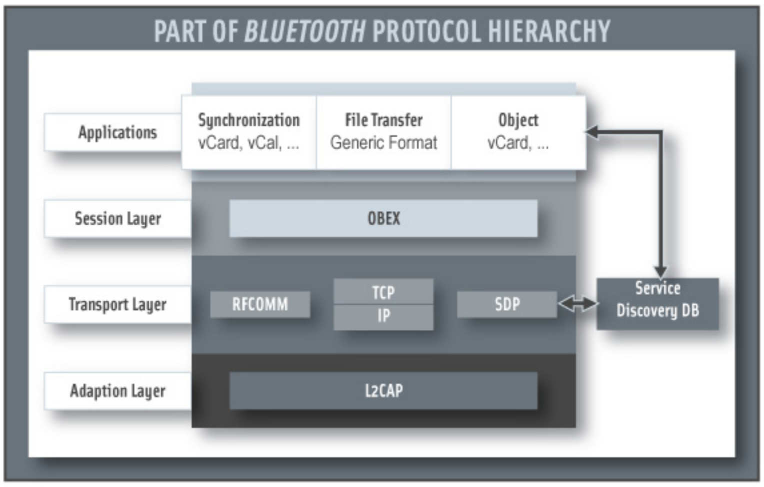


Figure 4.5: The Bluetooth protocol hierarchy

There are other protocols and layers which supports data transfer, such as the

General Object Exchange Profile(OBEX) protocol, Radio Frequency Communication (RFCOMM) and Logical Link Control and Adaptation Protocol(L2CAP)layer. The figure 4.5 shows that RFCOMM, OBEX and L2CAP are on three seperate layers.

L2CAP support Bluetooth with higher level protocol multiplexing, packet segmentation and reassembly. L2CAP makes it possible for higher level protocols to transmit and receive data.

RFCOMM is used to emulate the serial cable line setting and provides serial data transfer. In a simple configuration RFCOMM provides support for a Bluetooth link to another unit. RFCOMM supports the OBEX protocol.

OBEX is used to transfer data from one unit to another, the data can contain anything from images to pure text. In order to transfer data from one unit to another, the units has to be paired. Pairing means the units recognizes eac other as safe, by confirming a common passkey which both units shares. The passkey is a four digit number.

The specifications for L2CAP, RFCOMM and OBEX can be found at the Bluetooth SIG website[6].

4.3.2 Languages with Bluetooth support

The Bluetooth API is available at the Bluetooth SIG website. The problem is though, that the technical API for Bluetooth is not easy to comprehend for developers. To make it easier for developers, there are language specific APIs for the Bluetooth hardware. Most programming languages today contains support for Bluetooth. Bluetooth programming is essentially socket programming, where the programmer in most cases operates with protocols based on the L2CAP layer.

A good example is the tutorial by Albert Huang[7], where he describes the first steps for the languages C[7] and Python[8]. The programs developed on C or Python are meant for units that contains support for the languages, in most cases the units are computers.

For development on mobile units such as a smartphone[9]. JME (see section 4.5) and .Net (see section 4.6 can get support for development on the Bluetooth part of the smartphone.

[6]http://bluetooth.com/Bluetooth/Technology/Works/

[7]C can use the Bluez stack, which can be downloaded from http://www.bluez.org/

[8]Python can use the LightBlue stack, downloadable from http://lightblue.sourceforge.net/

[9]A smartphone offers capability beyond a normal mobile phone, where the functionality can compared to a computer (http://en.wikipedia.org/wiki/Smartphone)

4.4 RFID

Radio-frequency identification (RFID)[10] can be reckoned as an alternative to Bluetooth. RFID is mobile, wireless and does not take much space.

There are two types of RFID tags;

- passive
- active
- semi-passive

The passive RFID tag, has no powersupply either internal or external. A passive RFID tags way of getting enough power to transmit a response, is by inducing an electrical current in the antenna from an incoming radio frequency signal. That current alone, lets the Complementary metal-oxide-semiconductor (CMOS)[11] get enough power to send out an response. Depending on the radio frequency and antenna design, the reading range of an passive tag can go from a few centimeters to a couple of meters. The biggest advantage with a passive RFID, is its capability to induce power on its own, which makes it self reliant.

The active RFID tag on the other hand, has a powersupply. The power is necessary for the embedded services aswell as broadcasting a signal to the reader. The active tag is more reliable than the passive, due to the ability to initate a session with a reader. The active tag is not as challenged when it comes to signal strength, the passive tag will most certainly struggle with terrain obstacles due to the low signal strength. Also due to the stronger signal strength, the active tag can have a signal ranging for hundreds of meters instead of just a few. Active tags can have a battery with a life cycle up to 10 years depending on the service.

The semi-passive RFID tag is similar to the active tag, since it has its own power supply. However, the semi-passive tag does not broadcast a signal, it typically reflects a signal back like a passive tag. The passive state makes the semi-passive tag able to do other jobs while there are no readers in its vicinity. The semi-passive tag can monitor or log activities from other devices connected to it, then give the reader the data on request.

An semi-passive RFID tag can be ideal for this project, since it does not require a strict battery maintnance. The semi-passive has a good signal strength, a low signal might involve alot of tries before a connection occurs. The RFID tag does not need to broadcast endlessly or at all, the power surplus can be used to monitor devices that has connected with it.

[10]http://en.wikipedia.org/wiki/RFID
[11]http://en.wikipedia.org/wiki/CMOS

4.5 JME

Java Platform Micro Edition[12] (JME)[8] is developed by Sun Microsystems. JME provides with an flexible platform for mobile devices. The core of JME is based on the Java platform where JME deals with the constraints of developing on small devices (such as mobile phones).

JME is available on pretty much every phone sold on the market, JME is essentially the most common thing on a phone. If the phone has a game installed, it is most likely based on JME. JME is the platform that is most used among the producers of mobile phones[13] today, but the compatibility of the software for each type of phone varies.

In order to program Bluetooth on the JME platform, a framwork has to be available. Currently the Marge[14] is a recommended framework for Bluetooth development on smartphones. Marge is based on the JSR 82 API[15] specifications for Java.

4.6 Microsoft .Net Compact Framwork

Microsofts .Net Compact Framework (CF) shares the same philosophy as Suns JME, where it is meant to ease the development of software for mobile devices. CF do however emphasize on smartphones[16]. CF requires .Net available on the unit, phones such as the HTC phone from TyTN is based on .Net.

CF development can be done in C# or C++, but C# is preferable due to its easy memory management. The development .Net occurs directly through Visual Studio 2005, the CF grants the developer a graphical user interface(GUI) for the interface part of the development. From there the developer can customize each GUI control, making it easy to have the overhead view of each component created.

Much like JME requires a framework for development on Bluetooth, CF needs it as well. The framework from 32feet[17] gives CF the capability to develop Bluetooth programms on smartphones.

[12]JME has recently replaced J2ME

[13]Especially the major phone companies such as Nokia, SonyEriccson, Samsung and Motorola, have JME installed on their phones by default.

[14]https://marge.dev.java.net/

[15]http://www.jcp.org/en/jsr/detail?id=82

[16]A smartphone can be viewed as a mini PDA with the same capability, but is not as large as a PDA

[17]http://32feet.net/forums/p/396/1436.aspx

4.7 Discussion

Argos is a required part of the system and cannot be avoided. Argos is a personal middleware container, there are other similar systems available such as Jboss[18]. Argos is developed for systems like this thesis presents, since the system is small and contains just the core of services are required. The main difference between Argos and for instance Jboss, is the scalability. Argos is meant as a personal application server and therefore it does not meet the requirements for a corporation. The reason for the lack of scalability is to make it easier to develop components. Creating components for a scalable service takes vastly more time than a simple service, due to more scalability checks and testing.

4.7.1 RFID versus Bluetooth

Although on paper the RFID seems more ideal than the Bluetooth for this project, RFID is alot more restricted then Bluetooth. Very few, if any, commercial mobile devices has an RFID reader embedded. Bluetooth on the other hand, is embedded with virtually every mobile device produced today, ranging from controller units to cellphones and even in car radios. The RFID tags however, requires special equipment in order to work as intended.

The similarity between RFID and Bluetooth ends, when the topic multipurpose pops up. RFID is mainly used to tag or to transmit an identification. RFID can be seen used as a means to identify merchandise, cars passing a toll boot and so on. RFID can be embedded with other devices to create an information service, but in order to retrive the service, a customized device with RFID has to be used. In comparison, a bluetooth device can be of any size or type and just require software to interface with another bluetooth device. Normally the software comes follows the device unless the service is custom made. Even if the software is not available on the device originally, the software can be installed by transfering it through the Bluetooth interface. Since Bluetooth devices can adapt to any situation more easily than RFIDs, Bluetooth is a more valid choice as an client interface.

4.7.2 JME versus .Net Compact Frameworks

Though JME is a more common platform for mobile units, it does not mean it is easy to develop on. .Net emphasizes on easy implementation from scratch, where .Net allows the developer to just focus on the functionality, the underlying structure can be ignored.

[18]http://labs.jboss.com/

Both JME and .Net requires a third party framework in order to program on the Bluetooth. The Marge framework for JME, and 32Feet framework for .Net are easy to download.

Creating a sensible GUI is easy on both platforms, since the developer can see the GUI as he "draws" it. One major issue with both platforms, is the developement of the GUI, the screen varies from phone to phone. The thesis will only emphasize on the HTC Tytn phone with incorporated .Net, to limit the development time on the GUI.

Chapter 5

Design and Implementation

In this chapter both the design and implementation of the system will be displayed. Since the design has a very low priorty in the agile method Scrum, the design will not really be that descriptive. The design gives an overhead view of the system, sort of like a pointer how the system should work. The reason for Scrum to minimize time spent on design, is to limit the time spent on every working detail in the system, so the time can be used to handle implemenation and implementation issues. Therefore the implementation will describe the system in terms of details.

5.1 Platform choices

The server service is developed for Argos, and therefore the language platform is locked to XML and Java. The components are developed in Eclipse, where Argos is directly imported from the local repository [1] into the Eclipse environment.

The mobile unit (cell phone) platform however can be either on the well known JME platform or in .Net. However, since .Net is regarded as an easier platform to develop mobile software on, .Net Mobile Development / Compact Framework is used in the thesis. The Bluetooth part of the mobile unit requires additional library which is maintained by 32Feet. The library from 32Feet makes it possible to do the required Bluetooth operation, in this case the SDP query.

[1]Available from the ArticBeans Lab at the University of Tromsø and Sourceforge.

5.2 System Overview

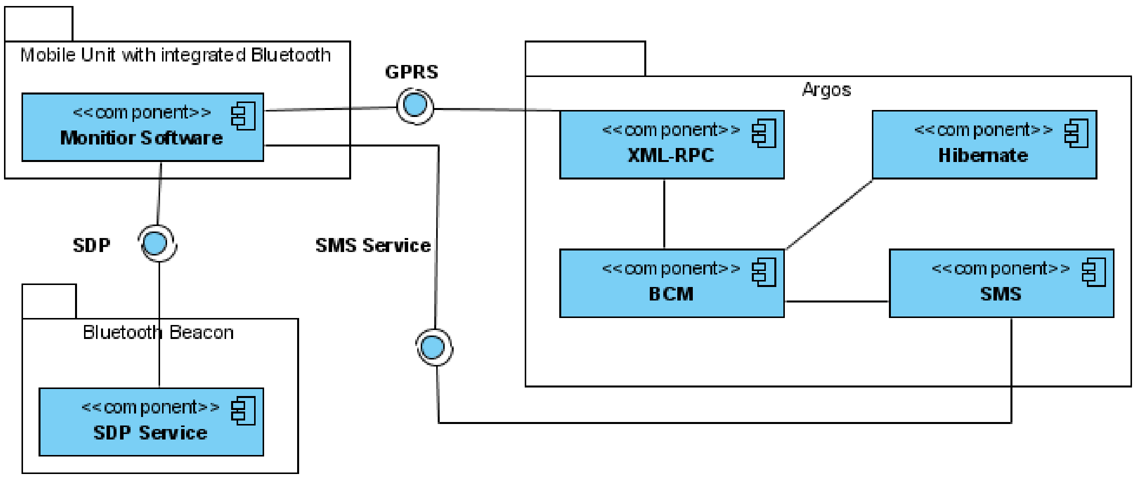


Figure 5.1: System Overview

The figure 5.1 is based on the figure 4.1. The very basic part of the system is described in figure 5.1. Argos uses some core components where the custom component (BCM) is included. The Hibernate component gives database support, the SMS enables Argos able to send setup messages to a Blueooth Device (mobile phone), and the XML-RPC component makes the mobile unit able to access Argos to download and upload missions.

5.3 Argos

Argos maintains an environment for the Participant where he can create mission specified by himself. The missions created are stored in the Argos database, each mission is sent to the Participant specified by a timer. The Participant himself decides how many missions he will complete each week.

The mission contains the details of the location of a device, a timer for when the mission is sent aswell as which device it is looking for.

5.3.1 The Argos component

The classes the Argos component is composed of are put into one package:

- encourager.db

Inside the package there are three classes;

- Participant
- Mission
- DBComponent

5.3.1.1 Overview of the Participant class

Contains the details of the participant. Keeps track of the current active mission as well as storing the missions that are completed. The main difference between a current and completed mission appears in the variable completedTime. The variable completedTime is null in a current mission. Once a mission is completed, the mission is moved from the current to the completedlist, and the completedTime variable is updated to time the mission was completed. However, just one mission is stored in the current mission list, this is due to the way hibernate stores objects. Hibernate cannot store a object within a object, unless it is through a hashset, it was found at the time to be the simplest solution to the problem.

5.3.1.2 Overview of the Mission class

The Mission class contains the details of both the current and completed missions, however the current mission and completed missions are stored seperatly.

The beaconid or hardware address is put in this class as a variable, this is to simplify the implementation regarding creating a mission. Since hibernate does not allow objects to be stored within a object.

5.3.1.3 Overview of the DBComponent class

Stores and retrives objects from the hibernate component in Argos. The operations available in hibernate are defined here. Operations like storing and updating a participant can be easily manipulated if needed.

5.4 The Mobile Unit

The mobile unit[2], has a specificly made software for detecting beacons specified in the missions received from Argos. The software is referred as "Monitor Software". The software on the device communicates through SDP, GPRS/3G/Edge and SMS.

[2]Can be a PDA, mobile phone or smartphone, it is not correct to referr the mobile unit as just a cellphone, due to the requirements.

When Argos got a mission ready for a specific device, Argos will send an SMS to that device. When a mission is ready, depends on the time setting of the mission. Whenever a mission related sms is received, the software on the device will intercept the SMS message and initiate the interface which accesses Argos, in this case it is GPRS.

Once the GPRS connection is established, the bluetooth device uses XML-RPC to download the mission specified in the SMS received from Argos. As soon as the mission has been completely downloaded, the mission can be started by the user.

The SDP service on the device, tries to locate the beacon specified in the mission, by looking at the returned hardware address from the beacon at the current location.

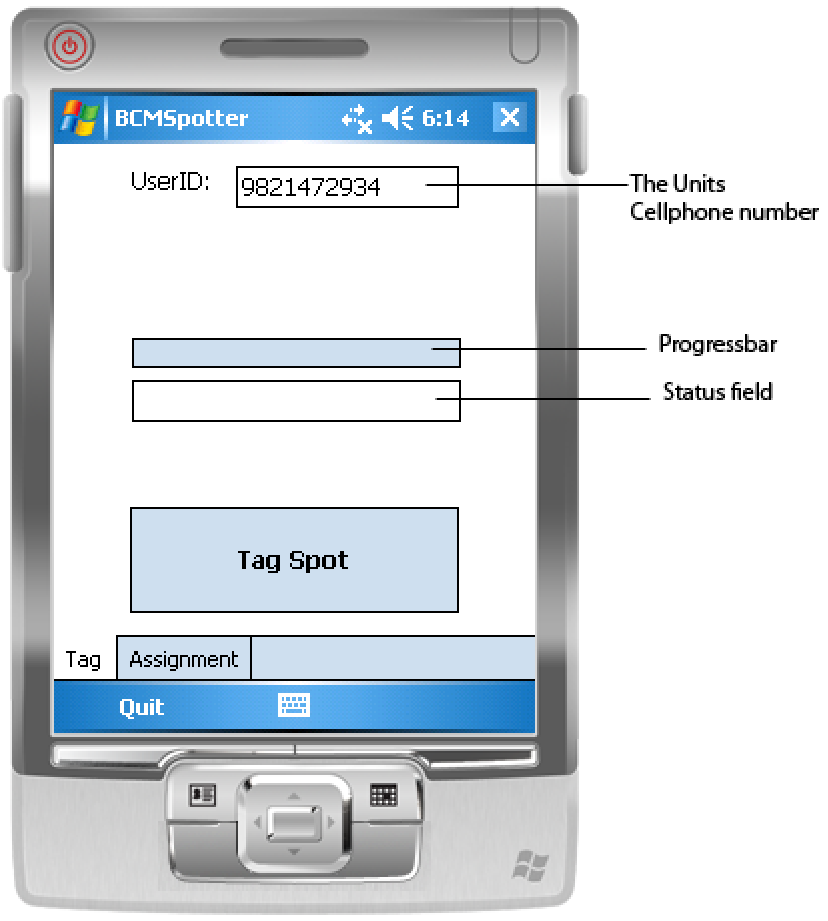


Figure 5.2: The Tag menu for the mobile client

Once the correct beacon is located, the device connects to Argos to upload data telling that a mission was completed. The uploaded data will also be stored in a record showing completed missions.

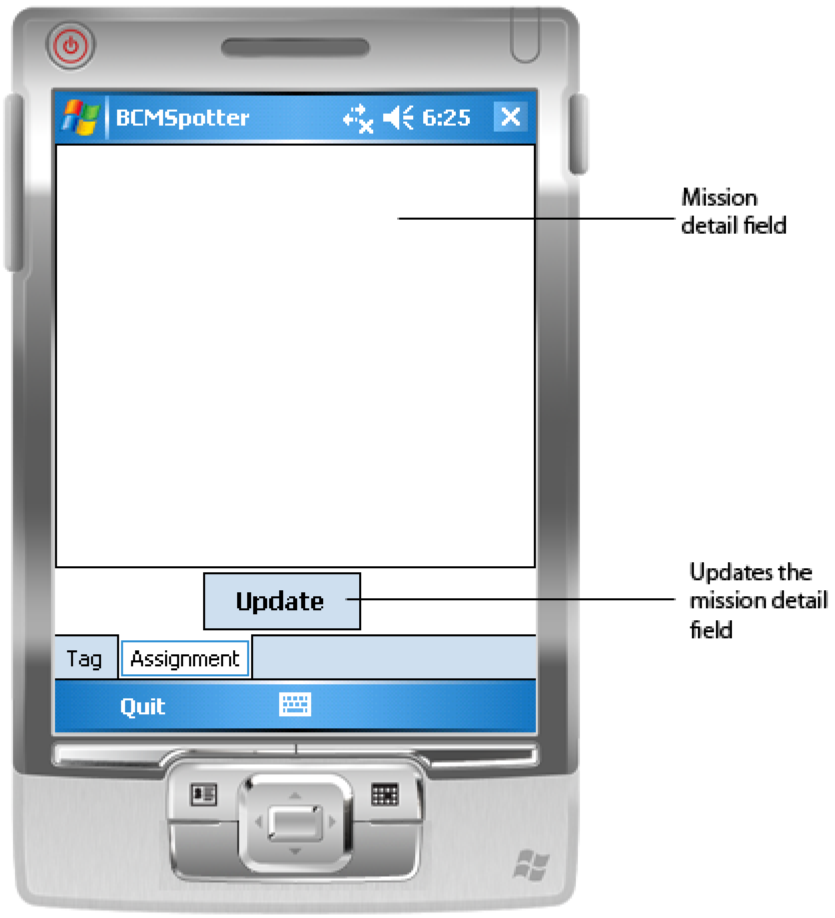


Figure 5.3: The current mission details downloaded from Argos

The only thing the participant shall do with his client, is to simply find the beacon, and the client will take of the rest.

5.5 Bluetooth Beacon

The Bluetooth beacon is "accessed" through a SDP query done by the mobile unit, the mobile unit checks the wheter the hardware address on the beacon is the same as the one mentioned in the mission details. The beacon itself does not require any software running it, it just needs to have a preset name such as "BCMSpot" in order to be detected as a beacon. By standard the Bluetoooth units have a name which referrs to the model, mobile phones especially have by default their producer name and model number set[3]. The software on the mobile unit looking for the beacon, does not search for any kind of services on

[3]Such as Nokia 5500, Thinkpad, Holux240Slim and so on.

the beacon. Therefore the beacon can be any sort of device, as long as it is possible to detect it with another Bluetooth Device, with a preset name.

Chapter 6

Evaluation

In this chapter the methodology used is evaluated, the sytem created is also evaluated against the requirements described in chapter 3. Both the functional and non-functional requirements are evaluted. The tables which contains the evaluation, evaluates wheter a requirement has been successfull, failed or been partially sucessfull.

In the section for functional requirements, the requirements for Argos and client are evaluated in the tables 6.1 and 6.2. The non-functional requirements are evaluated in the Non-functional section in the table 6.3.

6.1 Method

As described in section 1.5 the methodology used in this thesis were Scrum. The details regarding Scrum and how it works, were found in the book called "Agile Software Development with Scrum". The details from the book were applied when using the software from Scrumworks. Scrumworks lets the user setup his scrum project and backlogs for easier management. Normally Scrum is used by a team of developers, in this case it was set to one person. This means the methodology had to be modificated in order to fit with what resources that were available. A scrum team normally has a daily scrum, however not alot of work can be produced by just one person during one day to justify a daily scrum. A daily scrum is used to evaluate what has been done during one day, as well as coordinate work between the team members. Since there is just one person available to do all the work, the amount of daily scrums has to be reduced. Therefore the daily scrum was set to each Friday, however the scrum master was normally available any day, in case problems came up.

A scrum project starts out with identifiying all the important tasks, or at least as many as the user can initially detect so it possible to setup a sprint. In the beginning, although it is not allowed according to the Scrum methodology, backlog items could be inserted into the sprint because it was a necessary requirement and was not forseen when the sprint was setup. As the experience with Scrum grew, it became somewhat easier to setup a sprint. The biggest issue still remained; setting up a sprint by evaluating how much it is possible to do within two weeks.

In the beginning, a sprint filled with tasks that were deemed possible to achieve, quickly turned out to be more of a struggle than anticipated. Therefore some moderation had to be applied for each sprint. Setting up a sprint can be annoying, in the sense of evaluating how much work can be done within the time a sprint lasts. If the estimated work is too poorly set, the result will most likely be stress and perhaps self doubt wheter the sprint really can be completed. All tasks that were not complete during one sprint, can be moved on to the next. However, it certainly is not motivating to pretty much feel stuck or not being able to achieve the goal of the sprint.

Therefore its important to remember during a sprint, any obstacles needs to dealt with quickly. If something does not work, find a easier way out to do achieve the same. The main obstacles while working on this thesis, were related to learning technology fast enough and applying it.

Though Scrum is really a neat tool in order to have controll over the development, it is certainly not an easy habit to get into, especially when working alone. The full benefit of Scrum appears when working in a team, where everyone has to coordinate their work with others.

6.2 Functional requirements

Requirement-ID	Status	Evaluation
AP-1	Success	It is possible to add a participant to the hibernate table
AP-2	Fail	Only partially implemented. The function lacks a reference in the database component in Hibernate.
AM-1	Success	It is possible to add a mission to the hibernate database
AM-2	Fail	Not implemented
AM-3	Fail	Not implemented
AB-1	Partial	The beacon was embedded in the mission details. The reason for the solution, was to avoid complexity in the hibernate tables.
AB-2	Partial	The beacon was embedded in the mission details. So if a mission is removed, the beacon is removed with it.

Table 6.1: Evaluation of the Argos component

Requirement-ID	Status	Evaluation
CA-1	Fail	Not implemented
CA-2	Fail	Not implemented
CM-1	Fail	Not implemented
CM-2	Fail	Not implemented
CM-3	Fail	Not implemented
CB-1	Success	The software on the mobile unit can detect and determine wheter a beacon is present. Also it can determine it fits with the mission details.

Table 6.2: Evaluation of the client device

6.3 Non-functional requirements

The non-funtional requirements of the system, are evaluted in the table 6.3.

Requirement-ID	Status	Evaluation
NFR-1	Success	The GUI on the mobile unit is easy to use, and offers enough options for the Participant to complete his mission. The low amount of options makes it possible for the Participant to easy understand how to operate the mobile unit.
NFR-2	Fail	Since the database is not ready, nor the services around it, the webinterface could not be implemented.
NFR-3	Unknown	It is individiual what motivates a participant to continue his physical activity. However, a system that a participant actively uses, can possibly keep his interest over time.

Table 6.3: Non-functional requirements

Chapter 7

Conclusion

This chaper summarises this theses by first concluding wheter the problem definition has been successfully achieved. Then future work will be looked at.

7.1 Achievements

The problem definition from section 1.2 states:

> *The goal of this thesis is to create a software system that will promote physical activity to its users. The software purpose is both to get the users in activty as well as monitoring their progress. The user is from this point referred as an participant.*
>
> *The software system will give the participant missions, which can be given daily, weekly or at some other rate. The missions are created by the participant himself. The missions must be completed within a certain amount of time. A mission consists of a waypoint the participant has to go to and be registered at. The mission the waypoint represents, is completed when the participant has registered it.*

During this thesis a prototype according to the description of the system, has not been successfully developed. The system lacks features in both the Argos and client side in order to function as a whole. Only small parts of the system really works, such as applying a participant object into the hibernate database, and a mobile unit software which can identify Bluetooth beacons.

The struggle has been trying to create a service using Argos. Though Argos is developed to create personal applications very easy, there are alot to consider. Creating a simple structure for Argos and the client side has proven itself quite difficult. Often the structure had to be re evaluted due to unexcpected impendiments, which delayed the development. It

Eventhough the software is not fully functional, the platform and structure has been designed. It is just a matter of removing the impendiments that currently exsist in the software.

7.2 Future work

7.2.1 Fully functional hibernate service

The hibernate tables proper support for putting missions into the hashset in a participant object instance. Also Bluetooth Beacons should be added as a class, to make it easier for the participant to locate beacons which he is interested in. Having a list of beacons which can be access through the web interface, is by far easier than typing the hardware address of each device by hand.

7.2.2 Argos and the mobile unit

Need to setup SMS and GPRS/3G support for the Argos application and the mobile unit. In order to send and receive mission details between Argos and the mobile unit, both SMS and some sort of internet connection must be available.

7.2.3 Web interface

An easy to use webinterface which allows the participant to connect with the Argos application, must be integrated. Right now, the hibernate tables are accessed through Argus, which is not a satisfying situation.

7.3 Conclusion

Though the system in its current state is not usable, there are still useable parts in it. Creating a service which is composed of two unique platforms, proved itself harder than first imagined. Still, the design of the GUI on the mobile unit is still good and can be developed further. Also the Argos component can be expanded with more of its intended functionality.

After working with Argos, the understanding of how Argos works and how it is to implement a custom application for Argos has increased. Though Argos has been a struggle to work with, the benefit is visible. It is just a matter of learning how to apply more complex services to the Argos platform.

Bibliography

[1] Mariva H. Aviram. Self help industry. 2001. URL: http://www.mariva.com/essays/self-help.html. 11

[2] Tom Baranowski, Karen W. Cullen, Theresa Nicklas, Deborah Thompson, , and Janice Baranowski. Are current health behavioral change models helpful in guiding prevention of weight gain efforts? October 2003. URL: http://www.obesityresearch.org/cgi/content/abstract/11/suppl_1/23S. 12

[3] Shelle Rose Charvet. Real behavior change: What does it take to break a habit? 2007. URL: http:www.selfgrowth.com. 11

[4] Mortbay Consulting. Jetty. 2007. URL: www.mortbay.org. 28

[5] James Elliot. What is hibernate. September 2005. URL: http://www.onjava.com/pub/a/onjava/2005/09/21/what-is-hibernate.html. 30

[6] Dan Peder Eriksen. Argos container, core and extension framework. June 2007. URL: http://henry.ub.uit.no/munin/handle/10037/1138?language=no. 28

[7] Albert Huang. An introduction to bluetooth programming. URL: http://people.csail.mit.edu/albert/bluez-intro/. 32

[8] Sun Microsystems. Java me technology. 2008. URL: http://java.sun.com/javame/index.jsp. 34

[9] George Orwell. *1984*. Secker and Warburg, June 1949. ISBN Paperback: 0-452-28423-6. 13

[10] Pan Pantziarka. Java database with derby. November 2006. URL: http://www.regdeveloper.co.uk/2006/11/08/java_databaseadfas_derby/. 27

[11] The Apache DB project. Apache derby. December 2007. http://db.apache.org/derby/. 27

[12] Sun. Enterprise java technologies: Jmx. February 2005. http://java.sun.com/developer/EJTechTips/2005/tt0222.html. 29

[13] Wikipedia. Self-help. URL: http://en.wikipedia.org/wiki/Self-help. 11