INF-3981

MASTER'S THESIS IN COMPUTER SCIENCE

Design and Implementation of a Prototype For Capturing Immediate User Experiences

Espen Ekvang

June, 2007

FACULTY OF SCIENCE
Department of Computer Science
University of Tromsø

Abstract

Everywhere we go we are overwhelmed with impressions and experiences making our day. Research has proven these to change over time, thus psychologists and experimenters in surveys are interested in what a person actually experienced at a particular point in time, referred to as *immediate experience*. Whether it was as a reaction to some event or just at specific times during the day, is determined by the purpose of the project. By allowing an experimenter to ask questions to a set of participants at predefined times and dates, the approach called Experience Sampling Method (ESM) aims at capturing user experiences. To be able to ask such questions participants are required to carry some kind of device running the actual survey. Since most people possesses a mobile phone, this can serve as an excellent host for ESM surveys. In this thesis we have studied the ESM approach, and the design and implementation of a prototype framework for creating, running and collecting results from surveys based on ESM will be presented. Several mobile operating systems and devices exist, hence an important concern in our work has been interoperability between different devices. To achieve this goal we have emphasised on expressing all information exchanged between a computer and a mobile device in eXtensible Markup Language (XML).

From our work we have seen that the flexibility of the prototype makes the system not limited to psychological surveys only, but the approach can be used as a very powerful tool for example in market analysis.

Acknowledgements

First, I would like to thank my primary supervisor, Anders Andersen, for providing feedback, ideas and time to proofread my work. Also thanks to co. supervisor Arne Munch-Ellingsen for his function as the Scrum Master, providing feedback and very useful discussions related to this thesis. Thanks to Gunnvald Bendix Svendsen (Telenor) and Joar Vittersø (Department of psychology, University of Tromsø) for their expert knowledge from the psychology domain. Finally, thanks to Telenor Research & Innovation for giving me the opportunity of presenting my work in both Tromsø and Oslo.

This thesis would not have been completed with them. Thank you.

Contents

List of Figures

List of Tables

Listings

List Of Acronyms

API Application Programmer Interface

CAB Cabinet

CBSD Component-Based Software Development

CPA Content Provider Access

CSS Cascading Style Sheet

CSV Comma-Separated Values

DLL Dynamic Link Library

DTD Document Type Definition

ER Entity Relationship

ESM Experience Sampling Method

FRD Functional Requirement esmDesk

FRM Functional Requirement esmMobile

GPS Global Positioning System

GPRS General Packet Radio Service

GSM Global System for Mobile communication

GUI Graphical User Interface

HTML HyperText Markup Language

HTTP Hyper Text Transfer Protocol

IDE Integrated Development Environment

IP Internet Protocol

J2ME Java 2 Micro Edition

JAXB Java Architecture for XML Binding

JMX Java Management Extensions

MBean Managed Bean

MCSF Mobile Client Software Factory

MMS Multimedia Messaging Service

NFR Non-Functional Requirement

NST Norwegian Centre for Telemedicine

OO Object Oriented

OS Operating System

PDA Personal Digital Assistant

PIM Personal Information Manager

SDK Software Development Kit

SIM Subscriber Identity Module

SOA Service Oriented Architecture

SOAP Simple Object Access Protocol

SMS Short Message Service

TCP Transmission Control Protocol

UMTS Universal Mobile Telecommunications System

UDDI Universal Description, Discovery and Integration

URL Uniform Resource Locator

WAP Wireless Application Protocol

W3C World Wide Web Consortium

WLAN Wireless Local Area Network

WSDL Web Service Definition Language

WWW World Wide Web

XAML eXtensible Application Markup Language

XML eXtensible Markup Language

XUL XML User Interface Language

Chapter 1

Introduction

1.1 Background

Collecting information on how people experience different situations, their impressions of new products and their responses to certain events are necessary to determine the next step in any evolving process. A widespread approach for collecting such information is the retrospective technique where a set of participants are given a number of questions to answer. The technique is retrospective because it asks people to look back and recall the characteristics of their experiences in a specific situation. Such an approach requires that each participant has the ability to reflect over a passed situation, and remember what they experienced in that situation. Research within this field has shown that there is a significant difference between *immediate* and *remembered* experiences [1]. The former of these two captures impressions a person has immediately after a situation has occurred, consequently these experiences have not been distorted over time.

The ability to remember is one of the most important qualities of a person. Without some kind of memory we would not know where to go, what to do or when to do it. When a person is asked a question related to the past, that person uses his or hers memory to recall what happened and what impressions he or she got from that event. In [2] a difference is made between experiential knowledge and memory, illustrated using a roller coaster example. Immediately after riding such an attraction, a number of feelings and impressions can be reported by the person who rode it. With time these impressions are weakened, and after a certain time the only experience a person has is that he or she remembers riding it. Another example includes a survey done between the election in 1972 and 1976 in the United States of America. Out of the people in 1976 that said they voted the same as in 1972, 91% voted differently. These two examples illustrate the importance of asking people questions immediately after an event has happened. The problem is not us remembering incorrectly, but how and what we remember. As time goes by, other events and experiences replace and distort older remembered ones.

The method called *Experience Sampling Method (ESM)* aims at capturing immediate experience from participants in a survey. Originally the method was created to capture details around how people spent their time in everyday life, and to get experiences *in situ*. Combining this method with today's mobile technology can make a very powerful tool for capturing immediate experiences. Most people carry a mobile phone or a Personal Digital Assistant (PDA) wherever they go, making it easy to ask them questions immediately after a certain time or at specific

intervals during the day.

The ESM approach opens for other possibilities than just asking participants questions related to how they feel under certain circumstances. Instead of asking questions at predefined time intervals, a mobile device can be equipped with sensors allowing it to collect context information surrounding the mobile user. Many new mobile devices have Global Positioning System (GPS) as a built-in feature, hence the device can prompt questions dependent on the geographical position of the device, or store information about where the user has been. It also opens for the possibility of presenting a participant with information when and where he or she might need it [3].

1.2 Problem Definition

The main focus in this thesis is to develop a software tool for simplifying the process of creating, running and collecting results from surveys based on the ESM approach. The development shall be done with help from the middleware platform known as Argos[1] to clarify how applications built on top of Argos can benefit from its features.

1.3 Interpretation

From the problem definition we have split the software development in two parts:

- Design and develop a desktop application for creating, distributing and collecting results from ESM-based surveys.
- Design and develop a generic ESM engine for running the actual surveys on PDA devices.

Due to the limited amount of time available in this thesis, the software tool will be developed as a prototype. When building a new software system it is very important to focus on the acquisition of the knowledge that is to complete the system, rather than actually building a fully functional system. In [4] this is illustrated through a very simple example. Starting out coding a new system a lot of functionality is added to it and during the development cycle various testing is performed to verify that the system behaves intentionally. During this process one can make a separation between *knowledge* and *non-knowledge*. The first is the one we want, while the second is errors or behaviour that we do not want the new system to employ. The non-knowledge is removed from the developed system, and at a certain state the system contains only the wanted knowledge. Even though the system does not contain the non-knowledge, there may still be leads to where in the code they actually occurred. Only the developer knows and understands what would have to be redesigned to remove it completely. Over time he or she might forget it or stop working for the company that made the system. The company will have a working system, but no one that knows why certain parts of the code may lead to errors due to faults in an early phase of the development. The process one should follow is to rewrite the code once all the necessary knowledge is in place. The process of rewriting the code should be rather quick if the observations done in the first phase are documented and well understood. This approach is called prototyping and will provide us with knowledge about a future system.

[1] http://argos.cs.uit.no

With prototyping it is important to demonstrate a concept, this way understanding the problem at hand and get an overview of the possible ways to solve it.

Another important aspect is that in an ESM survey, a participant should not have to give away its mobile device in order to get it set up for a particular survey. The software tool must be built in such a way that a mobile device simply can register to be a participant of a survey, from that point the ESM platform should automatically install itself on the mobile device and receive the survey that the person wanted to participate in. Alternatively, there should be a technique for distributing the software out to a set of mobile devices.

1.4 Limitations

Because the software is going to be built as a prototype, considerably effort must be put into making both parts of the software as generic as possible. This way future work and extensions can be applied to the tool without having to modify the fundamentals on which it has been built. Running an actual survey with real participants is not possible within the time constraints of this project. There are two other parts of a fully functional software tool not included in the development of this thesis, they are analysis and the ability to instruct a mobile device to read data from sensors. Analysis is the process of presenting results from a survey using charts and tables.

Many challenges are introduced when developing mobile applications. These include [5] small screens, limited power supply and variable level of connectivity. At the time of writing a mobile device does not have a static IP address like a desktop computer can have. The only way to uniquely identify a mobile device is by its mobile phone number residing on the Subscriber Identity Module (SIM) card inserted into the device. When defining the communication paradigm for the software tool, this limitation must be taken into consideration.

1.5 Method And Approach

The software development in this project should follow the lightweight agile method called *Scrum* which is described in [6]. In 1993 Scrum was first applied in Easel Corporation by Jeff Sutherland, John Scumniotales and Jeff McKenna. The method is very different from other traditional software development processes such as Rational Unified Process (RUP)[2] and the Waterfall[3] process. These other methods are iterative or linear and follow a normal approach which includes these phases: analysis, design, implementation and finally testing. That said, the Waterfall process is more bound to the sequence of these phases than RUP. As most developers know, requirements do change over time and the Scrum approach allows them to. Scrum is a team based approach, where each team member works individually but towards the same goal. Each team has a leader called the Scrum Master being responsible for implementing and managing the Scrum process in the project.

Scrum defines a set of backlogs; *Product backlog*, *Release backlog* and *Sprint backlog*. The product backlog contains all the high level requirements aimed at some release in the future,

[2] http://www-306.ibm.com/software/awdtools/rup/
[3] http://www.buzzle.com/editorials/3-13-2005-67039.asp

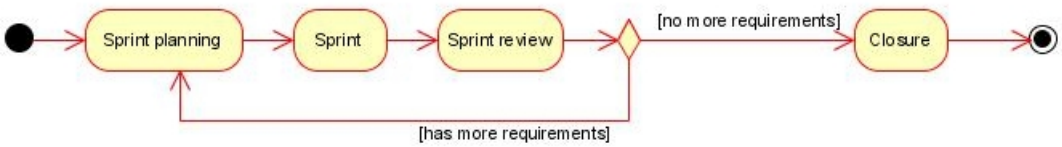


Figure 1.1: The activities of the Scrum software development process

and they typically come from the stakeholders of the new product. The release backlog contains requirements taken from the product backlog and identified to be a part of a certain release. Here the requirements usually carry more details than the one in the product backlog. The final backlog is the sprint backlog, and consists of requirements from the release backlog, these requirements are specified down to tasks taking no more than 8-12 hours. A *sprint* lasts up to four weeks and the team should design, develop, test and document software components based on the requirements specified in the sprint backlog. Before each sprint, the team meets with the Scrum Master and they have a sprint planning meeting where they decide what requirements to include in the upcoming sprint. Within a sprint there are *daily scrums*; a meeting of no more than 15 minutes where the Scrum Master meets the team members to discuss the progress of their work. For this meeting only three questions are allowed:

1. What have you done since the last daily scrum?
2. What has impeded your work?
3. What are you planning on doing for the next daily scrum?

Following a sprint is a *sprint review* where they meet again to compare the accomplished work with the backlog items, possibly resulting in adding more requirements to the product backlog. At the time when no more requirements are present in the product backlog, the development enters the last stage called the *closure* where the final debugging, marketing and promotion take place. Figure 1.1 illustrates the activities of the development process.

This particular project is an individual project, hence the team in the process consists of one person only. The role of the Scrum master is given to the supervisors of the project, but the author will also be able to edit the product backlog. Each sprint has been set to a period of approximately 2 weeks and daily scrums are to be held every Wednesday and Friday. Organisation of the different backlogs and sprint should be done using ScrumWorks[4], Unleashed Team Self-Organisation.

1.6 Relation to Telenor and NST

In today's society a very important factor is to increase efficiency in all of its areas. As a consequence of many working hours per day, people eat more quickly prepared and unhealthy

[4]http://danube.com/scrumworks

food, and an increasing problem is the weight gain among the world's population, especially the western part of the world. In addition to gaining weight, an unhealthy diet can cause what is characterised as lifetime diseases. Telenor, Norwegian Centre for Telemedicine (NST) and University of Tromsø are collaborating in a project called the *Lifestyle Tool*, which goal is to help people change their unhealthy lifestyle and maintain a lifestyle that can prevent them from being hit by lifetime diseases. Typical areas of interest in the project are eating habits, lack of exercise and substance abuse.

The preferred method for investigating and evaluating the *Lifestyle Tool* is ESM, hence the motivation behind this thesis comes from that project. That said, the development of the ESM prototype is done from a general perspective meaning that it should be able to employ it in other types of projects and surveys as well.

1.7 Major Results

Our study of the ESM approach has shown that it is very useful for capturing immediate user experiences. Through the development of the prototype framework for creating, running and collecting results from ESM surveys we saw that the approach is suitable also for other purposes than just regular psychological surveys, including market analysis. Employing the approach in early market analysis for a product could help a company determine what to focus on in developing new products.

After giving two presentations for Telenor Research & Innovation (R&I), they have decided to adopt the work done in this thesis in psychological test projects scheduled to take place in summer 2007.

1.8 Outline

The structure of this master's thesis is as follows:

- **Chapter 2: Related Work**
  This chapter gives an overview of work done in the area of ESM when it comes to software systems.
- **Chapter 3: Requirements**
  In this chapter we present the functional and non-functional requirements of the software developed in this thesis.
- **Chapter 4: Technology**
  Here we present different technologies relevant to the work done in this thesis.
- **Chapter 5: Design**
  Describes the design of the two software systems identified in section 1.3.
- **Chapter 6: Implementation**
  The implementation of the two parts of the software tool from section 1.3 is outlined here.
- **Chapter 7: Experiment**
  Presents the experiment done using the software designed and implemented in this thesis.

- **Chapter 8: Evaluation**
  An evaluation of the work done in this thesis with respect to the requirements listed in chapter 3.
- **Chapter 9: Conclusion**
  The conclusion of this thesis is drawn here.

- **Appendix A: Functional Requirements**
  A structured representation of the functional requirements for both of the systems.
- **Appendix B: esmDesk - Package Diagram**
  The package diagram for the desktop application esmDesk.
- **Appendix C: XML Schemas**
  XML Schemas for surveys and results.
- **Appendix D: esmService - Diagrams**
  Class and Entity-Relationship diagram for esmService.
- **Appendix E: esmMobile - ER Diagram**
  Entity-Relationship diagram for the database on the mobile device.
- **Appendix F: Experiment**
  The XML file representing the survey created during the experiment in this thesis.
- **Appendix G: User Guide**
  A user guide for esmDesk and esmMobile with screen dumps from both systems.
- **Appendix H: CD-ROM**
  A description of the contents on the CD-ROM attached to the thesis is outlined here.

Chapter 2

Related Work

In this chapter we will present a brief history of ESM, different protocols within it and existing systems available for use with an ESM study. The next part of this chapter will present some work done with respect to representing user interfaces in textural form such as XML. Finally, a set of communication schemes between a desktop computer and a mobile device will be presented.

2.1 ESM History

The concept of ESM is not new and the first ESM-like study goes all the way back to J.C. Flugel's study of mood in 1925 [7]. Even though this was the first occurrence of what can be characterised as ESM, the methodology has changed over the years. The approach done by Ed Diener and his colleges in the 1980s for measuring mood across situations [7] is close to the approach we want to achieve in our work. The problem with the early versions of ESM was that they were mainly observer reports, hence not a reflection of the feelings or experiences from participants. The reports described how a participant experienced a situation seen from an observer's perspective. With the introduction of self-reporting this changed. Self-reporting means that participants report their experiences by themselves and give them to the experimenter(s). The motivation for doing studies based on the ESM-approach can be shown through a simple illustration as given in figure 2.1. A closer inspection of this figure reveals that the person who has been tested, has its pain peak approximately half way through the test. At the end, the person reports a value to indicate the maximum pain he or she felt during the recorded period. The value is lower than previously registered peak level. In a situation where the level of the pain peak had not been registered, the observers would never know that the person felt so much pain.

2.2 ESM Protocols

In [8] a categorisation of ESM studies is presented. An ESM study can be bound to one of the following protocols, or it can combine two or more protocols to achieve a better result.

2.2.1 Interval-contingent

As the name of this protocol states, it will ask questions or require the participant to report at fixed times during the day (e.g. in the morning, afternoon or evening). Within a certain

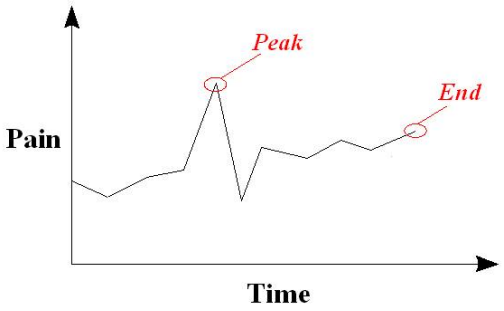


Figure 2.1: Registered pain for a person over time

interval a participant can either be asked to report something at a specific time (e.g. every hour) or the participant can be instructed to report if some situation has occurred within the current interval.

2.2.2 Signal-contingent

This approach is different from the previous one in that there is no pattern for when the participant is required to report or answer questions. The idea is that the participant should report as a response to a signal given at random times during the day.

2.2.3 Event-contingent

The last protocol requires the participant's attention immediately after or closely following a particular event of interest. As with the others, when the event has happened the participant might be required to answer questions or simply report its experience of the particular event.

2.3 ESM Approaches

History of ESM reveals that different methods are used within the field of experience-sampling studies. Deciding what approach to use is often determined by different factors such as project goal, budget and number of participants. It is important to choose an approach that is applicable to the projects profile.

2.3.1 Pager And Booklet

The first type of self-reporting tools included alarm watches and pagers worn by participants. An alarm clock gave random signals and the participant had to report their feelings and experiences by writing them on a piece of paper. For the pager an ESM researcher had to call it to notify the participant to make a note of its current experiences. Along with the pager the participant was equipped with a booklet [9] which contained a set of questionnaires. Each time the experimenter paged the participant, one of these questionnaires were required to be filled in. At the end of a survey period (usually 1-2 weeks) the booklet contained useful information about that persons life and experiences in certain situations. With these approaches it was required that the person taking part in the survey always carried around a pencil and a piece of paper/the booklet. This

sometimes became a burden, which resulted in wrong and distorted results, in some cases the person did not want to report his or her feelings at all. As a consequence of including all questionnaires in the booklet, some participants were tempted by the opportunity they had to fill in these questionnaires in advance. Hence, when they were paged they simply ignored the signal because the questions were already answered.

2.3.2 Hand-held device

Fortunately, with the emergence in mobile technology, running an ESM survey has become less cumbersome than before. Most people carry around their mobile devices anyway, meaning that no additional equipment is required for them to participate in a survey. That said, a mobile device is not as generic as a computer and in many situations software developed for one type of mobile devices may not work, or work incorrectly on a device from another vendor. This puts a lot of additional requirements on the software developers, thus the normal approach is for them to create ESM software targeting one particular type of devices (examples include PDAs and palmtop devices). Another important limitation of mobile devices is the small screens, which cause the people creating the questions to make them as short as possible to fit into the screen boundaries. Ideally a participant should be presented with both a question and an area to give his or her answer.

The idea with the hand-held device approach is to minimise the burden put on participants using the booklet and pager method. Questionnaires are preloaded into the device and the participant is not able to the view questions before he or she should. Advanced techniques in mobile computing [3] enable the approach to combine different ESM protocols in a more sophisticated way than the pager and booklet does. Allowing the mobile device to absorb context information is such a technique. The geographical position of a device is an example of context information, obtainable using GPS. The ESM study can then be configured to periodically check for particular context information or listen for events fired by the context provider of the mobile device. When a certain condition is met, the software can ask the participant questions. This eliminates the stage of the ESM process where the participant had to report when he or she was in a particular situation, consequently minimising the burden on the participant.

2.3.3 Web Use Experiences

Internet has become one of the most important information channels in today's modern society. One of the challenges that information providers face, is how information should be presented on the Internet. What is the best way to get the attention from the wanted audience and so on. A way to obtain user experiences is the traditional way of sending surveys out on randomly selected email accounts. Such emails are often characterised as spam mail, hence filtered out and in many situations never read by the receiver. In [9] a different approach is presented using the ESM technique. This software asks a person questions while he or she is browsing the web. A small application runs in the background of the web browser and prompts the user with questions based on the different sites and activities visited by the participant. Questions will not be asked in situations where the user is not browsing the web, for example reading a newspaper etc. This is because it should not interfere too much with the participants habits. The advantage of this approach is that the results from questionnaires are directly transferred

Figure 2.2: ESP - Experience Sampling Program

to the experimenter after a study has been finished. This minimises the physical contact needed between the participant and the experimenter of an ESM survey.

2.4 Existing ESM Systems

For a team that wants to launch an experience-sampling study a number of systems are available for purchase or download from the Internet. In this section we will present a selection of these existing systems with respect to the hand-held approach mentioned earlier. Their advantages, disadvantages and how to obtain them will be outlined. The software presented in this section can be separated into two main categories: *Free, open source software* and *proprietary software.*

2.4.1 Free, Open Source ESM software

This first type has the obvious advantage that it is free, hence it can simply be downloaded from the Internet and set up to meet an experimenter's requirements. The main disadvantage of this type of software is that it has no technical support at all, apart from user forums usually available on the site providing the software. In a situation where you have a problem, you most likely have to figure them out by yourself, or create a thread in one of the user forums to get in touch with people experiencing the same problem.

ESP - The Experience Sampling Program

ESP[1] is free and developed for running questionnaires, surveys or other experiments on Palm Pilot devices. The software is open source and has been created by Daniel J. Barrett and Lisa Feldman Barrett and the current version number is 4.0 (2005). Like most of the ESM software it contains two parts: one part to install on a desktop computer and a second part to install on the handheld device. Using the desktop ESP, the experimenter can define new experiments and view old ones. This application has been designed as a web-site and, according to its documentation, it runs best using Mozilla[2] web-browser. Because of its simplicity ESP has become very popular, and among its features are support for all the ESM protocols and the possibility of latency recording. That is the time taken from a question is presented to the user until the user actually replies with an answer.

After the survey has been created, it must be transferred to a set of Palm Pilots. When using

[1] http://www.experience-sampling.org
[2] http://www.mozilla.org

ESP it is required that the experimenter has all of the devices in its possession before the survey starts. The reason for this is that the survey must be transferred by wire from the desktop computer to the handheld device using some synchronisation program provided with the device. In situations where a large group of participants have been chosen, the time taken to install and set up each device can be quite extensive. Of course, this also means that when the survey is finished, the experimenter must collect all devices and bring them back to the desktop computer. Here each device must be connected to download the results given by each participant. According to the information site for ESP, a Palm Pilot or PDA running ESP must not be used for other purposes, hence a participant must carry a device around with the only purpose of participating in a survey. The main reason for this is that the Palm Operating System (OS) up to version six does not support multitasking. Figure 2.2 shows screenshot from the handheld device running ESP to the left and the ESP desktop application to the right.

Since ESP is open source, it can be modified to suit the needs of anyone. A team at Intel Corporation did this and introduced the iESP[3] in June 2003. The modifications done to the software include new features listed on their website, also they fixed several bugs in the previous version. The original ESP has an associated user group at Yahoo[4] which can be very useful for troubleshooting, and is fully active. The iESP is less frequently used, hence no user group has been created.

PMAT - Purdue Momentary Assessment Tool

PMAT has been designed and developed at Military Family Research Institute, Purdue University and the people behind it are Howard M. Weiss, Daniel J. Beal, Samantha L. Lucy and Shelly M. MacDermid [10]. The software is very similar to ESP, yet it has some other features. The desktop application is not a web site, but a stand alone application offering a quite good user interface (see figure 2.3). One of the improvements of this software in contrast with ESP is the ability to combine different ESM protocols when creating a survey. A simple example illustrates this feature. Say a survey combining the event and interval protocol has been issued. When a specific event occurs, the participant reports/registers this using the device. Then after a predefined interval, the device prompts a question to the user. This question follows the interval protocol because it shall be prompted at a specific time, but that time is relative to when a particular event has happened. Such a configuration is setup by the experimenter using the desktop application which is a part of PMAT.

Once the study has been created using the desktop software, it must be transferred onto the devices to be used in the survey. The first step in this process is to create a number of database files describing the survey. These database files are Palmtop specific and are called Palm DataBase files, hence the system assumes that whatever devices used in a survey must have set up and configured their Palm DataBases. From the documentation it seems that the generation of these database files is only possible ones, this assumption is based on the fact that they encourage the experimenter to create more database files than needed. It is much better to delete files if it turns out that there are too many, as opposed to not having enough files for the participants of the survey. Similar to ESP, each PDA must be physically connected to the desktop used when creating the survey. The survey is then transferred using some synchronisation tool. If

[3] http://seattleweb.intel-research.net/projects/ESM/iESP.html
[4] http://groups.yahoo.com/group/experience-sampling/

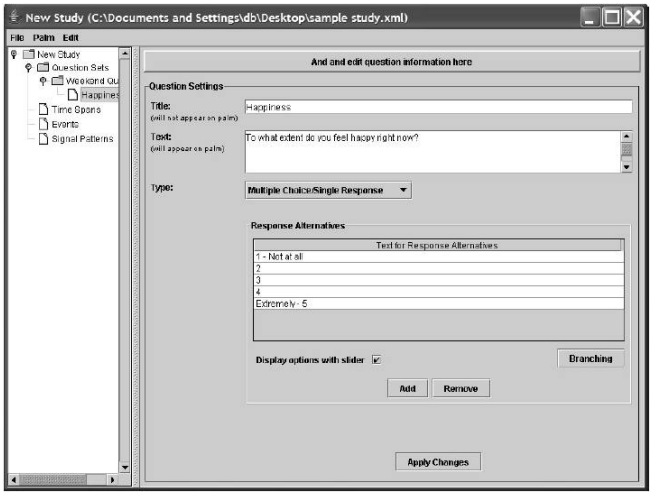


Figure 2.3: PMAT - desktop application

the device has not run a survey like this before, an additional part is also transferred. The second part is the engine that can interpret the content of the database file and run the survey. Collecting the result of the entire survey is done following the same approach: connect the PDA to the desktop computer and perform a synchronisation between the computer and the device. A file with the answers from the user should then be transferred to the desktop computer. Using the PMAT desktop software it is then possible to convert the data-file into a CSV-file which again can be imported into spreadsheet programs for further analysis.

C.A.E.S. - Context Aware Experience Sampling Tool

The last software to be mentioned in the section for open source systems is the one developed by Changing Places Consortium at MIT [1]. In addition to include all features mentioned earlier in this section, it also has the great possibility of adding context information to a situation. In the first round they have tried to incorporate the powerful GPS into the software. According to the web page, contextual information should be possible to retrieve via sensors attached to the participant carrying the device used for a prospective ESM study. The advantage of adding contextual information can be seen when combining the different protocols of ESM. As an example take the event-contingent approach in combination with the interval-contingent approach. For the other systems the participant would notify the ESM software of some event by registering it on that participants handheld device. In some predefined interval after that registration has been done, a question is prompted to the participant. This works fine, but it puts an extra burden on the participant, namely the fact that it has to report when the event has occurred. With C.A.E.S. it would be possible to attach sensors of some kind to detect whether the event actually is happening, then automatically trigger the system to ask the participant a question following the event.

Another difference with this software is that it has been designed for devices running Microsoft Windows Mobile Pocket PC operating system, rather than the Palm Pilot system. Differences between mobile operating systems are outlined in chapter 4. Figure 2.4 shows an image from the application using the signal-contingent protocol of ESM.

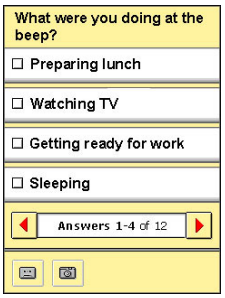


Figure 2.4: C.A.E.S - Signal-contingent ESM protocol

Like most open source software systems C.A.E.S. also has a web-site where it is possible to download the software and view the latest news posted by the developers. From this site and the project site at Sourceforge.net[5], there seems to be little or no activity for this project. The latest news post is dated October 15th 2005.

2.4.2 Proprietary ESM software

In contrast with the software presented in the previous section the software here costs money and technical support is available. Because the software is commercial, little or no documentation exists on how the systems actually work. Only the information provided by the product web sites is available.

PHT Product Suite

PHT Corporation[6] is an American company specialised in patient diary solutions. A patient diary is an electronic diary where patients can report how they feel in certain situations, questions can be asked and the mobile device can read information from sensors attached to the device. The PHT product suite consists of several parts to provide a fully working environment for simplifying the clinical trials process. The three main components of this software suite are: *LogPad*, *StudyPad* and *StudyWorks*. The first one of these, *LogPad*, is the software component designed for mobile devices, more specific, field tested Palm units. The software enables participants to record their experiences and opens for the possibility of attaching sensors to it. Sensors prevent the participant from reporting wrong information, and communicate with the device using wireless communication (type uspesified). Another great feature of this software is that it has a two way communication system. The advantage of this is that a patient can get feedback and advice from the experimenter, because of this feature it is also possible to proactively troubleshoot any problems the patient may encounter.

The second part called *StudyPad* has all the features from *LogPad*, apart from the two way communication. On the other hand, it has a more powerful graphical user interface, which includes body diagrams and visual analog scales.

[5]https://sourceforge.net/projects/caes/
[6]http://www.phtcorp.com

The last part of the product suite, *StudyWorks*, is meant for the experimenter in any survey or simply a doctor wanting to monitor his patient. The software is web-based and is meant to capture, review and manage the results from the mobile devices which are a part of the survey. It is fully integrated with the two mobile products discussed earlier and is said to be an easy-to-use package. There is no information on how the questionnaires are actually created an put on to the mobile devices.

From the product web-site it is possible to download a demonstrating application before determining whether to buy their product or not. The price for the software is not listed, but people interested in their products are encouraged to send an email to get more information.

DiaryPRO and SitePRO

invivo data[7] has developed the *DiaryPro* and *SitePRO* software which are very similar to the previously discussed proprietary ESM software. It is not clear from the web-site if it supports context information retrieval from sensors connected to the device. One thing that is clear, is that the company delivers not only the software, but the DiaryPRO software is installed on specific devices before given to the customer. This also means that the company tailors and sets up the questionnaires based on the customers needs, all done before the customer gets the software package. When this process is done, the customer gets a number of Palm devices with the tailored DiaryPRO software installed. Together with this is a web-based application, SitePRO, which enables the experimenter to monitor the surveys as they carry on.

2.5 EAR - Electronically Activated Recorder

In this section we present a different approach for sampling user experience. The approach exists as a commercial tool and is called The Electronically Activated Recorder (EAR). A big difference with this tool is that it is almost fully automatic. The only thing a participant has to do is to carry a device around with them. The device is a simple recorder which is programmed to start to record the voice from the participant for 30 seconds every 12th minute. To differentiate the participant voice from the voice of other people he or she might converse with, the recording device must be calibrated before it can be used. In [11] some scenarios are described and in the beginning people had a lot of questions because they always had to carry the device around. A set of basic rules were made, the device was only to be turned on during daytime, and some activities were exempted from the survey (e.g. those involving water). Results pointed out that in the beginning of the survey people were impeded by the fact that they had an extra device attached to their body. But after a while they became comfortable with it and useful recordings were made. EAR is significantly different from ESM in that it captures the verbal expressions of the subject, also called the "observer" perspective, while ESM captures how the participant represents their own experiences, the "actor" perspective.

[7] http://www.invivodata.com

2.6 User Interface Representation

One of the challenges of designing an ESM study is to export it out to the participants of a survey. For surveys being entirely web-based this is really not that big of an issue. The reason for this is that the survey will be designed as a web-site, which there are standards for. In most cases the representation of a survey will look the same in all browsers as long as the standards are followed. The situation changes when portable devices are involved. The previously mentioned systems all assume a specific target device. This eliminates the challenges of different device types running the software. That said, they put a requirement on the participants. To be able to take part in a survey they have to be in possession of a certain device. Because of this we have decided to look into the possibility of designing a standardised format that can describe a user interface by using a mark-up language, such as XML. Therefore, in this section we will present some projects that exist when it comes to representing user interfaces using XML. Advantages to why a developer should do this significant separation of the Graphical User Interface (GUI) from the application logic, include how easy the localisation of an application can be changed. The GUI is dynamically loaded, hence to change the GUI simply involves adjusting the XML file given to the parser at runtime. Another advantage of separating the user interface from the rest of the application, is that the GUI can be changed without having to change the logic behind it.

2.6.1 Motivation

Implementing user interfaces are often very time consuming and error prone. As a consequence many vendors have introduced tools to help the developer build such user interfaces faster. These tools are in most cases visual designers where the programmer is given a toolbox and an area to draw the interface in. By dragging and dropping tools from the toolbox and onto the area allocated for drawing the interface, the GUI is created. The tool will generate code for the user interface drawn by the programmer. The generated code is kept in separate files or regions and should not be touched by the programmer. This approach is very efficient and creates a clear separation between the layout of the user interface and the application code. The problem with such a tool is that the user interface created, normally targets one particular programming language (e.g. Microsoft Visual Studio[8] generates Visual C# source code, while NetBeans[9] generates Java source code).

The motivation behind defining user interfaces in a declarative language as XML goes back to the problem where all the designer tools generate code in their own specific programming language. This means that the user interface must be re-implemented if another programming language is decided for the application logic. In an ideal situation, a programmer should be able to use the same description of a user interface between different programming languages and platforms. Consequently, a loose coupling between the application logic and the user interface of an application is achieved.

Describing properties in text files is a normal approach to transfer information between applications written in different programming languages. XML is very powerful when it comes to describing content in text files. Some of the advantages of XML include [12]:

[8] http://msdn2.microsoft.com/en-us/vstudio/
[9] http://www.netbeans.org

- *Easily readable and self-describing*
  An XML document contains tags that describe each element of data. With a good design any developer can quickly get an overview of the content of the file.

- *Interoperability*
  Nothing about XML ties it to any operating system or programming language. The only tool needed is a text editor to actually write the code.

- *Hierarchical structure*
  Related data can easily be added to a node in a document without making it unwieldy.

- *Parser availability*
  There exist a wide number of parsers for different operating systems, hence developers most likely do not have to write the entire parser by themselves.

- *Backward compatibility*
  As long as the written XML is syntactically correct, it will be compatible with earlier versions of the developed application.

- *Data structure representation*
  Complex data structures can be represented within a single XML document.

XML is no different from anything else, hence it has some disadvantages. It requires a parser, which again requires processing. In our case we are going to use a mobile device and processing time is an important issue when it comes to battery on the device. Another important issue is that when data is sent using XML it requires a lot more data to be sent. Describing information in XML requires a set of tags to be defined for each part of information. Adding such tags to the data makes the document structured and relatively easy to read. On the other hand, the size of the tags increases the total amount of data to be sent dramatically. [13] claims that representing data in XML on average increases its original size by 400%. Downloading such a large file using for example a Wireless Local Area Network (WLAN) adaptor, requires a lot more power, and since battery is a limitation for mobile device it is recommended to make the XML representation as compact as possible. Various techniques exists [14] for compression the XML data such that it does not take up so much space, but there is a fine balance between the amount of data to be compressed and the actual process of compressing and decompressing it. In mobile communications a user often have to pay for the number of bytes transferred, hence sending more data than necessary will not benefit the user of the mobile device.

A general opinion is that the advantages of XML oust its disadvantages, and its features are well suited for a standard describing user interfaces in textural form. Therefore, representing user interfaces in XML could allow a developer to design a single user interface for several programming languages and operating systems.

2.6.2 Existing Standards

This section will present a selection of known standards when it comes to describing user interfaces through XML.

XForms

Originally the World Wide Web (WWW) was a platform presenting the user with information. With the increased use of WWW as an interactive platform the HyperText Markup Language (HTML) is no longer sufficient to express all the possibilities of a web site [15]. The GUIs available on the WWW are far more complicated than ever imagined when the web was introduced. The first editions of WWW were aimed at one particular device, namely the desktop computer. With great emergence in technology, web-sites can be accessed via many different devices today, these include mobile phones, smart phones and PDAs. If a web developer has to develop a version for each device a site is to run on, that would have taken up much of his/hers valuable time. As an attempt to improve the features of HTML and lighten the work on the developer, World Wide Web Consortium (W3C)[10] has defined XForms[11]. According to W3C, XForms is the next generation of forms for the Web. In addition to offer the same features as the traditional HTML forms, it has some new and very powerful features. One of the main goals of the successor is to separate the content of a web site from the presentation of it [15].

The work of XForms is work-in-progress, and the latest requirements document for the XForms 1.1 standard is dated December 12th 2006. Because the standard is relatively new to the marked not all computers have yet incorporated an XForms processor. This is needed to interpret web sites that make use of XForms. The processor is responsible for interpreting XML statements, displaying correct user interface components and accepting any inputs from the user. One of the new features of XForms is that fields of a particular form can be marked as required. This eliminates the use for script-based testing after the user has filled in the form. In [15] the architecture, design and implementation of such a processor are described. The processor presented there has been built for desktop computers with Java 2 Standard Edition[12], mobile devices with Java 2 Micro Edition[13] and a digital TV-browser with Java Development Kit 1.1.

The XForms standard is mainly developed as the next generation of the Web forms, but according to W3C the standard can be applied to stand-alone desktop applications as well.

XUL and XAML

Among the available web browsers today, Firefox[14] from Mozilla has a share of 31% [16]. The browser is free open source software in contrast with the proprietary web browser from Microsoft called Internet Explorer. One of the great features of Firefox is called *extensions*. An extension is an additional feature integrated into the browser. Such an extension can be everything from weather information to an integrated FTP client. All of these extensions are written by other developers which follow the standard defined by Mozilla. To describe the user interface for a new extension Mozilla has defined XML User Interface Language (XUL) [17]. One of the main advantages of this XML representation, is that it builds on techniques and technologies familiar to developers, such as JavaScript and Cascading Style Sheet (CSS). Figure 2.5 shows a very simple example of a XUL web site. The screenshot is taken from a tab in Firefox 2.0.

[10] http://www.w3.org/
[11] http://www.w3.org/MarkUp/Forms/
[12] http://java.sun.com/javase/
[13] http://java.sun.com/javame/index.jsp
[14] http://www.firefox.com


```
<?xml version="1.0"?>

<?xml-stylesheet href="chrome://global/skin/" type="text/css"?>
    <window id="simple-example" title="Example"
        xmlns:html="http://www.w3.org/1999/xhtml"
        xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul">
        <button label="Normal"/>
        <button label="Disabled" disabled="true"/>
    </window>
```

Figure 2.5: Simple XUL example

With the release of Microsoft's new operating system, *Windows Vista*, comes Microsoft's reply to Mozilla's open source user interface language. The name is eXtensible Application Markup Language (XAML) and is a part of the new application model in Windows Vista. An important difference between these two languages is that the first is open source and the second is not. XAML will target applications running on the Windows platform only, also web sites developed using XAML will be correctly displayed only in the new version of Microsofts Internet Explorer. Apart from this difference, the two languages seem to have the same features when it comes to representing the user interface in a descriptive way like XML.

SwiXML

Java[15] is a programming language with increasing popularity among developers. Any developer that has created GUI in Java also knows that the code for this often becomes very complex, repetitive and sometimes error prone. SwiXML[16] is an open source extension to the Java language. In short it is a GUI generating engine for Java applications and applets. This extension is different from XUL mentioned earlier because it addresses the javax.swing package of the Java language directly, while the others are a generic specification for all languages. Apart from this clear difference the properties of this library are the same as the others. An advantage is of course the fact that developers familiar with the javax.swing library quickly can understand and learn how to build GUI using SwiXML.

Figure 2.6 depicts a simple GUI example defined using SwiXML. Clearly the XML file is not enough for this application to run. A Java source file is needed as well. This file creates and instance of the SwiXML parser and parses the file given in the figure. Any event handlers for the GUI is also implemented in the Java source file.

2.7 Communication

In a satisfactory ESM system it would be important to enable some communication between the experimenter and the participants of the survey. Since ESM-surveys involve the use of mobile

[15] http://java.sun.com
[16] http://www.swixml.org


```
<frame size="640,480" title="Hello SWIXML World" DefaultCloseOperation="JFrame.EXIT_ON_CLOSE">
    <panel constraints="BorderLayout.CENTER">
        <label LabelFor="tf" Font="Comic Sans MS-BOLD-12" Foreground="blue" text="Hello World!"/>
        <textfield id="tf" Columns="20" Text="Swixml"/>
        <button Text="Click Here" Action="submit"/>
    </panel>
</frame>
```

Figure 2.6: Simple SwiXML example

devices, there must exist a scheme or a paradigm defining how the communication between a desktop computer and a mobile device is to be done. In [18] the authors present an update scheme for dynamic content of web sites for mobile devices. They introduce two proxies, the first one is a proxy server located at a desktop computer, and the second is a proxy at the mobile device. The idea is that the user of a mobile device can mark of any parts of an HTML page that he or she wants to receive updates on. This information is given to the mobile proxy who notifies the server proxy with the user preferences. From that point on it is the responsibility of the server proxy to look for updates within the fields specified by the user. As updates are discovered, these must be given back to the user in some way. They present three ways for a the mobile proxy to receive updates from the proxy server.

2.7.1 Pull-based Scheme

In this scheme the mobile proxy polls the proxy server for updates at fixed times or intervals. If the proxy server signals that updates are available, the mobile proxy will pull these updates out from the proxy server and display them to the user. Such a pull-based approach is comparable to the way most people browse the web today and how the web cache works. The web cache stores copies of web documents passed through it in order to reduce bandwidth usage, server load and perceived lag. When certain conditions are met the web cache will use a copy of the document instead of fetching a new one.

2.7.2 Push-based Scheme

The previous scheme required the mobile device to ask the server for updates. Since battery is a limited resource for mobile devices, the use of wireless communication can lower the battery life. Therefore, an important goal in designing mobile applications is to reduce the use of wireless communication to an absolute minimum. In a push-based scheme the proxy server will send out updates whenever they are available. Following the mobile proxy receives updates and displays them to the user. This scheme requires that the proxy server maintains knowledge on how to contact the mobile device. In a typical desktop-to-desktop communication scheme this would mean storing the Internet Protocol (IP) address of the recipient and use this to contact another computer. A mobile device also has an IP address, but in contrast with a desktop computer, the IP address of a mobile device will change according to the current network provider it is

connected to. Consequently, if the push-based scheme is chosen the mobile proxy would be required to report to the proxy server every time the IP address of the device changes.

2.7.3 A Push-Pull-based Scheme

This last scheme combines the two already mentioned approaches. Every mobile device needs a SIM card, which is a removable smart card that securely stores the key identifying a mobile phone service subscriber. Using the phone number associated with the SIM, card any device or computer can contact the particular mobile device using the Short Message Service (SMS). Sending a message using this service is limited to 160 characters, of course new devices provide implementations that allow the user to think that he or she can send and receive larger messages. The real situation is that the software on the device splits the message up into chunks of 160 characters and the receiving end will combine the different parts into one message before delivering it to the user.

The approach is initiated as the server sends an SMS message to the mobile proxy. This message signals that an update is available for download. Upon receipt of such a message the mobile proxy will know that new updates are available, hence it will try to pull updates from the proxy server from for example a predefined web-service method.

2.8 Summary

In this chapter we presented the general concepts of Experience Sampling Method (ESM) and gave examples of existing systems within this field. The clear tendency was that open source ESM software was not as functional as the proprietary ones. On the other hand, the proprietary systems required that the user put a lot more money into their budgets and most of the system did not open for redesigning a survey without having to contact the vendor.

The last part of this chapter discussed different approaches taken when it comes to representing user interfaces using XML formatted documents. Finally, we presented some approaches taken by others when it comes to communication between a mobile device and a desktop computer. We saw here that the battery capacity on a mobile device plays an important role when a communication scheme is to be defined.

Chapter 3

Requirements

This chapter presents the main system goal and requirements identified for the two applications that together constitutes the software tool mentioned in the problem definition of section 1.2. The requirements have been elaborated in collaboration with NST, Telenor and the department of psychology at University of Tromsø[1], Norway.

3.1 System Goal

The system shall provide an experimenter of a survey with a software tool enabling him or her to take advantage of the ESM approach for running surveys. Because this software tool consists of two main parts; the system for creating a survey and the system that runs the survey on a device, we have chosen to split the requirements specification into these two categories as well. Figure 3.1 illustrates the relationship between the two systems. From this point on the desktop application will be noted *esmDesk* and the platform installed on the mobile device will be annotated *esmMobile*.

[1] http://uit.no/psykologi/

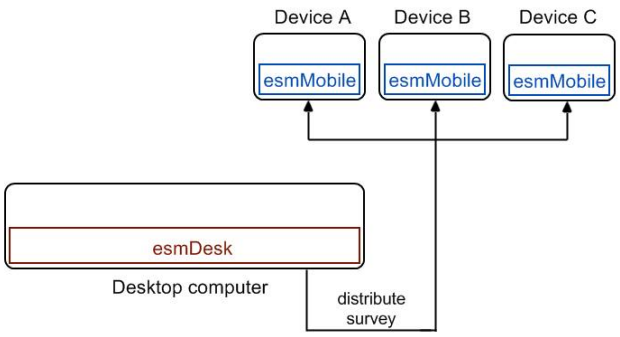


Figure 3.1: Relationship between esmDesk and esmMobile

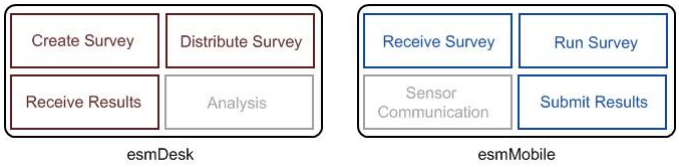


Figure 3.2: System Focus

3.2 System Focus

Figure 3.2 depicts the two applications constituting the software tool. For each of them we have identified four parts that we consider to capture the main functionality of a fully functional software tool based on the ESM approach. There are two grey boxes in the figure illustrating that they should be a part of a final system and not our prototype system. The first grey box called *Analysis* has to do with the interpretation of the results from a completed survey. Such analysis may include the presentation of different charts based on the results. The second grey box is the functionality needed on the mobile device to be able to read data from a set of given sensors. This functionality is needed in a fully functional system because the experimenter shall not only be bound to asking question to the participant, he or she should also be able to instruct the mobile device to read out a value from a sensor at a specific time. Based on that value the questions can be asked or values from other sensors read.

3.3 Functional Requirements

Functional requirements describe the intended behaviour of a system, they are statements of services the system should provide, and they may also explicitly state what the system should not do. The widespread practise for capturing functional requirements is through *use cases*. A use case is a scenario based technique that defines a set of interactions between external *actors* and the system to be developed [19]. An *actor* is defined as a party outside the system that interacts with the system. For our system we have keyed out two main actors; the *experimenter* and the *participant*. The experimenter is the person interacting with esmDesk, while a participant is the user of esmMobile, consequently answers the questions of a survey created using esmDesk. In the following two subsections we will present a description of the identified use cases for the two systems respectively. Each specified use case will contain at least one functional requirement that can be found in appendix A. The term *element* is frequently used when describing the different use cases in this section. An element can be either a question or a measurement method (attached sensor).

3.3.1 esmDesk

The use cases listed in this section are for the actor *experimenter*. Figure 3.3 is a use case diagram for all the use cases presented here. In this figure an actor named *esmMobile* is included to clarify the relationship between the two systems. In addition to the descriptive name of the use case from figure 3.3, each use case is given an identifier D-<number>. The letter D stands for *desk*.

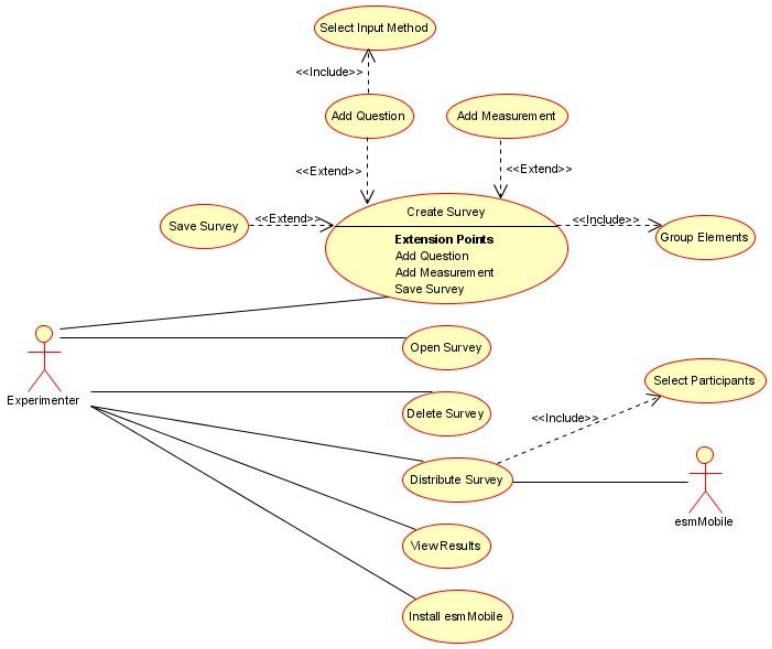


Figure 3.3: Use case diagram for esmDesk

D-1 Create Survey

Allowing the user to create a survey is a comprehensive use case, and as figure 3.3 shows several other use cases are related to it. The use case requires that the experimenter in a survey is familiar with the ESM approach. The building blocks of a survey are groups and elements, and the survey is created as an experimenter creates groups and adds elements to them. Elements can not reside outside a group, but the same elements can exist in multiple groups. Any time during the creation of the survey, the experimenter can choose to save, close or distribute the survey. Closing the survey automatically saves the preferences set by the experimenter. Requirements are listed in table A.1.

D-2 Add Question

A survey is made up from a set of elements, and a question is such an element. The experimenter must have the opportunity to add a question to an already created group within an opened survey. As the question is added to the group, the experimenter is required to specify a set of properties for that particular question. These properties are related to how the participant should answer the question, the amount of time given to answer the question and finally a specification of any other elements in the survey that the new question depends on. Requirements are listed in table A.2.

D-3 Select Input Method

An input method defines how a participant shall answer a question in a survey, hence it must be specified as an experimenter adds a question to a survey. Based on requests from psychologists

at University of Tromsø Norway, four input methods have been defined for the prototype to be developed in this thesis. These input methods are:

- *Yes/No*, participants answer either *yes* or *no*.
- *Multiple Choice*, participants select an answer among several possibilities.
- *Slider*, participants indicate their answer by sliding a track-bar between two extremities.
- *Free text*, participants answer using their own words.

At first glance some of these methods might look the same. The Multiple Choice method can define answers to be 1-7 and the Slider can also have its boundaries as 1 and 7, consequently one could expect the same answer from participants choosing either methods. Psychologists know that how participants are instructed to answer a question may impact on the answer they actually give. Therefore they want to be able to select different methods for distinct questions. Requirements are listed in table A.3.

D-4 Add Measurement

Even though reading data from sensors is not a part of this thesis we have chosen to include its requirements to make future development of the prototype easier. Basically adding a measurement is similar to adding a question, apart from the fact that the experimenter has to specify other properties. Input method and time-out is not required because the execution of such an element does not require interaction with the user. Requirements are listed in table A.4.

D-5 Group Elements

Grouping elements is important to allow parts of the survey to be repeated with preset intervals, consequently following the interval protocol of ESM. Along with creating a group the experimenter shall specify a number of settings that defines when to start the first element of the group, how often to run elements in the group and the execution order of elements within a group. Requirements are listed in table A.5.

D-6 Save Survey

An experimenter is likely to spend days, maybe weeks creating a survey before it is ready for distribution. Therefore it is important to allow him or her to save the survey any time during its creation. Saving the survey typically involves saving all the added groups and elements, and of course their properties. Requirements are listed in table A.6.

D-7 Open Survey

The previous use case defined the possibility of saving a survey. In order to benefit from the saving, an experimenter must be able to open an already created survey. The application should present a list of stored surveys for the experimenter to select among. Once a selection has been made the system shall read out all the properties of the selected survey from a persistent storage and display it to the experimenter. Requirements are listed in table A.7.

D-8 Distribute Survey

Creating the survey is the first step in the ESM process. The next step is to distribute the survey to a set of participants. The process of distributing the survey shall require the experimenter to specify the participant to take part in the survey, then the system shall notify the device of these participants that a new survey is created and should be run. Requirements are listed in table A.8.

D-9 Select Participants

In order to select participants, the system shall require a survey to be created and saved. A survey must at least contain one group with one element, otherwise it would not make sense to distribute it out. Selecting participants involves specifying the phone number of which devices to distribute the survey to. Requirements are listed in table A.9.

D-10 Delete Survey

The experimenter shall be given the opportunity to delete any created survey. Deleting a survey means removing all the preferences set by the experimenter including the results sent in by participants. Requirements are listed in table A.10.

D-11 View Results

The third step in the ESM process is to collect results from participants and give the experimenter the opportunity of viewing the results. Without being able to view the results, an experimenter cannot determine the outcome of a survey. Once all results in a survey is collected by esmDesk, the system shall notify the experimenter such that he or she knows when it is possible to view the results of a previously created survey. Requirements are listed in table A.11.

D-12 Install esmMobile

Because the mobile application is required to store information an application must be installed on the mobile device before it can receive and run a survey created with esmDesk. If the application is not installed on the device, the experimenter shall be able to do so from esmDesk. By providing a download service to all devices, the system can have an option that lets an experimenter specify a set of device that shall receive a notification containing a reference to where the required application can be downloaded. Requirements are listed in table A.12.

3.3.2 esmMobile

The use cases listed in this section are for actor *participant*. Figure 3.4 is a use case diagram for all the use cases presented in this subsection, like figure 3.3 this figure also includes an actor which represents the esmDesk system, and illustrates the relationship between the two systems. A third actor called *Mobile device* is also included to illustrate that the installation is initiated by esmDesk but completed by the mobile device itself. In addition to the descriptive name of the use case from figure 3.4, each use case is given an identifier M-<number>. The letter M stands for *mobile*.

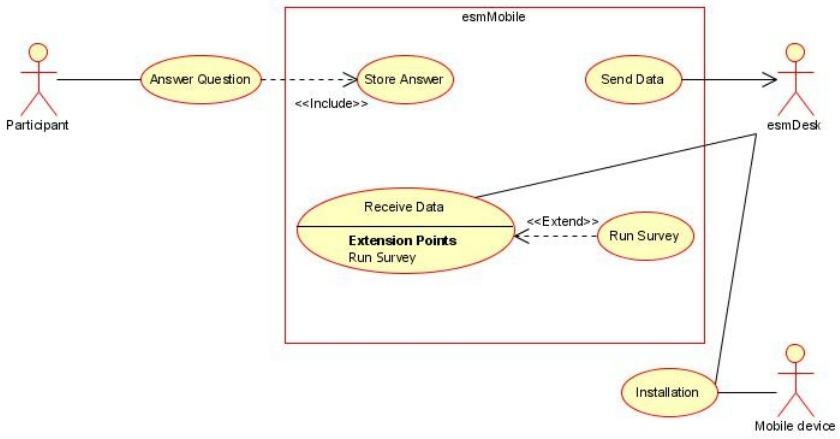


Figure 3.4: Use case diagram for esmMobile

M-1 Run Survey

The main purpose of esmMobile is to run a survey that has been created using esmDesk. Running a survey is about asking a participant the questions defined by the experimenter at the correct times also specified by the experimenter. The system is responsible for repeating the questions if so is specified. As a participant answers the different questions, esmMobile shall store them until the survey completes. A survey completes when the last element in a survey has been executed and the result is collected. Results are then transferred to esmDesk. Requirements are listed in table A.13.

M-2 Answer Question

A participant answers questions according to the input method that is set by the experimenter in esmDesk. Requirements are listed in table A.14.

M-3 Store Answer

Mobile devices are likely to be turned off for a number of reasons; empty battery, areas where mobile phones are prohibited and so on. For this particular reason it is very important for the system to store all answers persistently as they are received. Requirements are listed in table A.15.

M-4 Receive Data

When a new survey has been created by esmDesk, it will send a notification to all selected participants. To be able to get this notification esmMobile must be able to receive data following a particular format agreement that exists between esmDesk and esmMobile. Upon receipt of the data esmMobile determines the type of notification and takes action accordingly. Requirements are listen in table A.16.

M-5 Send Data

Sending data is the process of collecting and organising the results given by the participant from the persistent storage, and then submit these results back to esmDesk. Requirements are listed in table A.17.

M-6 Installation

The installation is related to the use case D-12 of esmDesk. The operator of the device observes that the notification sent by esmDesk has been received on the mobile device. As mentioned this notification contains information on how to download the software required to run surveys. After downloading the software it gets installed and the device is ready for use with surveys created using esmDesk. Requirements are listed in table A.18.

3.4 Non-Functional Requirements

In contrast with functional requirements specifying system behaviour, non-functional requirements outline criteria which can be used to judge the operation of a system. Since the software developed in this thesis is a prototype, most of our focus has been put into identifying functional requirements for the two systems. Nevertheless, we will present some non-functional requirements for the two systems as a whole in this section. All requirements specified in the following subsections are given a unique identifier that starts with Non-Functional Requirement (NFR) followed by an integer.

3.4.1 Usability

Requirement ID	Description
NFR-100	Both parts of the system shall provide a consistent user interface, menus and commands across the different parts of the two software systems to assist new users in getting started. *This is particular important for esmMobile, because it is likely that new users are frequently introduced.*
NFR-101	The systems shall follow ISO 9241-11 *Guidance on usability.*
NFR-102	The participant in a survey shall never have doubt on how to answer a question.

Table 3.1: Usability

3.4.2 Implementation

Requirement ID	Description
NFR-103	esmDesk shall be built on top of Argos middleware platform.

Table 3.2: Implementation

3.4.3 Interoperability

Requirement ID	Description
NFR-104	A survey created using esmDesk shall be represented in a mark-up language. Potentially the survey can then run on any platform that knows how to interpret the format.

Table 3.3: Interoperability

3.4.4 Documentation and help

Requirement ID	Description
NFR-105	A user manual addressed at the experimenter shall be provided. The manual shall include: • A presentation of the functionality of both esmDesk and esmMobile. • A simple walk through on how to create, save and distribute a survey.
NFR-106	The systems shall provide the participant with a built-in help feature.

Table 3.4: Documentation and help

3.5 Summary

In this chapter we have presented the functional and some non-functional requirements of the two systems to be developed in this thesis. We focused on presenting the use cases of the two systems in this chapter because the software is to be developed as a prototype, and we encourage the reader to look in Appendix A for a complete listing of all identified functional requirements for the described use cases.

Chapter 4

Technology

This chapter presents some technologies relevant to the domain for the prototype development of this thesis. In the previous chapter we presented the requirements identified for the two different systems that constitute the ESM software to be designed and implemented. Here, we will make the same separation when talking about technologies. We start out by presenting the server side, then the client side which in this particular case is a mobile environment.

4.1 Server Side

The application needed on the server side, also referred to as esmDesk in chapter 3, is required to have a rich user interface, ability to save data for later use, provide communication facilities and finally it must be able to interact with a web server such that it can put up files for downloading by clients.

4.1.1 Application server

With today's globalization, business leaders are challenges by other companies when it comes to a more rapid development. It is required that new systems and services are designed, developed and made available to customers as soon as possible. Any company wants to be the first to offer a service, consequently capturing customers from other competitors.

Up to the late 1990's the trend in the computer industry was to develop applications to run on a customers local machine, this applies to the term *thick client* where the desktop computer runs the entire application. From the late 1990's the concept of *thin clients* was believed to be on the way back. Only this time with a more extensive use of advanced graphical user interfaces. This thought introduced the concept of application servers and Component-Based Software Development (CBSD) [20]. A goal of CBSD is to provide a library of pre-built and standardised software components available to fit into some application domain, where an application is assembled using a subset of these components. A component is an independent module which implements some specific functionality in a software system, and its specification describes how to assembly, use and manage the particular component. Adopting the principle of CBSD application servers provide services and infrastructure assisting the developer in creating middle-tier applications [21]. Such applications are typically responsible for the exchange of messages between a persistence layer, also known as data-tier, and the presentation layer or a thin client application. An application server is meant to lighten the development process

for an application developer by allowing him or her to focus on the business logic in the application. The set of services an application server offers to the programmer, varies between the different implementations available on the market. Two of the most popular applications servers are JBoss[1] and WebSphere Application Server[2]. A feature supported by more and more application servers is the Service Oriented Architecture (SOA) technology which has become very popular over the last few years.

Service Oriented Architecture

The *web service* technology was a successful attempt to define a communication standard for software applications running on different computers connected in a network. In short, a web service relies on a standard network protocol, namely Hyper Text Transfer Protocol (HTTP), and applications communicate with each other by exchanging messages defined using XML. With the emergence of the web service technology platform, the major software vendors opened their eyes for Service Oriented Architecture (SOA) [22]. SOA technology is centred on the follow key concepts [23]:

- *Service*
  A service is an exposed piece of functionality with a platform independent interface. It is possible to dynamically load and invoke the service. The service is also self-contained in that it maintains its own state.

- *Message*
  Service providers and consumers communicate using messages. Since the interface defining a service is platform independent, it is common to construct messages using XML. An advantage of XML is that there is nothing in it that ties it to a particular platform or programming language.

- *Dynamic Discovery*
  Provide consumers with the ability to search for service providers.

- *Web Services*
  Because web services are built on top of the well-known and platform independent protocols HTTP, XML, Universal Description, Discovery and Integration (UDDI), Web Service Definition Language (WSDL) and Simple Object Access Protocol (SOAP), they are very attractive for SOA. SOA requires that a service can be dynamically discoverable and invokable, which is fulfilled by the use of UDDI, WSDL and SOAP.

Essentially SOA is a collection of services communicating with each other either by exchanging simple messages or involving two or more services in a collaboration to achieve a goal, or coordinating some activity. The basics of SOA is illustrated in figure 4.1.

One of the great strengths of SOA is the high degree of flexibility achieved by using it. Not many years ago most software systems were designed and implemented as ad-hoc solutions, making them very inefficient and both difficult and expensive to change as new requirements arose from stakeholders. Services in SOA are defined using interfaces, thus making it easy for

[1] http://www.jboss.com/products/jbossas
[2] http://www-306.ibm.com/software/webservers/appserv/was/

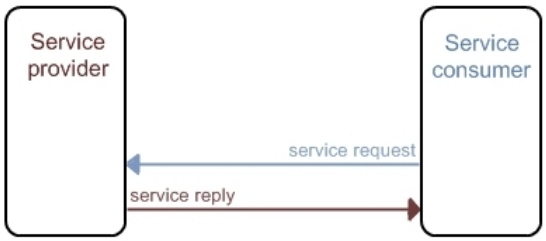


Figure 4.1: Basic Service Oriented Architecture(SOA)

service providers to change their implementation of a service without affecting the output given to the consumer, consequently achieving high flexibility. Another important aspect of SOA is that it drives the company's focus more towards business-level activities and interactions rather than technical sub-tasks. The technical sub-tasks are solved by making use of already existing services deployed in the SOA.

Argos

At university of Tromsø, Norway, there is an ongoing project that aims at developing a personal middleware system called *Argos*[3] [24]. The platform is meant for personal use as opposed to the enterprise application servers mentioned earlier, JBoss and WebSphere. Because Argos utilises Java Management Extensions (JMX)[4], we will give a short introduction to JMX before presenting the architecture of Argos.

JMX allows monitoring and management of Java resources on the same computer or across different computers in a distributed fashion. From Java's own description of JMX it is said that the typical uses of JMX can be summarised into the following:

- Consult and change application configuration.
- Collect statistics about application behaviour and make them available.
- Notification of state changes and erroneous conditions.

A resource made available through JMX is called a Managed Bean (MBean). There are three types of MBeans; standard, dynamic and open MBeans. The standard MBean is the least complex to implement. Three steps are involved in making a standard MBean. First, an interface defining the attributes and the available methods is defined, then a class is defined implementing the interface, and becomes the actual MBean. The last step is to register the newly created MBean with the MBean server which is a part of JMX. Once registered with the MBean server, the object can be monitored and managed through the interface defining the MBean. The dynamic MBean is more complex compared to the standard MBean, but more flexible and powerful because it will expose a management interface at runtime containing generic methods such as *getAttribute*, *setAttribute* and *invoke* for operations.

[3]http://argos.cs.uit.no/, named changed from APMS to Argos in the latest version
[4]http://java.sun.com/developer/technicalArticles/J2SE/jmx.html

Figure 4.2: Argos - architecture

The Argos middleware platform is a light-weighted microkernel entirely written in Java[5]. It utilises JMX and offers a component model, lifecycle management and hot deployment of new services to the system. Services are categorised into *system* and *user* services. A system service is a service that extends the functionality provided by the Argos Core. Such a service can be database abstraction, a web server implementation and so on. A system service also differs from a user service in that it can subscribe to events fired by the Core itself. Such events are typically not of interest for a user service. A user service can use features provided by one or more system services to complete its final task. The main parts of the Argos architecture is illustrated in figure 4.2. As of writing the Argos middleware platform is shipped with the following system services:

!!hibernate
Hibernate[6] provides an easy and powerful way of working with databases following an object-oriented idiom. Hibernate provides relational persistence for both Java and .NET. That is, it handles the mapping of objects in the programming language down to tables in the database. Advanced techniques such as association, inheritance, polymorphism, composition and collections are supported. Hibernate has been tested and works with a large number of databases including Oracle, Microsoft SQL Server 2000, MySQL, PostgreSQL and more. Currently, the database used in Argos is Apache Derby[7] which is also listed among the databases supported by Hibernate.

!!jmxconnector
This !!service provides the core with the necessary functionality for utilising JMX.

!!jetty6
Jetty 6[8] provides Argos with an open-source, full-featured web server implemented entirely in Java.

!!webservice
Web service is a powerful technology where information is exchanged in a standardised format, independent of the platform an application is running on. The protocol followed for exchanging the information is called Simple Object Access Protocol (SOAP), and is a light-weight W3C

[5]http://java.sun.com/
[6]http://www.hibernate.org/
[7]http://db.apache.org/derby/
[8]http://www.mortbay.org/

standard for the exchange of structured information in a decentralised and distributed fashion. By incorporating Apache Axis[9], this system service provides Argos with the ability to create and deploy web service applications. Using such a standardised approach, Argos is able to exchange information with any application, neither limited by the programming language it is written in nor the platform it is running on.

We believe Argos is well suited for our prototype development because we can benefit from its set of system services. By directly utilising these system service we can focus on the business logic in the application, instead of having to worry about implementing support for web services, database handling and more.

4.1.2 Graphical User Interface

The prototype development involved in this thesis must provide a rich and powerful GUI. Having a poorly designed GUI can cause people to abstain from using it. That said, everyone with experiences in GUI development knows how much time is required, and that the actual code often becomes very repetitive and difficult to maintain. Many solutions exist where the developer is offered help when designing user interfaces. Examples of such help include visual designing of the user interface and code generation. In this section we will give examples on how user interfaces can be developed using some existing techniques.

Simple editor
This technique requires only a simple editor and the Software Development Kit (SDK) for the programming language to be used installed on the computer. Here the programmer has no help apart from the already defined GUI components in the programming language. This approach is time consuming and requires high knowledge of how programming languages design user interfaces. There is no doubt that the programmer has full control using this approach. A clear separation between the application logic and user interface can be made by the programmer since no code generation is done. Another advantage is that no special editor is required to be able to read and understand the source files for the project.

NetBeans and Eclipse
The previous technique is not bound to any programming language, in this paragraph and the next we will present available technologies for Java and Microsoft .NET framework respectively. NetBeans[10] and Eclipse[11] both provide a very powerful Integrated Development Environment (IDE) for building Java applications. They have form designers where the programmer can create user interfaces by dragging and dropping components to visually construct Java applications. NetBeans' designer is preferred over the one available for Eclipse. Developing the interfaces requires less time, NetBeans has more advanced features and is more intuitive to use. The code behind the designed user interface is generated by the IDE software and is tagged as "do not touch"-parts in the particular classes. A negative aspect of this approach is that the programmer will have a mix of generated code and programmer specific code. However, the programmer will probably save a lot of time with help from a visual designer.

[9] http://ws.apache.org/axis/
[10] http://www.netbeans.org/
[11] http://www.eclipse.org/

Visual Studio 2005
Visual Studio 2005[12] is a very powerful IDE from Microsoft for .NET Framework and .NET Compact Framework. Like NetBeans, Visual Studio provides the programmer with a visual designer where the user interface can be drawn. An advantage this editor has over NetBeans is that the generated code is kept in separate files. Due to the fact that these files are regenerated by the visual designer, the programmer should never modify their content. This is possible because of a language feature of Visual C# where a programmer can split a class definition into more than one file. From this feature the programmer gets a clean separation between the generated code and the code written by the programmer. On the other hand, without Visual Studio installed, it can be very difficult to get hold of how the code actually works by just looking through the files.

Deciding what approach to use is often determined by the programming language chosen and the time available to develop the software. It is also worth mentioning that NetBeans and Eclipse are both free and can be downloaded from their respective web sites. Visual Studio comes in several editions, and the simplest one called Express is also free and obtainable from Microsoft's home pages. The edition has limited functionality and to get a fully functional copy of Visual Studio it must be purchased from Microsoft Corporation.

4.2 Client Side

Essentially ESM strives at asking a participant questions when he or she is about to experience something, is in the middle of a situation or immediately after some event has happened. A mobile device is a natural carrier for these questions, hence the client side in our prototype development targets mobile devices. Developing applications for mobile devices introduce many challenges that must be taken into consideration as we mentioned in section 1.3 in this document. Roughly, a mobile device can fall into one of the following categories: a mobile phone, a smart phone or a PDA. The first, mobile phone, is a very simple device meant for calling and sending SMS and/or MMS messages. A smart phone has everything a mobile phone has, and in addition it includes an advanced Personal Information Manager (PIM) and email options. The last type, the PDA, is a small portable computer. In the beginning, PDA devices were not equipped with the ability to make phone calls, but the latest editions include most features offered by mobile phones and smart devices. For our purpose we will focus on the technologies available to PDA devices.

4.2.1 Wireless Technologies

A wide range of wireless technologies exists and new technologies are introduced to improve transfer rates between devices. Roughly, the technologies can be divided into long and short distance. The latter of these is typically limited to a few meters of range, examples of such technologies are Bluetooth™and Infrared™. Long distance wireless technologies are depicted in figure 4.3, also illustrating the evolution within this area. In addition to the standards for mobile communication from figure 4.3 many of the latest mobile phone are also equipped with a WLAN adaptor.

Among the technology transitions, the change from Circuit Switched to Packet switch and the

[12] http://msdn2.microsoft.com/en-us/vstudio/aa718668.aspx

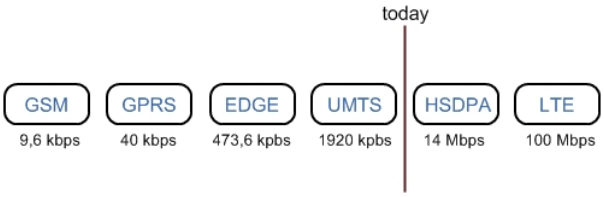


Figure 4.3: Wireless technologies

introduction of Universal Mobile Telecommunications System (UMTS) is worth mentioning. Global System for Mobile communication (GSM) is the first and most popular standard for mobile communication and is based on the Circuit Switched Data connection approach. After a connection has been established a circuit reserves required bandwidth for the entire life time of the connection. A circuit like this is very convenient for carrying phone calls as data is sent throughout the life of the connection. With the introduction of instant messaging and web browsing, the Packet Switch data connection approach was brought forward. Here a circuit reserves bandwidth only when data is received or transmitted, hence the technology allows for better sharing of the bandwidth, and can provide higher transfer speed.

UMTS is a new mobile communication technology not applicable to GSM[13]. The main difference between UMTS and GSM is the air interface. Using UMTS a mobile device can connect to several types of networks, including the Internet, ISDN or another UMTS network. Since the technology is emerging most countries are not fully equipped with the necessary base stations to forward requests in a UMTS network. As a consequence, UMTS phones are dual mode, meaning that in areas where UMTS is not available the phone uses GSM. In areas with both technologies, UMTS is preferred over GSM.

4.2.2 Exchanging Information

The different wireless technologies within the mobile domain open for other ways of exchanging information than just speech.

SMS

SMS was the first technology where the consumer was allowed to compose a short message on their mobile device and send it to one or more recipients. The expectations to this technology were very low, and many people condemned it to die even before it had been launched. Too many of the experts surprise, the use of this technology literally exploded and today all new devices support this form of message communication. The technology itself allows for sending a chunk of 140 bytes in binary format over the GSM network or as General Packet Radio Service (GPRS) packets using the GPRS technology. A single SMS is bound to the maximum of 160 digits, but new devices have software which automatically splits messages into chunks of 140 bytes on the sender side. When received, the receiver software merges the messages together before delivering them to the user.

[13]http://www.umtsinfo.co.uk/gsmvsumts.html

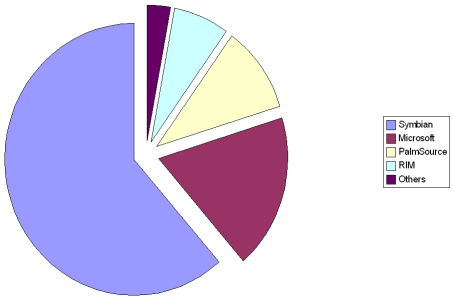


Figure 4.4: Mobile OS Market Share 2005

MMS

Multimedia Messaging Service (MMS) is the next generation of SMS messages. People using SMS tried to express themselves by drawing figures using the different symbols from the alphabet. The next step in the evolution was to allow the consumer to attach multimedia content in addition to the text in SMS messages. Using a voice recorder and/or digital camera which often are a part of the mobile device, the consumer can for example take a photo, attach some text to it and send it. The channel used for transporting SMS messages is to narrow to send multimedia content, hence sending multimedia message requires the use of the Wireless Application Protocol (WAP) protocol, because the minimum speed required is 14.4 Kbps. Unlike SMS, MMS is *not* bound to a maximum number of bytes before sending, this is to cope with future requirements.

In addition to these two device-to-device approaches for exchanging information, new and more advanced devices such as PDAs, open the possibility of using web services as a communication channel between a server and a mobile client.

4.2.3 Mobile Operating Systems

As with desktop computers, various operating systems are available for mobile devices. The mobile Operating System (OS) determines a phone's features, performance and security. Which system to select depends on the user needs. In the world of business it is normal to look beyond the phone capabilities of text messages and normal phone calls, and search for the ability for handling word-processing, spreadsheets and of course business specific applications. Analysis have shown that when considering different operating systems, advanced security support has become an important issue for many users [25]. Figure 4.4 shows the market share of the different operating systems available for mobile devices. The graph is taken from Canalys[14], a provider of consulting and market analysis for the IT industry.

Symbian

Symbian OS[15] is, as illustrated in figure 4.4, the OS with the highest market share in the world. The system is owned by Ericsson(15,6%), Nokia(47,9%), Panasonic(10,5%), Samsung(4,5%),

[14]http://www.canalys.com
[15]http://www.symbian.com

Siemens(8,4%) and Sony Ericsson(13,1%). There are many reasons to why this OS is so popular, the most important ones are probably its architecture and openness. Symbian has a plug-in architecture which makes it easy for manufacturers to add their specific functionality on top of the OS, for example to support device differentiation. It is also referred to as an open OS. Symbian is not open source, but it is open in the way that its APIs are public available. Due to its popularity the platform has become a target also for various viruses. Applications can be developed using several programming languages, these include Java, C++, C# (Symbian version, plug-in available for Microsoft Visual Studio).

Microsoft Windows Mobile

Microsoft's mobile OS is called Windows Mobile. Currently, they have just released version 6.0. The OS can be seen as a subset of the normal Microsoft Windows operating system for desktop and laptop computers, and comes in two flavours; a smart phone edition and a pocket pc edition. Both versions include support for multimedia such as media player and instant messaging. In addition to this, the pocket pc edition also includes PDA releases of the popular word processing program Microsoft Word and Microsoft Excel. The operating system supports multitasking and is one of the most popular mobile operating systems in the United States of America today. One of the reasons for its popularity is that the transition to using a mobile device is easy because of the similarity to a desktop computer (e.g. start button). The primary programming language for applications running on the Windows Mobile platform is Visual C#, it is also possible to use Visual C++ and Visual Basic .NET.

PalmSource

Palm OS is well established among mobile users, being the first commercially successful OS for PDAs. Today it can not compete with new systems such as Symbian and Windows Mobile, because it lacks support of multitasking, 3G and more. Despite of its limitations, it is by many considered to be the most easy-to-use operating system available for mobile devices. The Palm OS is mainly developed as a personal organiser, thus it is not like a standard computer OS. It focuses on the management of an owners personal information like contacts, calendar and tasks. As mentioned the Palm OS up to version six does not support multitasking, which basically means that an application closes instead of running in the background as on Microsoft Windows Mobile. Consequently, the user is not able to swap between running applications, which is possible on a desktop computer. Devices running Palm OS are best suited for users that plan on using their device for personal organisation only.

RIM

Research In Motion (RIM) is the number one smart phone producer in the United States of America and number four mobile operating system in the world [25]. Their product is the Blackberry[16] and focus on providing the business user with applications that go beyond just reading emails. The latest release of Blackberry supports a total of more than 1500 business applications. They are all written in Java making it possible to run them on various models and work over different cell technologies.

[16] http://www.blackberry.com/

Linux

Greenphone is a mobile device developed by Trolltech[17] and comes with a Linux kernel version 2.4.19. Mobile operating systems built on Linux belong to the *Others* group in figure 4.4. Having a Linux based operating system means low development costs, no licence costs and the most important fact that there is a far reaching community of developers ready to write applications for the mobile platform. As of today a mobile application written for one Linux platform is most likely just working for that platform, the reason is that no standard exists for how a mobile Linux operating system should be. This limitation makes it difficult to share mobile applications between different operating systems built on Linux.

Mac OS X

At Macworld Conference & Expo 2007 a new product from Apple[18] was presented, namely iPhone. This mobile device is Apple's first step into the mobile device market. The mobile device will run Mac OS X, and is expected to be released in late first half of 2007. Many people have expectations to what this device is capable of when it reaches the market. A closer inspection has shown that it does not support 3G and the software allowed for the device is to be under strong control by Apple and Cingular[19].

4.2.4 Mobile Architectures

Mobile applications are built around various assumptions differing from one device to another. Software can be built to run on different devices, to operate on many different networks or possibly integrate with different back-end systems. There are two main categories for mobile architectures, namely *Thin clients* and *Smart Clients* [26]. Basically the difference between these two is that a thin client has no software installed, and a mobile device typically interacts with an application residing on the server using the browser available on the device. For a smart client, the situation is different. Application and data is kept both on the server side and the client side. The application on the client side is able to run entirely by itself for a period of time before it has to synchronise or exchange information with the server. This approach does not demand network connectivity at all times, which is required for the thin client. In situations where the mobile application can be split into two independently and logical parts, it is recommended to follow the smart client approach. An important limitation to take into consideration for the smart client approach is the deployment and portability. Smart client software must be installed on the device before it can be used, also they are typically developed for a particular operating system, meaning that they are bound to only a set of mobile devices. Thin clients require no installation, and is developed using a standardised format such as HTML and can be interpreted by the majority of mobile devices since they are equipped with a web browser.

4.3 Interoperability

The term interoperability refers to the ability for different programs to exchange messages in a well defined way, and use the traded information to perform some operation specific to the

[17] http://www.trolltech.com
[18] http://www.apple.com
[19] http://www.cingular.com

application. We are going to establish communication between a desktop application and a set of mobile devices. It is important that the format used to exchange information is not tied to any programming language or platform specific properties. One of the reasons here is the large number of available operating systems in the mobile application domain, as presented in section 4.2.3. A popular approach to define formats between applications is to use the declarative language XML. Because of the rapid emergence of XML as a data transport format, it is very important that the structure of an XML document is created and stored in a flexible and organised manner.

4.3.1 DTD

Document Type Definition (DTD) defines valid building blocks for an XML file. The definition of DTD is either included at the top of an XML file or it is possible to make a reference to an external file containing the DTD. Elements and attributes are specified in the definition, along with their applied restrictions. Using DTD, an XML file can carry a description of its own format, and independent groups of people can agree on a standardised format that they should use when exchanging information. DTD can also be used to verify the data in your own files.

4.3.2 XML-Schema

The successor of DTD is said to be XML-schema and was approved as a W3C standard in May, 2001 [27]. The goal of an XML-schema is the same as with DTD, namely to define valid building blocks for an XML file. The next important aspect of XML-schema is that it expresses Object Oriented (OO) design principles found in common OO programming languages into its specification, and also provides rich data typing support similar to the data typing functionality found in relational databases. According to W3C they believe that XML-schema will be used in most web applications in the near future as opposed to DTD. The reasons for this is that a schema is more extensible to future additions, it is possible to use a predefined set of data types in addition to creating your own, and it is written in XML. The fact that the schema is written in XML means that existing XML parsers can be used in conjunction with schema validators to provide well-formedness and validation facilities.

4.3.3 Design patterns

As with object oriented programming, various design patterns exists when creating XML-schemas. Using a design pattern helps the XML-schema to achieve higher degree of flexibility and extensibility. There are basically three design patterns addressing decoupling and cohesiveness that have emerged recently; *Russian Doll*, *Salami Slice* and *Venetian Blind* [28] with increased extensibility respectively. A component in an XML-schema based on the first pattern includes all relevant information to that component, hence making it inaccessible to anything else. The advantage of this approach is that the schema is self-contained and does not interact with other schemas. Changing the schema is decoupled from schema components because the schema is not visible to other schemas. The major disadvantage of this design pattern is that it is not reusable, hence it is usually applied for use within a single application.

Salami Slice defines all elements globally but type definitions are kept locally, making it possible for other schemas to reuse the elements. Schemas based on this approach is often coupled to other schemas, hence changes to one schema impact another. The approach is commonly used

since it is easy to understand and create reusable components, and is recommended when trying to establish data standardisation across different applications.

The Venetian Blind pattern requires all elements and components to be globally defined as XML-schema types, and illustrates that components are factored into their most atomic state and can be shared across different XML namespaces. In contrast, the Salami Slice approach defined elements globally and types locally. Since both simple and complex types are defined globally they are available for reuse, that said a schema becomes very verbose because it is not self-contained and may be coupled with other schemas. Venetian Blind is suitable when flexibility, reuse and namespace exposure are important, and is considered an appropriate design when data is transferred between diverse organisations or business unit. The main argument is that it provides each group the flexibility of modifications for each specific requirement.

4.3.4 XML Binding

As mentioned in the previous section an XML-schema adopts principles from OO programming languages. Given this relationship it is possible to map the XML-schema to represent the valid building blocks of an XML document as classes and variables within an OO programming language. This is called *XML Binding*. To perform XML binding a grammar is needed. The grammar is a set of rules defining the structure of a family of XML documents. Defining elements and attributes that can appear in a valid document together with how they are nested, an XML-schema is such a grammar. The process of generating an XML representation for an object in memory is called *marshalling*, while the opposite, building an object from a representation in XML is called *unmarshalling*.

A number of tools for automatically generating classes from a given XML-schema is available both for free and purchase. The tools also provide a convenient way of marshalling and unmarshalling XML files constrained by a schema. The programmer only works with the OO classes and the tool automatically populates (unmarshals) the classes with data from a given XML file, or writes (marshals) a new XML file based on the data the programmer has stored in the classes. The advantage of using generated code is that you can be certain that the data object are properly linked with the XML document, also the code generated gets the appropriate types because of the rich support of data types in XML-schema. The main disadvantage of data binding is that it creates a very tight coupling between the application data structure and the XML document structure. A selection of tools assisting in the XML binding process is given below:

- Java
  - Java Architecture for XML Binding (JAXB)[20]
  - XMLBeans[21]
  - Hydrate[22]
  - Castor[23]

[20] https://jaxb.dev.java.net/
[21] http://xmlbeans.apache.org/
[22] http://hydrate.sourceforge.net/
[23] http://www.castor.org/

- .NET has built-in functionality for XML binding through System.Xml.Serialization
- C++
  - Code Synthesis[24]
  - xmlbeansxx[25]

Sun has defined an Application Programmer Interface (API) making it easier for developers to access XML documents, JAXB. It reference implementation and how to work with XML in .NET is discussed in more detail in the following two subsections.

JAXB

JAXB is an open-source project at java.net and provides a convenient way of binding an XML-schema to a representation in Java-code. The tool includes a binding compiler, *xjc*, that generates the specific Java classes from the schema definition. The classes represent elements and complex types in the schema. xjc is a command line application that requires the name of the XML-schema as argument when using it. Once this compiler has run, the Java classes are ready for use and the programmer can include them into his or her project. When running the compiler it is possible to specify the package in which the generated files should be placed. The necessary classes and functions needed by the programmer to perform the marshalling and unmarshalling of XML documents are provided by JAXB.

.NET XML Serializer

The great advantage of representing data using XML is that there is nothing in a standard XML document that ties it to a programming language. In practise this means that an XML file generated from Java classes using for example JAXB and an XML-schema can be unmarshalled and represented in any other programming language. Microsoft also has a rich library for working with XML documents supported both in the .NET Framework (desktop) and .NET Compact Framework (mobile). As with JAXB a binding compiler called *xsd* is provided, and is much like xjc. It is a command line application requiring the name of the XML-schema to generate the specific classes. The tool has a number of advanced options to make the generated classes fit into the target application. It is worth mentioning that .NET XML Serializer does not follow the code conversion of the language in which the class definitions are generated. For example, a complex type in XML is usually given a name with lower case letters. When xsd generates the class to represent such a type, the class is given the same name as stated in the XML file. Most code conventions require that the first letter in any class is capitalised, hence the convention is not followed.

Using the above mentioned technologies it is possible to exchange information between two different programming languages, and at the same time working with native classes of each of the programming languages. This illustrates the high degree of interoperability achieved when using XML to represent data.

[24] http://codesynthesis.com/products/xsd/
[25] http://xmlbeansxx.touk.pl/

4.4 Summary

In this chapter we have had a look at some technologies relevant to the work we are doing in this thesis. First we had a look at the server side were we presented the personal middleware platform Argos. Important features from Argos are its ability for persistent storage and web service support through Hibernate and Axis respectively. The development for the client side in this thesis targets mobile devices and a presentation introducing different wireless technologies, operating systems and mobile architecture was given. Symbian OS has the highest market share for mobile devices worldwide, while Microsoft Windows Mobile is most popular in the United States of America. We saw that interoperability between communicating applications running on different computers and/or devices can be achieved by expressing messages in XML.

Chapter 5

Design

In this chapter we first present the design of a new system service added to the Argos platform. Next, we present the design the two applications, esmDesk and esmMobile, that together provide an experimenter with the opportunity of creating and running ESM based surveys.

5.1 Limitation

Because of the time constraint for this thesis, the software designed and implemented is not a fully realisation of the requirements of chapter 3. The focus has been on creating two systems that together can demonstrate the concept of ESM. From the requirements in appendix A the requirements Functional Requirement esmDesk (FRD)-112, FRD-130, FRD-134 and FRD-135 have been left out.

5.2 Argos

In section 4.1.1 we gave a presentation of the middleware platform known as Argos. From the problem definition we stated that we want to investigate how applications can benefit from using Argos. The main reason for doing that was because Argos comes with a set of system services that we could take advantage from when building our ESM software. Among the features of Argos, we were particularly interested in the ones that could provide us with a web server (!!jetty), web services (!!webservice) and persistent storage (!!hibernate).

5.2.1 !!SMSservice

In addition to the system services shipped with Argos, we needed to be able to send and receive SMS messages for communicating with devices. Sending messages to devices is currently limited to the SMS technology because a mobile device does not have a static IP address. Telenor[1] provides a gateway for SMS messages between content providers and end users. This gateway is called Telenor Mobile's Content Provider Access (CPA) and offers transmission, administration and billing capabilities for SMS messages. This thesis is a collaboration with Telenor, hence we were given permissions to use the design and implementation of the CPA in our application. Therefore we have designed a system service that wraps around the CPA implementation and provides Argos with support for sending and receiving SMS messages. The motivation behind

[1] http://www.telenor.no

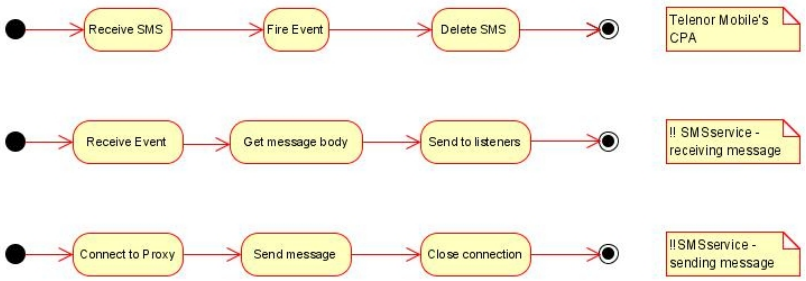


Figure 5.1: !!SMSservice - activity diagram

designing the SMS service as a system service is that it extends the functionality of Argos, hence any component deployed within Argos can utilise the SMS service. In short, the architecture of the CPA is centred around two message queues; one for sending and one for receiving messages. The design is event-based and the message listener does not poll the queue to check if there are any messages, an event is fired when a new message is waiting at the queue. !!SMSservice should employ Telenor Mobile's CPA both when sending and receiving SMS messages. Even though this design is bound to Telenor Mobile's CPA it should be fairly easy to use another implementation for the actual sending and receiving of the messages.

Figure 5.1 shows the activity diagrams for !!SMSservice. The top most activity diagram is for Telenor Mobile's CPA. Here an SMS is received in the dedicated message queue and the CPA system fires an event to all listeners, in our case and event to the !!SMSservice. Once all listeners have been notified about the new message, the system will delete the SMS from the queue. The second diagram is for the receipt of messages in the system service. The service shall receive an event sent by Telenor Mobile's CPA containing the message. Upon receipt of such an event !!SMSservice shall forward the message to all listeners within Argos. The last diagram is for sending messages, which would typically be done in three steps. First, establish a connection to the message proxy used for sending the message, then send the actual message and finally close the connection to the proxy.

5.3 esmDesk

esmDesk is the first of the two software systems from section 1.3. The main goal for this application is to provide the experimenter with a tool where he or she can create, distribute and collect results from an ESM based survey. As mentioned earlier, the non-functional requirement NFR-107 specifies that esmDesk shall be built on top of the middleware platform Argos. In this section we will first present the architecture of the application seen from Argos, and later we will bring forward a more detailed description of the design of the application.

5.3.1 General Architecture

In section 4.1.1 a presentation of the different parts of Argos were given, and the design of !!SMSservice is already presented in this chapter, hence the reader is encouraged to read these sections for further details of those parts of the application. esmDesk has been designed as a *user service* when referring to the terminology used in Argos. The architecture of the application is

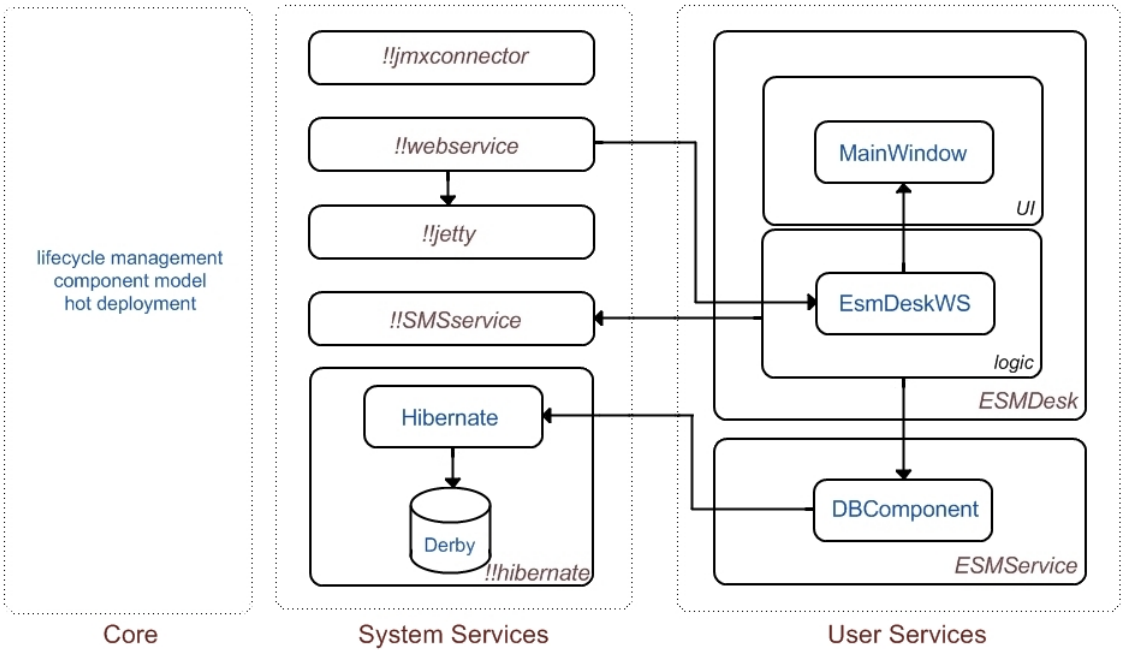


Figure 5.2: esmDesk as a service in Argos

shown in figure 5.2, which conforms to the general architecture of Argos from figure 4.2. Only the parts of esmDesk exposed as components in Argos are depicted in figure 5.2. In addition to esmDesk, there is another service called *ESMService*. The service should contain one component which interacts with !!hibernate in Argos. The component inside ESMService would provide esmDesk with all functionality needed for storing and retrieving persistent data.

Together esmDesk and esmService constitute the architecture of the entire application, making a 3-tier architecture. At the top layer is the GUI (UI in the figure), which contains all parts of the system visible to the experimenter. The middle layer (logic), is the core layer of the application, it is the linking pin between the two other layers. Objects within this layer live just in memory for the duration of a particular operation or request from either of the two other layers. The bottom layer (esmService) takes care of persistence of objects from the middle layer. Here the preferences for a particular survey set by an experimenter should be stored, hence making it possible to retrieve them later. The 3-tier approach was chosen based on the divide and conquer strategy. It is better to have three simple layers, than one complex whole. In large scale projects this approach is also preferable because each layer can be assigned a specialist to improve the overall quality of the application (e.g. a GUI designer for the UI layer). Splitting an application into these three layers also makes it easier to maintain the different parts because it is a well defined task assigned to each part of the architecture.

esmDesk is mainly a GUI application and has one main window where it displays the user interface. This main window, *MainWindow*, should be exposed as a component within the service to be able to start the application. A different approach would be to place the GUI outside the middleware platform, but because of the tight coupling to services within Argos it was decided

to include it as a component within the middleware platform. A second argument for incorporating the user interface in Argos is that the application requires a two way communication between the different layers of the architecture, and placing the user interface outside Argos may cause an unwanted delay when requests are made to the object or persistence layer in the architecture.

A common disadvantage of the ESM software presented in chapter 2 was that devices had to be physically connected to a computer in order to install the survey created by the experimenter. In a survey with many participants such an approach can be very time consuming. Our focus has been on designing a system where the distribution of surveys can be done without physically connecting the device. The component *EsmDeskWS* should handle all communication with esmMobile initiated by a mobile device. Situations where the mobile device will initiate the communication are typically when it wants to retrieve a new survey or submit results from a completed survey.

5.3.2 Persistent Storage

It is likely that an experimenter will not finish creating a survey in one go. Therefore the system must have some way of storing information about the created survey. This way the experimenter can return to where he or she left of during the construction of a survey. An obvious choice for storing information is using a database. Using databases has many advantages over using plain text files for storing information. The most important features are reduced data entry, storage and retrieval costs, and the fact that it improves data access to users through the use of host and query languages [29].

The information needed in the database is basically everything that is required by a mobile device to run a survey. This information includes all the elements constituting the entire survey, their properties, when to start the survey and what element should be the first to run. In addition to properties related to actual running the survey, esmDesk shall maintain details on the author of the survey, at when it was created, modified and if it has been distributed or not.

In different surveys it is expected to use some common elements, which are questions or measurement methods. Hence, information about an element should also be stored in the database. An element is not connected to a particular survey, thus it can be used in all surveys. It is important to make a separation between the two sets of elements. A question is just a string of characters meant to be asked directly to the participant, whereas a measurement method requires detailed information on how to instruct the mobile device to use the correct sensor etc. Running a survey depends on a set of participants, and according to requirement FRD-138 esmDesk shall provide persistent storage for a number of details about each participant in a survey. Storing such details can be helpful for further analysis of the results in a survey. Some participants may take part in more than one survey, hence when storing them in a repository, the experimenter may not have to specify all participant details for every survey.

When results are returned by mobile devices, they must be stored and organised in the database. The database has been designed in such a way that it is possible to determine what a particular participant has answered in all surveys that he or she has participated in. There are certain

rules that must be followed as it is possible to determine what a participant answered in a survey, such rules are not applied in this development due to timing constraints.

With reference to the architecture in figure 5.2, *ESMService* shall provide esmDesk with all the functionality for working with the persistent storage. Together with the Entity Relationship (ER) diagram for the database, the class diagram for this service can be found in appendix D.

5.3.3 Communication Paradigm

In section 2.7 of the Related Work chapter we presented a set of ways to establish communication between a mobile device and a computer. We have designed our system with the *Push-pull-scheme* as a starting point basically because of the lack of a static IP address for mobile devices. When the experimenter has finished creating a new survey and is ready to distribute it out to the participants, he or she should start out by selecting the participants. A participant could be selected from the repository of participants, or a new participant can be added to the repository and successively attached to the survey. To be able to distribute the survey the system must have the unique phone number identifying the participant. It should not be possible to add a participant to the repository without entering a valid phone number for him or her. Once a set of participants have been selected, the system can start the process of notifying devices that a new survey is ready. Using !!SMSservice the system shall create an SMS message and send it out to all participants. The content of this message must be kept at a minimum because of the 160 character constraint for SMS messages. Information contained in the message shall be structured using a mark-up language such that it can easily be parsed by a receiving end. Provided the information in the message a mobile device shall contact esmDesk to retrieve the representation of the survey. This way esmDesk *pushes* a small message to esmMobile, and esmMobile *pulls* out the information available from esmDesk. An advantage of using SMS message is that it is not required that the mobile device is in range of a network or actually turned on. The message remains at the network provider up to seven days[2].

Depending on the preferences set by the experimenter, esmMobile shall either send results after each element in a survey or when an entire survey has completed. Allowing the system to receive answers immediately after the participant has answered, opens the possibility for communication between an experimenter and a participant. We saw in the chapter 2 that PHT Product Suite offered a system where a participant could get feedback from an experimenter. Certainly, that system was a personal diary more than an approach towards ESM, but in relation to the work done at NST and Telenor it is important to incorporate this feature in esmDesk. The approach for sending back the results is almost the same as for distributing the survey. The mobile application shall create a representation of the results in a mark-up language and using the information from the first SMS, esmMobile can contact esmDesk to submit the results.

5.3.4 User Interface

[30] summarises the principles for designing user interfaces. Among this set of rules there are two of particular interest, namely the one that is related to knowing your audience and the one that has to do with coherence. The user group of our system is identified as psychologist, doctors and people within the domain of ESM. An important observation is that these people are not

[2] varies between network providers

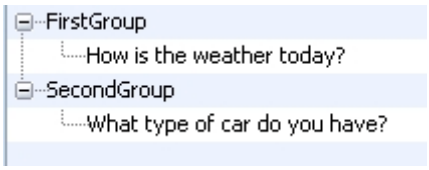


Figure 5.3: File tree metaphor

software developers and it was essential for us to make the user interface as simple as possible, but still powerful enough to meet all the requirements from the stakeholders. Throughout the entire development process we have had a tight dialog and exchanged ideas with people from both Telenor, NST and department of psychology at University of Tromsø to get inputs on how they regularly design surveys when done on normal paper. The goal was to visualise their ways of working in the ESM software, making the transition from paper based surveys to ESM surveys as uncomplicated as possible. Creating an ESM based survey involves the specification of several parameters that the participant does not see or care about (e.g. the start time of a survey). As a consequence the representation of a survey seen from the experimenter perspective is different from the one seen from the participant, making it a big challenge to display the entire survey to the experimenter in an easy way.

The concept of using metaphors in user interface design is widely used. This means that developers borrow a well known abstraction from another user interface and incorporate it into their application to cut back on the time required for new users to learn the system. We chose to use a file explorer metaphor. Most people familiar with a computer have been using a file explorer where folders are organised in a tree structure. We decided to split a survey into groups that could contain one or more elements. Such a group shall have a set of preferences allowing it to run its elements at specific times, and it shall also be possible to specify that the elements within a group can be repeated. This way the interval-contingent protocol of ESM is covered. Displaying these groups as top level “folders” with their elements as “files” was considered to be a good metaphor. This way the experimenter can move elements between groups the same way he or she is used to move files between folders in a file explorer view of the operating system. Figure 5.3 is an attempt to illustrate the file metaphor in action, here two groups are defined, each containing one element.

The support for globalisation and internationalisation has become an important factor in software development. All the language specific parts of esmDesk shall be separated from the rest of the application making it relatively straightforward to define a different language for the application.

Designing user interfaces often introduces a large number of classes. Therefore it is important to make the design as extensible and easy to follow as possible. An application may contain different parts that are visually represented in the same way. From the requirements there are three repositories defined; one for questions, one for measurements and the third for the available input methods. All these repositories shall be represented as tables in the user interface. The only thing that could differ is the actual content inside each table. Figure 5.4 proposes a design for the different tables and the actual presentation of the table in the GUI. The interface

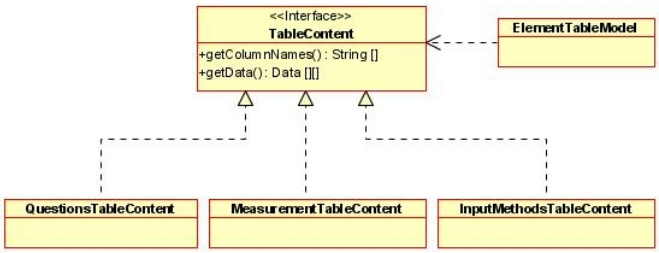


Figure 5.4: Contents of a table

TableContent specifies the methods required by any class that wants to display some content in a table, and *ElementTableModel* works with instances of this interface. This design makes it very easy to extend the user interface with other tables and also change the data displayed in the existing tables.

A complex user interface often contains a lot of actions and events fired as a result of user interactions. In figure B.1 of appendix B there is a large package called *Action*. This package shall contain all the actions defined for the user interface, hence an action container. Within the container the different actions should be divided into sub-packages based on which parts of the user interface they can occur in.

Most programming languages that supports user interfaces creation, handle all actions and events in a separate thread of the application called the event-dispatched thread. A rule of thumb is to make sure to limit the execution time of this thread to a small number of milliseconds each time it is given a task. The implication of this is that if an action followed by a task is expected to take up several milliseconds, seconds or even minutes it should be dealt with in a separate thread. The reason for doing that is to prevent the user interface from hanging or "freezing". If the interface freezes the user might think that something is wrong, and may take unwanted actions. In esmDesk a separate package in figure B.1 is named *worker* and shall contain all the workers performing tasks required by the user. These tasks may include sending messages, reading and writing files and more. Once the task is finished, the worker should intercept the event-dispatched thread signalling that the result is ready.

5.3.5 Activities

Having presented the general architecture and underlying principles of the application for creating ESM based survey, this section will cover the activities of the application. Figure 5.5 gives an overview of the possible options an experimenter has working with esmDesk. The following subsections will discuss these options in greater detail.

Building a Survey

The most important activity of esmDesk is the one giving the experimenter the ability to create a survey. A survey is compound of groups and elements, and it is the job of the experimenter to create and correctly arrange these different parts. Adding a group to the survey shall involve setting several preferences for when to start the first element of the group, how often the group

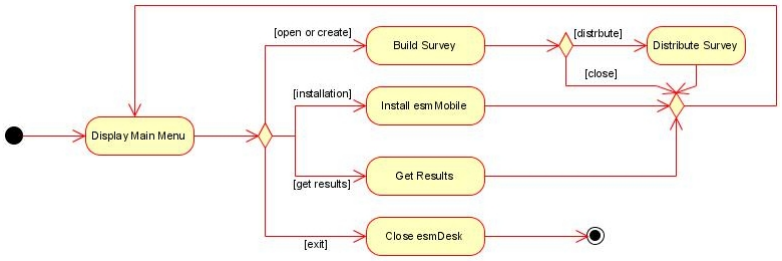


Figure 5.5: esmDesk - Activities

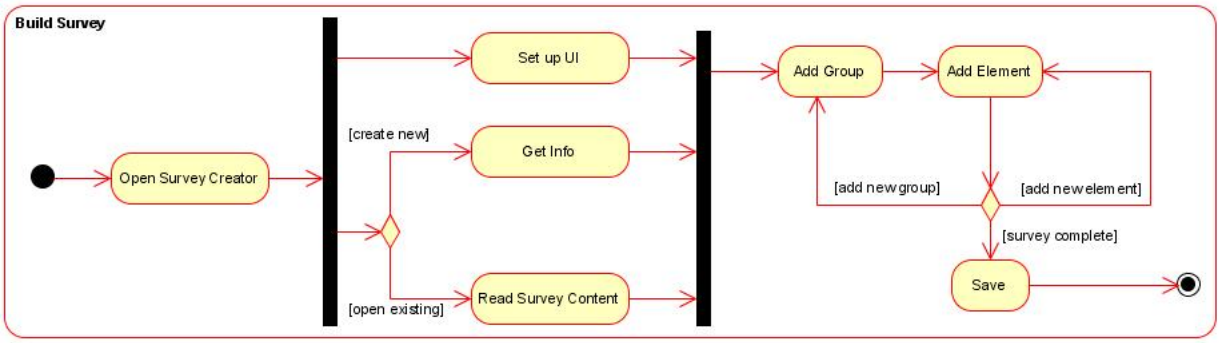


Figure 5.6: Building a survey

is to be executed and the delay between each element in the group. Adding an element is slightly more complex and shall require the experimenter to specify a number of particular settings for the element. These settings are related to how the participant should answer a question and how much time he or she is given to answer the question. More details on this are covered in the implementation chapter of this document. Figure 5.6 shows the activity diagram for building a survey.

As the survey is completed the experimenter can choose to distribute it out to a set of participants. Before distributing the survey, a set of participants must be selected. Each participant shall be in possession of a device running esmMobile or another engine being able to interpret the content of a survey created using esmDesk. At this point esmDesk shall notify all selected participants about the created survey, and each device shall obtain it from esmDesk. Consequently, the distribution is done in two steps (see section 5.3.3).

Installing esmMobile

To be able to run surveys created using esmDesk it is required that a set of mobile devices are running an engine capable of interpreting the contents of the survey. Because the mobile application is required to do some processing by it self, the smart client architecture for mobile devices should be followed, consequently software must be installed before it is able to run a survey. Using the *Install esmMobile* option of esmDesk it shall be possible to install the required software on a set of devices. It is assumed that the experimenter has the set of devices at hand

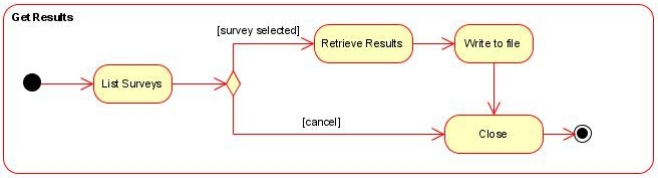


Figure 5.7: Retrieving results

when trying to install the software. The reason for this is that installing applications on devices often require confirmation from the user, and one would typically arrange such that all the necessary software is installed before handing out the devices to the participants of a survey.

Results

Any mobile device running a survey shall submit the answers from its participant according to the specifications of the experimenter. As these results are received in esmDesk they should be persistently stored, and when results are back from all participants in a survey the system shall notify the experimenter. This notification tells the experimenter that results from a survey is back and by selecting *Get Results* from the main menu, results can be retrieved from the persistent storage and given to the user. Chapter 3 outlines the focus in the thesis and states that analysis of results is not a part of this project, hence results shall be written to a file following a pre-defined format.

Closing esmDesk

The last option for esmDesk is one for shutting down the entire application. Before closing, it is important that all preferences set by the experimenter is persistently stored, consequently no information will be lost.

5.4 esmMobile

The second part of the ESM software tool is the part actually running the survey on a mobile device, *esmMobile*. The main goal for this application is to run a survey according to the specifications set by an experimenter using esmDesk. When the survey is finished, the application should send the results back to esmDesk. First a general description of the architecture is given here, before going in more depth of the design of esmMobile.

5.4.1 General Architecture

The architecture of esmMobile is shown in figure 5.8, and follows the smart client approach. Even though the application is less complex than esmDesk the principle of separating the application logic from the user interface has been followed here too, and because of its lesser complexity the persistence layer is included in the logic, making it a two-tier architecture. The application logic can be split into four main parts as the figure shows. Basically these four parts should be handlers for different resources in the application logic. The database handler should interact with a mobile database providing the application with persistent storage. The

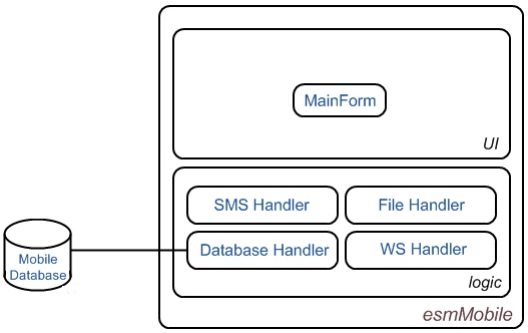


Figure 5.8: esmMobile - architecture

WS Handler would be responsible for the communication esmMobile initiates towards esmDesk. Upon receipt of SMS messages these should be handled by the SMS handler, and finally working with files is to be done by the File Handler. The user interface of the mobile application shall be fairly simple because the only task it has is to display a simple menu and of course the user interface required when running an element from the created survey. Like MainWindow in esmDesk, esmMobile also has one main instance that displays the current user interface, namely *MainForm*.

Figure 5.9 shows the activity diagram for esmMobile. Due to technology limitations SMS is currently the only way to contact a mobile device, thus the SMS Handler is displayed in this diagram. Using the push-pull approach, esmMobile shall retrieve the new survey when an SMS is received and store the survey persistently in a local database. As mentioned each group in the survey should have a start time attached to it, when the first of these groups reaches it start time esmMobile shall start to run the survey. Each element is then run and answers are collected. As long as there are elements left in the survey, esmMobile will continue to run them. After the completion of all elements, esmMobile should organise all the results and submit them back to esmDesk given that the experimenter has specified that results shall be submitted at the end of a survey. If not, each answer shall be submitted before picking the next element for execution. In some situations experimenters might find it useful to run multiple surveys at the same time, this can easily be solved by implementing a multi-threaded version of esmMobile.

For esmDesk an action container was introduced for handling all events occurring in the application. The container approach has been followed for esmMobile as well, but here two different containers are needed. Events that can occur in the application are separated into two groups. The first group shall contain events relevant to the entire esmMobile application (e.g. event fired when an SMS is received), and the second group ought to handle events only relevant to a particular survey (e.g. a particular question is answered), named EsmMobileEventContainer and SurveyEventContainer respectively. For EsmMobileEventContainer the singleton pattern shall be applied, while an instance of the SurveyEventContainer should be created along with every new survey.

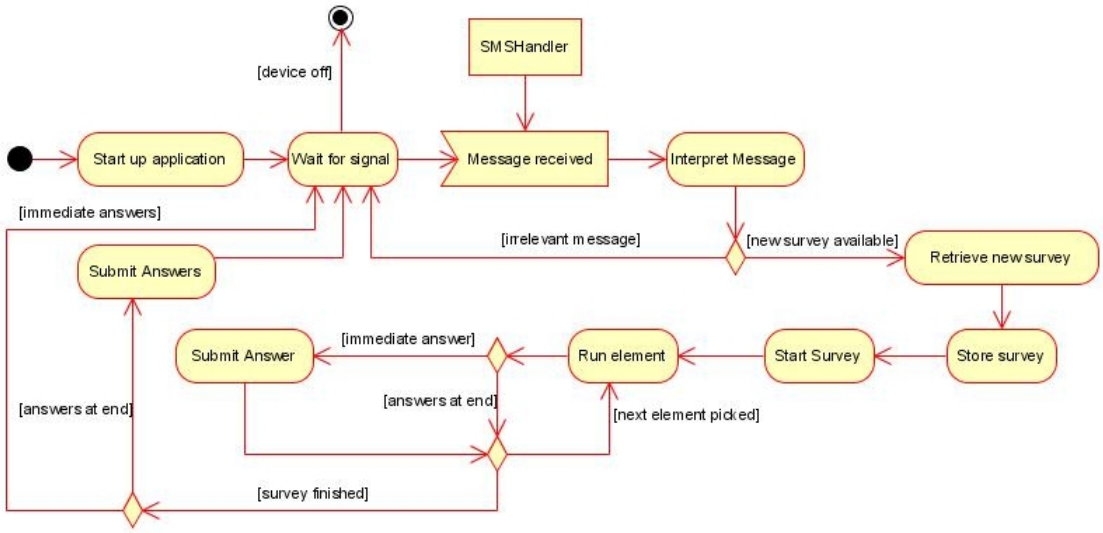


Figure 5.9: esmMobile - running a survey

5.4.2 Persistent Storage

A traditional survey is normally completed within a number of minutes, maybe hours and participants write their answers on paper, and hence answers are persistently stored unless the paper is lost. The ESM approach introduces some new aspects when it comes to storing the answers for later use. Since it deals with electronic devices one possibility could be to keep all the answers in memory until the survey is completed, and then returning them to the experimenter. Unfortunately, this approach would not work for very long. Mobile devices run out of power and many people turn their devices off when they go to bed in the evening, being on board aeroplanes etc. In a situation where all answers were kept in memory, the system would loose them as soon as the device is turned off. With the rapid development in mobile technology it is now possible to store information in a local database on the mobile device. esmMobile shall utilise this feature and store the entire survey and the answers from a participant in a local database. Immediately after esmMobile has retrieved a new survey, it should interpret and store all details about the new survey in a local database. Such an approach could minimise the risk of loosing the survey due to reasons mentioned earlier. The Entity Relationship (ER) diagram for the database used by esmMobile can be found in appendix E.

5.4.3 Communication Paradigm

Assuming that the mobile device is within range of a network, the SMS message containing information about the new survey will be received and interpreted by the mobile device. SMS messages are a very popular way of communicating between owners of mobile devices. Therefore it is important that the SMS message sent by esmDesk follows a particular format such that esmMobile can distinguish it from other messages meant for the user of the device. The SMS Handler must be able to examine the content of the SMS before giving it to esmMobile. If the message conforms to the predefined format, it must be removed from the message queue in the device to prevent the user from seeing it. On the other hand, if it turns out to be a regular SMS, the SMS Handler shall ignore the message and leave it in the message queue for other

applications to retrieve it (e.g. the inbox program).

Having parsed the SMS and determined that a new survey is available, the mobile device shall pull the new survey from esmDesk using information provided by the SMS message. In chapter 4 a presentation of the different wireless technologies for mobile devices where given. Determined by the type of device and the location of it, it is important to know which of the technologies are available and ready to use. There are advantages and disadvantages of all types, for example using a WLAN connection provides relatively high bandwidth and is considered stable. That said, it requires a lot of power which is a big constraint for mobile devices. Pulling data from esmDesk must be done using one of the supported wireless technologies for a particular device. The WS Handler of esmMobile shall maintain a collection of all available wireless adaptors for the device. In advance it should be defined what type of wireless adaptor is required for all possible requests to the WS Handler. Whether to execute a request or delay it, should be determined by the type of request given and the currently available wireless adaptor. A request shall be delayed until the required wireless adaptor is available.

5.4.4 User Interface

Designing a user interface for a mobile device has many of the similar challenges as desktop user interface development, but also new challenges are introduced. A mobile device can be split into three categories; mobile phone, smart phone and PDA. The software developed here aims at PDA devices, consequently they have a touch-enabled screen and a pointing device called a stylus. A user can choose to use either the finger or the stylus operating the device. The problem using the finger is that buttons are usually very small, hence using the finger could result in tapping more than one button or even the wrong button. Therefore, it is preferable to use the stylus because of its similarities with a pencil, thus very precise. A goal within ESM is to put as little burden on the participant as possible. Designing a user interface that constantly would require the participant to use the stylus does not fulfil this requirement. All user interfaces in this application shall be designed with the goal that a participant should be able to use his or her finger most of the time operating esmMobile.

Input Method Plug-in Framework

Requirement FRD-117 presents a set of different input methods. An input method is the way an experimenter allows participants to answer a given question, accordingly a method is directly associated with the user interface on a mobile device. An evaluation was made to see if it would be possible to express this interface entirely in the description containing the survey sent from esmDesk. It was considered to be to complex due to the time constraints in this thesis. The proposed solution is to have a one-word description of the user interface in the mark-up language describing the entire survey, and leaving it to the developer of the mobile application to support the different input methods. As an attempt towards making the device application as extensible as possible a plug-in framework for the different input methods has been designed. This framework allows for a developer to add new input methods without actually changing the source code for the mobile application. An input method is a stand-alone library which conforms to an interface defined by esmMobile. esmMobile shall organise all the input methods it is aware of in a collection, when reading the survey representation it will try to determine if it has the input method which is set in the description. If not available, esmMobile can make a request to esmDesk where it reports that the requested input method is not available.

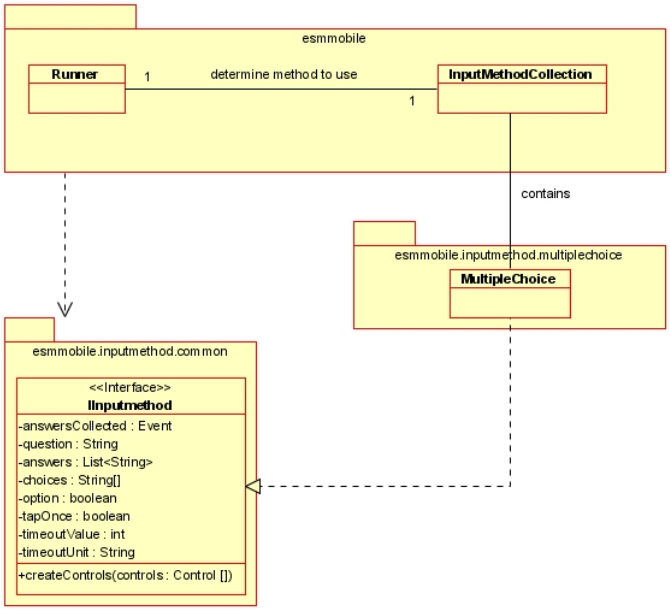


Figure 5.10: esmMobile - class diagram

esmDesk could then have the opportunity to return the library containing the input method, and consequently esmMobile can plug this into its collection and use it.

Figure 5.10 displays the proposed class diagram for the plug-in framework as it has been designed for esmMobile. A package named *esmmobile.inputmethod.common* contains the interface which is realised by an input method (multiple choice in the figure) and used by esmMobile. The class named *Runner* in the figure represents the instance that could run an entire survey.

5.5 Summary

In this chapter the design of the two software systems, esmDesk and esmMobile, has been presented. esmDesk is built as a user service within the middleware platform Argos, and allows an experimenter to create a survey based on the ESM approach. The survey will be stored and distributed according to a well formed mark-up language description. The communication between esmDesk and esmMobile has been designed after the push-pull approach from section 2.7. When a survey is finished, esmMobile creates a representation in a mark-up language containing the results and sends it back to esmDesk. The use of a mark-up language when exchanging message between the two systems makes a very loose coupling between esmDesk and esmMobile when it comes to the platform on which the two systems run.

The design also shows that the ESM process comes down to three steps following each other, and never happen at the same time for a particular survey. These three steps are the (1) the creation of the survey using esmDesk, (2) the execution of the survey using esmMobile and the final step (3) where esmMobile sends the results back to esmDesk.

Chapter 6

Implementation

In this chapter the implementation of the design from previous chapter is outlined. First the implementation of the new system service added to Argos, then the implementation of esmDesk and finally the platform running the entire survey, esmMobile. Screen dumps from the different applications can be found in the user guide in appendix G.

6.1 Technology

From the brief presentation of existing technologies in chapter 4 we have selected a subset to use in our development of the ESM prototype software. The esmDesk GUI application is implemented as a user service in Argos. A survey created using esmDesk is represented in a format based on a well formed XML-schema, consequently achieving a loose coupling between esmDesk and esmMobile. Having a loose coupling like this has many advantages when working in mobile environments. The greatest is probably the fact that the only requirement put on the mobile device running a survey is that it must have the ability to parse and interpret the content of a survey created using esmDesk.

For the mobile development Microsoft Windows Mobile platform has been chosen. Microsoft Windows Mobile was chosen because it is an excellent way to illustrate the loose coupling of the two systems by developing them on different platforms. The choice of platform was also based on the fact that the work to be done at NST also targets devices running the Microsoft Windows platform. Their argument for choosing this platform instead of for example developing software using Java 2 Micro Edition (J2ME) was the lack of Bluetooth support in J2ME.

From the design in previous chapter the communication between the two applications follows a push-pull scheme. The technologies involved in this scheme are SMS and web services.

6.2 Dependencies

Argos is a dependency already identified in requirement NFR-107, and esmDesk has been implemented with the goal of deploying it as a user service in Argos. It is also assumed that none of the system services shipped with Argos have been removed when deploying esmDesk. Establishing communication with mobile devices requires the ability to send and possibly receive

SMS messages to and from participants. To provide such a feature, the implementation of the new system service, !!SMSservice, is a wrapper around Telenor Mobile's CPA [31].

The mobile application developed is platform dependent and assumes a device running Microsoft Windows Mobile 5.0 Pocket PC Edition operating system. It is recommended to use Microsoft Visual Studio 2005 for further development of esmMobile. In addition to the built-in features for mobile development in Visual Studio, esmMobile depends on Microsoft Mobile Client Software Factory (MCSF)[1]. This software factory provides developers with guidance and help in developing mobile applications. Several of the application blocks are used by esmMobile. The most important one is the Disconnected Service Agent, which provides a powerful way of executing web service requests in a disconnected environment. The application block ensures that a request is processed when the device is connected to a network. If the device is not connected, all requests are delayed until a connection has been established. The mobile application needs to store information persistently and depends on Microsoft SQL Server 2005 mobile edition.

6.3 Programming Languages

The two systems are developed on different platforms; hence distinct programming languages are used.

6.3.1 esmDesk

A strong dependency for esmDesk is Argos, which is written entirely in Java[2]. For a service to be deployed in Argos it is required that it is written in Java, consequently the programming language for esmDesk is Java. Java is an object-oriented, robust and widely used programming language. It has an extensive library of parts which may be used for building programs, thus saving time in the development stage. A great strength of Java is that it is platform independent; it can run on any machine that has a Java virtual machine installed. In the early days of Java the most considered disadvantage was its performance compared with other languages such as C and C++, but with the latest release of Java it is said that Java should be very competitive with for example C++[3]. The new system service for sending and receiving SMS messages is also written using Java because of the integration with Argos. esmDesk is implemented using the editor Eclipse[4].

6.3.2 esmMobile

Microsoft Windows Mobile 5.0 Pocket PC supports three programming languages being Visual Basic, Visual C++ and Visual C#. Among these three, Visual C# has been chosen because of its similarities to Java and the fact that MCSF is written in Visual C#. Like Java, Visual C# is object oriented and provides a rich library for the programmer to exploit. Applications developed using Visual C# can only run on an operating system from Microsoft, hence Java has an advantage over Visual C# being platform independent. Together with Microsoft Visual Studio 2005, Visual C# provides a very powerful and easy way to get started developing

[1] http://msdn2.microsoft.com/en-us/downloads/default.aspx
[2] http://java.sun.com
[3] http://www.idiom.com/ zilla/Computer/javaCbenchmark.html
[4] http://www.eclipse.org/

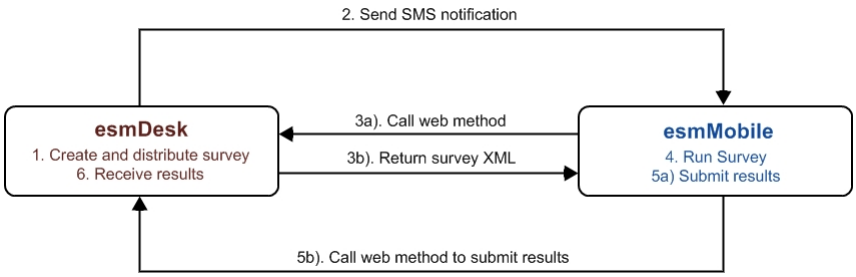


Figure 6.1: Overview

mobile applications. esmMobile is implemented using the editor Microsoft Visual Studio 2005 Professional[5].

6.4 Overview

Figure 6.1 illustrates the relationship between the two systems designed and implemented in this thesis. With esmDesk the experimenter creates a new survey and selects a set of participants to distribute the survey to. The actual distribution is done in two steps. First esmDesk generates an SMS message and sends it out to all participating device. A device that receives such a message reads out the content and calls a pre-defined web service method to obtain the XML representation of the survey. Once the survey has been retrieved, esmMobile interprets and runs it before calling another web service method for sending the results back to esmDesk.

6.5 !!SMSservice

A system service in Argos provides new functionality to services and components deployed within the middleware platform. Facilities for sending and receiving SMS messages are made available through a new system service. The service is not bound to be used together with esmDesk, hence any component that needs the ability to send and/or receive SMS messages can take advantage of this system service. Figure 6.2 depicts the class diagram for the service. A component in a user service of Argos obtains a reference to the SMSservice instance using *dependency injection*, meaning that Argos passes the reference to the SMSservice instance directly to the component asking for it. The code sample below shows how to obtain a reference to the SMSservice instance in Argos.

```
@Component("SMSservice") public SMSservice smsService;
```

Listing 6.1: Dependency injection and SMSservice (Java)

[5]http://msdn2.microsoft.com/en-us/vstudio/default.aspx

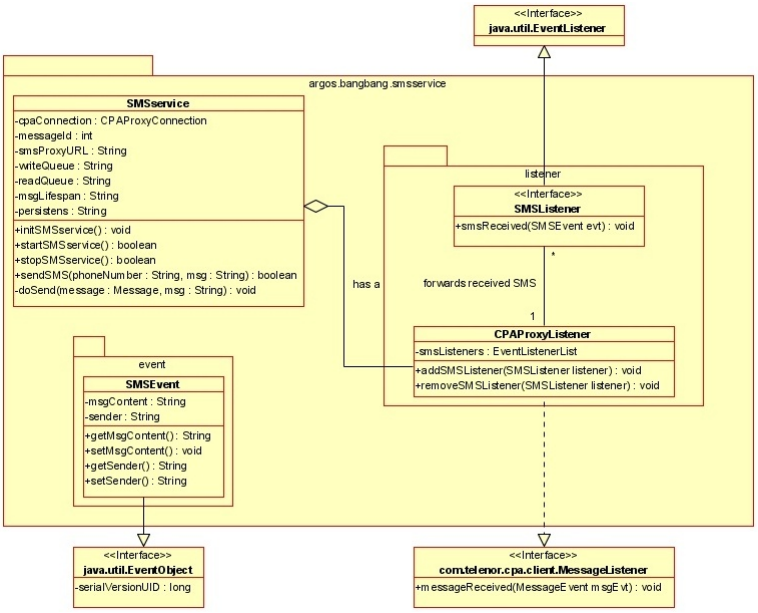


Figure 6.2: !!SMSservice - class diagram

6.5.1 Receiving SMS

Any component can register to listen for incoming SMS messages by implementing the interface *SMSListener* and invoking the function *addSMSListener* of *CPAProxyListener*. Once the proxy listener is notified about a new message from the CPA proxy, CPAProxyListener forwards the content of the SMS message to all registered listeners. See sequence diagram to the left in figure 6.3.

6.5.2 Sending SMS

To send an SMS message the component needs to obtain a reference to the SMSservice using dependency injection as shown in listing 6.1. Before sending a message, the component must establish a connection with the *CPAProxy* using the method *startSMSservice*. After a successful connection sending an SMS is done using the method *sendSMS* provided the number to send the SMS to and the actual message itself. The connection to the *CPAProxy* is closed by using *stopSMSservice*. It is recommended that a component starts and stops the SMSservice for every time it wants to send a message. The reason is that it is not desirable to have a Transmission Control Protocol (TCP) connection open at all times. See sequence diagram to the right in figure 6.3.

6.6 esmDesk

esmDesk is implemented as a user service in Argos and has a GUI that allows an experimenter to construct and distribute an ESM based survey to a set of mobile devices running esmMobile. Results from surveys are automatically sent back to esmDesk and an experimenter can choose to dump these to a text file. The user interface has been implemented using both the standard

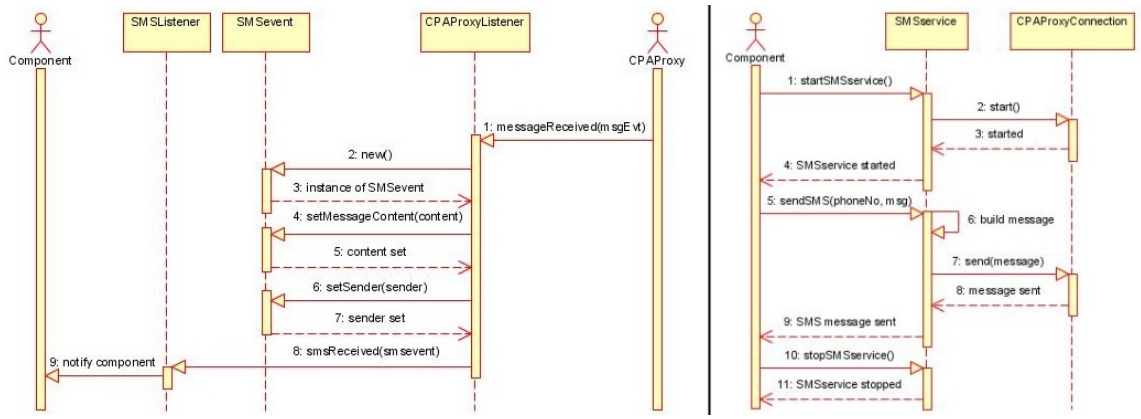


Figure 6.3: !!SMSservice - sequence diagrams

Swing package of Java and the extension called *swingx*[6]. In addition, all tables in the user interface apply the look and feel implemented by Jon Lipsky[7]. This look and feel is based on the colours and layout of Apple's iTunes[8] software, hence providing a nicer look than the standard look and feel from Java. As mentioned in the design chapter, esmDesk uses *esmService* to store data persistently in a database. esmService implements the functionality for inserting, updating and deleting information in the database defined for esmDesk. The service integrates with Hibernate and Derby which is built into !!hibernate of Argos. In addition to providing the database functionality, the service also defines all the mapping classes needed by Hibernate to get the Object/Relational mapping correct. Each table in the database corresponds to a Java class file and an XML file with the *.hbm.xml extension. The XML file is the actual mapping between the database and the programming language. Here, all fields of the table are specified with a name and a type. Information stored in the database includes surveys, participants and results. The class diagram and ER diagram for esmService can be found in appendix D. To establish communication with mobile devices esmDesk utilises !!SMSservice and defines a web service, EsmDeskWS, containing a set of web methods used by esmMobile to retrieve a new survey, update participant information, send back result and more.

Deploying a GUI application in Argos introduces a small problem concerning the closing of the application. At the time of writing there is no user interface for starting and stopping components within Argos. Since esmDesk is an application and a user service within Argos, it is important to have the ability to both open and close the application, in addition to start and stop the entire middleware platform. To cope with this problem esmDesk installs an icon in the system tray of the computer running Argos. Given that Argos is running it is possible to start the application using the tray icon. Of course, stopping Argos also causes the icon to be removed from the system tray.

[6]https://swingx.dev.java.net/
[7]http://blog.elevenworks.com/?p=18
[8]http://www.apple.com/itunes/overview/

6.6.1 Survey Representation

XML is powerful for reasons mentioned in section 2.6.1, and by using a schema to define the building blocks for the XML, we open the possibility for easy extension and changes to this prototype implementation. The information stored in the XML file is only the information that is needed for the mobile device to actually run the survey. This information includes when to start each question, all the questions and specifications on how participants should answer the question. Most of that information is just text which is well suited for representation in a declarative language such as XML. The byte representation of the survey XML is stored in the survey database, together with the name of the author and the date the survey was created. Instead of storing the XML in a separate file, the choice was to store the entire representation in the database. There are two reasons for this choice, first the application can query the database for the byte representation of a given survey and forward it directly to the device asking for the survey. Secondly, as of writing, a component in Argos is not given a *home* folder. Storing a survey in a file would require esmDesk to create a folder within the middleware platform. For further releases of the middleware platform such privileges might be restricted for components. In the requirements chapter it is outlined that the experimenter of a survey shall be able to set different methods for how the participant can answer to a given question. As a consequence, the user interface on the mobile device needs to be dynamically loaded for each question. After evaluating the complexity of expressing user interfaces in XML we decided on not to go for that approach. Our approach is to indicate in the XML what type of input method that should be used for a particular question, and is described in greater detail in the section which outlines the implementation of esmMobile.

In addition to representing a survey in XML, we have also chosen to send the results given by a participant in XML. Visual representations of the structure of both XML-schemas are included in appendix C. The mapping between the XML and Java objects are done using JAXB as presented in chapter 4.

In section 4.3.3 design patterns for XML files were presented. During the creation of the XML-schemas for both the entire survey and the representation of the result, we considered the three patterns. The two schemas of appendix C use a combination of *Salami Slice* and *Venetian Blind* design patterns. The implementations done in this thesis are prototype, hence it is important to make the schemas both flexible and extensible for future change. Combining the two XML-schema design patterns assists us in achieving that goal.

6.6.2 Building A Survey

From the menu of esmDesk an experimenter can choose to create a new or open an existing survey. Selecting either of these opportunities causes the application to display the Survey Creator to the experimenter (see figure 6.4). If creating a new survey is selected, the application asks the experimenter for his/hers name and the title of the survey before displaying the Survey Creator. For the open survey selection, the application shows a list of previously created surveys. Based on a selection made by the experimenter esmDesk reads out the XML representation of the survey from the database using esmService, parses it, and displays the survey creator with the details of the selected survey.
A survey is compound of a set of elements, being questions and/or measurement methods. To allow for the repetition of a set of elements, they are organised in groups (see figure 5.3). In the

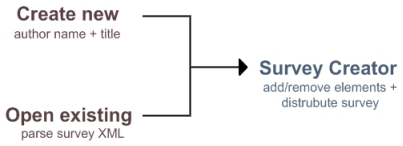


Figure 6.4: Building a survey

Survey Creator the experimenter can add/remove groups and elements. An element cannot exist outside a group. When adding a new group the experimenter must specify a set of properties for the new group. These properties can be changed any time during the creation of the survey and are:

- *Name*
  A name describing the new group.
- *Start time*
  When the first element of the group should start, specified in both date and time.
- *Repetitions*
  The number of times the execution of this group shall be repeated.
- *Repetition delay*
  The delay between each repetition of the group.
- *Element delay*
  The delay between each element inside the group.
- *Element order*
  Sequential or randomised execution of elements in a group.

Once a group has been created, an element can be added to the group. All elements are organised in two different repositories residing in the database, one for questions and the second for measurement methods. In the user interface, repositories are viewed as tables where the experimenter can maintain the repositories by adding or removing new entries in the tables. Using drag-and-drop features of Java Swing elements from the respective repositories can be added to a previously created group. Adding a question to a group requires the experimenter to at least specify what type of input method that should be used. The implemented software supports four kinds of input methods:

- *Multiple choice*
  A set of options are specified to allow the participant to select among. It is possible to specify whether only one or several of the choices can be selected.
- *Yes/No*
  The participant is allowed to answer only yes or no to the question asked. An option is also available to indicate whether the user should be able to answer *no opinion* or not.
- *Free text*
  Using keys on the mobile device the participant can express his answer in free text.

- *Slider*
  Much like a scroll bar with a value/expression on each edge, the participant indicates a value between the extremities.

In addition to specifying the input method, several other options are available. It is possible to specify how many seconds or minutes the participant should be given to answer a question, instruct the mobile device to play a signal or vibrate to get the attention of the participant before showing the question. This shows that it is possible to combine signal- and interval-contingent protocols of ESM. The last option is specifying dependencies. The first part of a dependency is the name of the element from the current survey another element should depend on, secondly an experimenter has to specify the answer required by a participant for the dependency to be fulfilled. For example, in a survey a question A, *Do you have a car?*, has been added to a group with *Yes/No* input method. Next, the experimenter adds another question B, *What kind?*, that should only be executed if the answer to A is *yes*. The dependency on A with answer *yes* is added, and question B is only executed on the mobile device if the added dependency is achieved. To cope with the small screens for mobile devices an experimenter also has the ability to set an instruction to each element in the survey. That is a piece of text displayed to the participant *before* the actual question is prompted. Questions in paper based surveys are often too long to display on a small mobile device screen, but using the instruction option the experimenter can paraphrase questions from these surveys and split them into an instruction and the actual question.

During the creation of a survey the experimenter can choose to save the survey. As mentioned earlier the survey is represented in XML, hence the saving of a survey involves the transformation of the properties specified in the user interface down to an XML representation. Using JAXB, properties from objects in esmDesk is mapped down to tags in the XML according to the schema definition in appendix C. The byte representation of the survey XML is given to esmService which again handles the actual storing of the survey in the database using !!hibernate. To prevent the user interface from "freezing", marshalling and unmarshalling of XML documents are done in separate threads using Java SwingWorker. SwingWorker is designed to perform time-consuming operations without affecting the performance of the GUI, once its task is completed it will invoke the event-dispatched thread of the GUI to get the attention of the user interface, giving the programmer the ability to show information to the user or perform some post-task action. SwingWorker is included in Java Platform Standard Edition (SE) from version 6.0.

6.6.3 Distribute A Survey

Inside the Survey Creator the experimenter can choose to distribute a survey when he or she consider the created survey to be complete. Distributing the survey involves the selection of a set of participants. The user interface presents the experimenter with a table representing the participant repository. All previously added participants are stored in the database and can be included in any upcoming surveys. It is possible to add and remove participants from the repository. To the left of each listed participant there is a check-box which can be checked to indicate that this participant should get a notification about the new survey. As a selection of participants has been made the experimenter chooses to distribute the survey. Now esmDesk builds up a very small XML message containing the id of the new survey and the address to where Argos is located. Using !!SMSservice esmDesk sends out an SMS message to all selected participants. The message body is the XML message. esmMobile is now responsible for parsing

this SMS message and call the web service method for the retrieval of a new survey with the id contained in the SMS. esmDesk uses the id to look up in the database, and returns the byte representation of the survey XML to the caller of the web method.

Details about the survey are stored in the database along with the XML representing the survey. When a survey is distributed the status of the survey is changed to *Distributed* in the database. If the distributed survey is opened at a later date, the Survey Creator displays the date at which the opened survey was distributed.

6.6.4 Results From a Survey

Each mobile device participating in a survey is responsible for submitting back results given by a participant. From the Survey Creator the experimenter can indicate whether esmMobile should submit answers after the entire survey completes, or after each element is executed. esmMobile uses a web service method defined in EsmDeskWS to submit the results, and results are formatted according to the XML-schema in appendix C. Upon receipt of such a result, esmDesk unmarshals the XML using JAXB and stores the results in the database. Each result is linked to the participant who gave it and the survey in which the answered element resided. When results from all participants in a survey are received, the experimenter can dump them to a file by selecting *Get results* from the main menu of esmDesk. Together with Telenor and department of psychology at University of Tromsø we have agreed on the following format of each line in the file representing the results:

<group name>,<element name>,<answers>,<time used>,<time started>,<status>

The first three values are self explanatory by their names. Latency in ESM is the time measured in seconds from a question is prompted to the participant until an answer is given, and is expressed in the *time used* field above. Knowing exactly when the question or measurement was executed is also essential to ESM and stored in the *time started* field. The last field, *status*, indicates whether the question was answered, ignored or not answered within the given time to answer. The reason for choosing Comma-Separated Values (CSV) instead of XML here is basically because of the request specified by Telenor and the department of psychology at University of Tromsø. The answers are originally received in an XML format before storing them in the database, hence it would be a simple task to represent the results according to some XML format instead of CSV format.

6.7 esmMobile

esmMobile is an engine capable of retrieving, interpreting and running any survey created using esmDesk. The implementation targets the Microsoft Windows Mobile 5.0 Pocket PC edition, and the device used in this development was HTC TyTn[9]. esmMobile is a reference implementation according to the XML specification defined for surveys created using esmDesk.

In figure 6.5 the class diagram for esmMobile is illustrated. The main class of the application is called *EsmMobile* and is a class where the singleton pattern is applied. All four parts from the

[9]http://www.europe.htc.com/products/htctytn.html

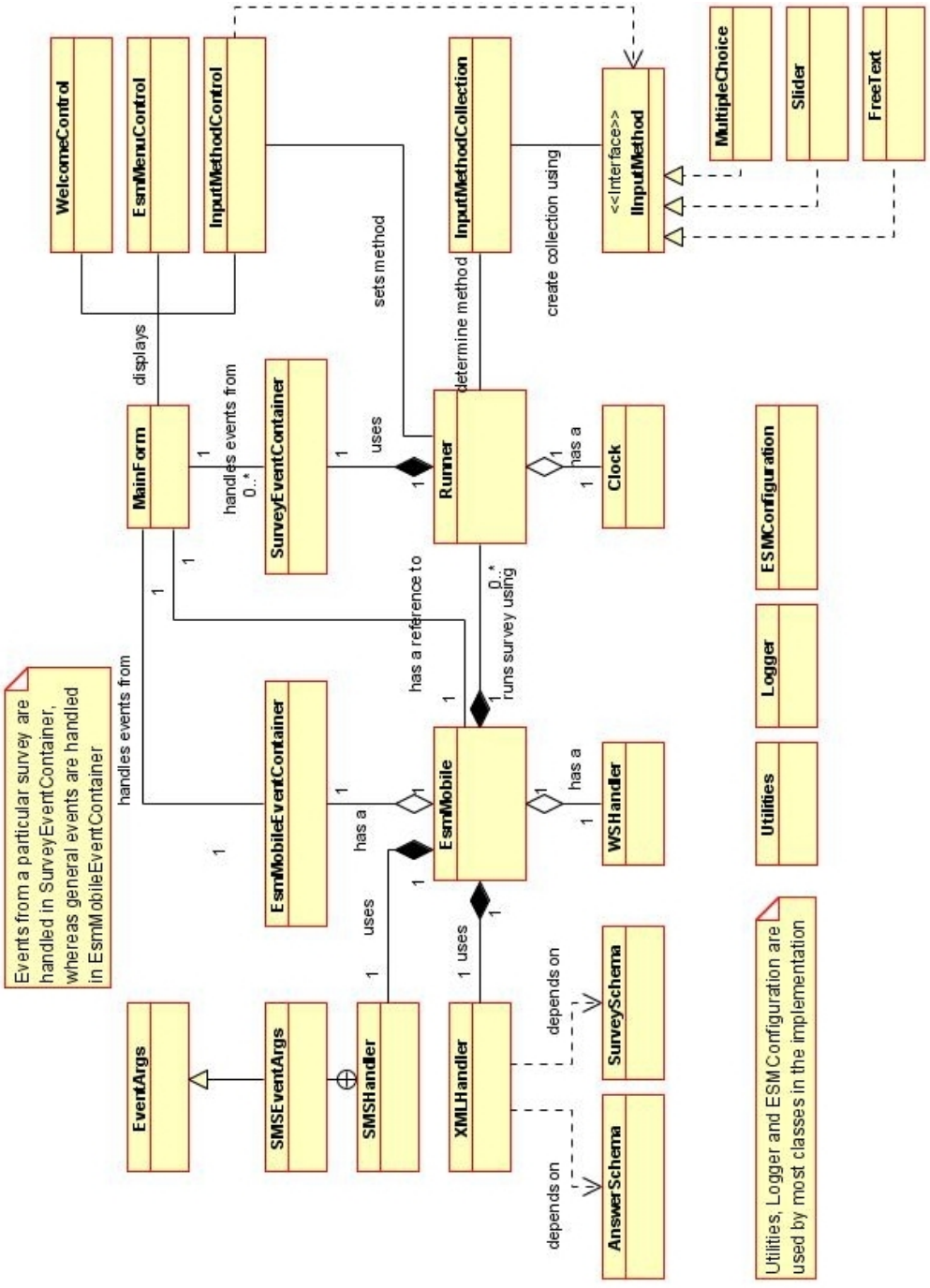


Figure 6.5: esmMobile - class diagram

logic layer in the architecture (see figure 5.8) are linked with the *EsmMobile* class. Exchanging information between esmDesk and esmMobile is done using messages expressed in XML, as a consequence the *File Handler* from figure 5.8 in the design chapter is called *XML Handler* when presenting the implementation of esmMobile. The only handler which is not visible in the class diagram is the database handler, the reason for this is that it is handled within the *EsmMobile* class. *EsmMobile* is responsible for retrieving new surveys, sending back results and starting/resuming all surveys. An actual survey is run using an instance of the class *Runner*, and in a separate thread. This design opens the possibility for the application to run multiple surveys at a time, if that is preferable by the experimenter.

Due to the lack of logging support for mobile development on the Windows Mobile platform, a separate *Logger* was created. Apache has implemented a logging framework for .NET Framework and .NET Compact Framework called log4net[10]. The framework is based on log4j[11] and includes the same features. For our small prototype development the log4net framework was considered to include a lot of features that we would never benefit our prototyping. Therefore a simple Logger was created instead of incorporating log4net into esmMobile. The main purpose of the logger is to report incidents during the lifetime of the application. It is very helpful for both debugging and assuring that events happen in the preferred order, and is an excellent alternative to printing information onto the very small screen of the mobile device. Listing 6.2 depicts how to make use of the logger functionality in different situations. The Logger writes information to a file located in the *log* directory of the application folder on the mobile device.

```
/* initialise the Logger, pass a reference to the object associated
 * with the logger instance */
Logger logger = new Logger(this);

/* log information message */
logger.log(Level.INFO, "Some information");

/* log warning message */
logger.log(Level.WARNING, "Some warning");

/* log error/exception */
logger.log(Level.ERROR, "Some error", exceptionInstance);
```

Listing 6.2: Using Logger in esmMobile (C#)

All applications have certain configurations that are needed for them to work properly (i.e. language settings). For esmMobile, all configurations are kept in an XML file that is loaded on application start-up using the Configuration Application Block from MCSF. In applications developed on the .NET platform it is common to have a set of configuration files defined in XML. Due to limitations on mobile devices, the API for reading and writing configuration files is not a part of .NET Compact Framework. The Configuration Application Block of MCSF provides mobile applications with a subset of the configuration features from the .NET Framework. A configuration is represented as a key-value pair in the XML, and when loaded the configuration is added to a dictionary in *EsmConfiguration*. Any class that needs a configuration value provides a key to the dictionary, and in return it gets the value read from the XML file.

[10] http://logging.apache.org/log4net/
[11] http://logging.apache.org/log4j/docs/

6.7.1 SMS Handler

esmDesk sends out an SMS message to all participating devices as a new survey has been selected for distribution. The SMS Handler within esmMobile is responsible for intercepting this message and removing it from the message queue before the participant sees the message. .NET Compact Framework provides a class called *MessageInterceptor* which allows the programmer to listen for incoming SMS messages, setting a message filter and removing a message from the message queue once received. Applying a filter to the message interceptor causes the interceptor to perform a comparison between the filter and the message received, if the message fulfils the requirements of the filter, it is removed from the message queue and given to esmMobile. In the opposite situation the message is ignored by the SMS handler and left in the message queue for other applications to read it. The filter applied in esmMobile checks the beginning of the message for the string "<?xml". Any message satisfying the criteria is given to the XML Handler and deleted from the message queue.

6.7.2 XML Handler

Information exchanged between esmDesk and esmMobile is expressed in XML. Both when receiving a new survey and sending back the results, the XML Handler is responsible for the unmarshalling and marshalling of objects in C# that represent XML information. Class definitions for these objects are automatically generated using .NET XML Serializer presented in chapter 4.

The previous section mentioned that the content of a received SMS message was also given to the XML Handler. Because of the small size of the XML contained in an SMS no schema has been designed for this message. The XML Handler simply parses the message in search for two tags; id and address, both of which are given to the WS Handler for further processing.

6.7.3 WS Handler

Most of the communication between esmDesk and esmMobile is done using web service technology. A web service provider, in our case Argos, provides a WSDL file defining signatures of all methods in *EsmDeskWS*. Microsoft Visual Studio 2005 uses this WSDL file to generate proxy methods having the same names and signatures as the web methods in the web service of esmDesk. Consequently, when making a call to a web method it appears to be the same as making a call to a local one. As mentioned, esmMobile utilises the Disconnected Service Agent of MCSF, which provides an asynchronous approach to the execution of web service requests. A request is put on a request queue until the device is connected to a network, and the reply is routed to one of two call-back functions; either the one for success or the second for the situation where an exception occurred trying to execute the request.

The WS Handler of esmMobile sets up and maintains a reference to the disconnected service agent, and provides esmMobile with functions for issuing web service requests to esmDesk. The call-back functions for each web method are defined in the disconnected service agent. For example when the WS Handler uses the id and the address provided in an SMS to retrieve a new survey, a successful call to the web method *getSurveyXMLStream* will return in an invocation of the function *OngetSurveyXMLStreamReturn* in the disconnected service agent.

6.7.4 Database Handler

To prevent the application from loosing information about surveys and/or results, esmMobile maintains all such information organised in a Microsoft SQL 2005 Mobile edition database (see appendix E). The Object/Relational mapping is done using DataSet from .NET Compact Framework, where each table in the database is represented as a DataTable. Access to the data in the database is done using an adaptor which is defined for every DataTable in the DataSet. The adaptor basically contains a set of methods for insertion, deletion and update of the content for the table the adaptor belongs to. Instantiation of an adaptor can be done by any object in esmMobile and will give the instantiator the opportunity to call all methods defined for that particular DataTable adaptor.

6.7.5 Input Method Plug-in Framework

The supported input methods of esmMobile are the same as the one listed for esmDesk in section 6.6.2. Each input method is implemented as a Dynamic Link Library (DLL) and can be plugged into esmMobile as described in the design chapter. esmMobile defines a DLL called *esmmobile.inputmethod.common* which includes the interface that must be realised by any future input method and also a definition of an *Attribute*. An attribute is a language specific feature of C# and is used to add meta-information to methods, variables and classes, and is equivalent to annotations in Java. A library containing an input method must label the class containing the input method implementation with the attribute *InputMethodAttribute*.

```
[InputMethodAttribute("MULTIPLE_CHOICE")]
public class MultipleChoice : IInputMethod{}

/* In XML an input method is specified as below
 *
 * <InputMethod>
 *       <methodType>MULTIPLE_CHOICE</methodType>
 * </InputMethod>
 */
```

Listing 6.3: Defining an input method (C#)

Listing 6.3 illustrates how the class that defines the input method *MULTIPLE_CHOICE* should be declared, it implements the IInputMethod interface from the common library and is annotated with the InputMethodAttribute. The meta information stored by the attribute is the name of the input method. This name is the one word description contained in the XML file describing the survey. All input method library files are kept in a sub-directory of the application named *inputMethod*. On application start-up, esmMobile browses this directory looking for library files. For each DLL, esmMobile creates an instance of all classes marked with the InputMethodAttribute using the reflection API of .NET Compact Framework. The instances of input methods are organised in a collection where the one-word description is used as key to get the correct instance of IInputMethod.

6.7.6 Running Surveys

Given that esmMobile has retrieved and stored the new survey in the database the execution of it can start. A survey runs in an instance of the *Runner* class in a separate thread, thus allowing

multiple surveys to be run at the same time. Attached to each *Runner* is an instance of the *SurveyEventContainer* and a *Clock*. The *SurveyEventContainer* receives all events occurring within the survey, and are handled by the runner. Using the *Clock* the *Runner* can determine when to run each element of the survey. Instead of polling the device clock with pre-defined intervals, the implemented *Clock* sleeps for a number of milliseconds and then fires an event caught by the *SurveyEventContainer*. The number of milliseconds it sleeps for is defined by calculating the difference between the current time and the time at which the next element in the survey should be executed. It is important to mention that the *Clock* approach assumes that the system clock on the device is correctly adjusted.

esmMobile is responsible for providing the *Runner* with an id of the survey to be run. The id is unique and the *Runner* uses it to obtain all groups and elements of the survey from the mobile database. A survey is started by picking the first group from the database belonging to the current survey. In the database groups are sorted based on the start date of the first element in the group. From esmDesk it is defined at what time this group should be started, hence esmMobile can calculate the number of milliseconds the *Clock* should sleep for. The event fired by the *Clock* is received in the *SurveyEventContainer* and causes the *Runner* to pick an element from the current group. Once an element has been selected, the input method is determined from its properties in the database and the question is prompted to the user. If the user does not answer the question within the given time-out period, esmMobile minimises the application user interface and removes the question input method. The application registers that the user did not answer and continues the survey by determining the next element. Hopefully the participant will answer most questions, and then the answer together with the time it actually took the participant to answer is stored in the database. A group of elements can be scheduled to run more than ones, and a counter in the database keeps track of how many times the group remains to run within the given survey. As a group has reached its maximum number of repetitions, it is removed from the database, including all elements within it. This is done to prevent the database from growing indefinitely since storage is a limitation for mobile devices. In the situation where there are no more groups for the current survey in the database, the *Runner* concludes that the survey is finished and sets its status to *finished* in the database.

After the completion of each element, esmMobile stores the answer given by the participant. These answers are read out from the database and organised according to the XML-schema in appendix C, before they are sent back to esmDesk using the web service method called *submitResults*. Answers can be submitted immediately after the execution of each element, or when a survey completes. What approach to use is available as a property in esmDesk. For each answer a particular tag is created in the XML, hence sending one or multiple answer requires no specification in the XML.

Assuming that a mobile device is never switched off is not possible because of several reasons. Many people turn their devices of during night-time and sometimes they simply ran out of power. In such situations it is important that esmMobile is automatically started as the device is turned back on, and resumes any surveys that were not completed when the device was turned off. Therefore esmMobile is added to the list of programs started with the operating system on device start-up. Any survey having the status *IN_PROGRESS* in the database is started in a new instance of the *Runner* and the survey continues from where it left off. Since groups are sorted by date and deleted when they have all run, esmMobile simply picks the first element of

the next group to be executed from the database.

6.7.7 Installation

esmMobile is packed into a Cabinet (CAB) file that can be shipped onto any device running Windows Mobile 5.0 Pocket PC edition. This file includes all application files and installation configurations needed to set up esmMobile correctly on the mobile device. From the main menu of esmDesk it is possible to remotely initiate the installation of a set of devices. The experimenter specifies the phone number to all devices that should install esmMobile. esmDesk then creates an SMS message containing a valid URL to where the CAB file can be downloaded. Here esmDesk utilises the web server support from !!jetty in Argos. The person in possession of the device clicks on this link on the receipt of the SMS, the file is downloaded and esmMobile gets installed on the particular device. During installation an icon for esmMobile is added to the list of programs on the device, a click on that icon starts esmMobile and it is ready for receipt of surveys from esmDesk.

6.8 Summary

In this chapter the implementation of the prototype software tools for running ESM based survey is presented. The main task of esmDesk is to provide a graphical user interface for an experimenter where he constructs the survey which is transformed into an XML representation. The XML representing the survey is distributed out to a number of devices running esmMobile. The distribution is done using the technology of SMS and web services. Once a survey has completed the results are sent back to esmDesk, again represented in XML. The implementation illustrates the loose coupling between the two systems since they are developed using different programming languages, Java and Visual C# respectively.

Chapter 7

Experiment

The prototype software developed in this thesis has been done in collaboration with NST, Telenor and University of Tromsø. The final step in our development was to run an experiment with people from NST, Telenor and University of Tromsø to validate the correctness of the developed software with respect to the requirements from chapter 3. This chapter presents details concerning the experiment.

7.1 People

A total of seven persons, including the author were present during the presentation of the two systems and the realisation of the experiment. The persons were:

- *Anders Andersen*
  Supervisor, Department of Computer Science, University of Tromsø
- *Arne Munch-Ellingsen*
  Co. Supervisor, Department of Computer Science, University of Tromsø
- *Dan Peder Eriksen*
  Argos developer, Department of Computer Science, University of Tromsø
- *Joar Vittersø*
  Psychologist, Department of Psychology, University of Tromsø
- *Gunnvald Bendix Svendsen*
  Psychologist, Telenor Research & Innovation
- *Geir Østengen*
  NST

7.2 Environment

The environment running esmDesk consisted of an IBM ThinkPad T60 laptop computer with the following specifications:

- Intel T2400 1.83GHz dual core Processor

- 2GB RAM
- Microsoft Windows XP operating system
- Set up with Argos 1.0 final[1] and esmDesk

The survey created during the experiment was run using a mobile device of type HTC TyTn[2] with the following key features:

- 400MHz Samsung Stacked CPU
- 128 MB ROM, 64 MB RAM
- Windows Mobile 5.0 Pocket PC edition operating system

The entire experiment was carried out using a projector capable of displaying 1400x1050 pixels connected to the laptop computer. Using Pocket Controller Professional from SOTI Inc.[3] it was possible to view and control the device used in the project from the laptop computer. Consequently, everyone was able to see the different actions taking place on the mobile device as well as in esmDesk during the entire experiment.

7.3 The Experiment

The purpose of this experiment was to let stakeholders in the project try out the developed software, provide feedback for future work together with evaluating the outcome of this project. The experiment was started with a brief introduction to the features of both esmDesk and esmMobile. On beforehand the two psychologists were given the task to prepare a set of questions to use in the creation of an experimental survey. After the brief introduction one psychologist was given full control of administrating esmDesk and started creating the survey. The survey created consisted of a total of eight questions spread across four groups. The input methods used were slider, multiple choice, yes/no and free text. Some elements also took advantage of the advanced features of esmDesk, such as dependencies and signal settings. The XML generated by esmDesk and distributed to esmMobile can be found in appendix F.

The two psychologists successfully created a survey using esmDesk, and it was distributed out to esmMobile where it ran as expected. After esmMobile completed the survey, results were sent back to esmDesk and they were written to a file demonstrating how to obtain results from a survey.

7.4 Identified Problems

During the creation of the survey in esmDesk two vital bugs in the user interface were identified. The first one of these had to do with the selection of multiple choice as an input method. Multiple choice consists of several possible answers that the participant can select among, in esmDesk an experimenter can add and remove these answers. In one particular situation the experimenter in our experiment deleted an answer from the list of answers. The answer option

[1] http://sourceforge.net/projects/jargos
[2] http://www.europe.htc.com/products/htctytn.html
[3] http://www.soti.net/

was removed from the user interface, and an attempt to add a new one was done. The new answer was not added to the user interface, at least it did not show. This was due to a repaint bug in the user interface, but the underlying logic had registered the new answer.

The second bug was identified when the experimenter tried to rearrange different elements within the same group using drag-and-drop feature enabled in esmDesk. When an element was dragged from one position to another, an empty new element was automatically inserted. This element had to be manually removed by the experimenter such that it would not be a part of the final experimental survey. Actually what caused this bug has not yet been identified, but most likely it has something to do with the transfer handler, handling all transfers within a drag-and-drop enabled area.

Out of the questions in the survey only one had been configured with the free text input method. This input method required that the participant made use of the qwerty keyboard on the mobile device used in the experiment. For HTC TyTn this keyboard is enabled by sliding it out from beneath, causing the orientation of the screen to switch from being portrait to becoming landscape. This caused esmMobile to minimise the user interface, consequently the question was no longer visible to the participant. At this point we had to manually click on esmMobile logo in the Programs section on the mobile device. This action caused the user interface to be maximised, hence displaying the question. The participant entered and submitted his answer.

7.5 Feedback

After the completion of the experiment everyone present was given the opportunity to provide feedback on the developed software systems. A general opinion among the psychologists was that the user interface for esmDesk was very intuitive and flexible when building surveys. Even though ESM introduces some new concepts into the process of creating a survey, the approach to designing a survey in esmDesk was much like the approach they follow when creating normal paper based surveys. Throughout the cooperation between the different parts of this project the author has tried to emphasise some of the challenges encountered when working in mobile environment, and during this experiment the psychologists got to see some of these in action. The most important one was probably the screen size. Many questions used in traditional surveys may have to be re-written to fit into a small screen along with an input method.

Based on the feedback received improvements have been done to both esmDesk and esmMobile to clarify that it is easy to extend them with functionality. The application block called Orientation Aware Control from MCSF has been included into esmMobile such that a change in the orientation for the free text input method is handled correctly. The input method slider has also been changed. The first version of the slider did not allow the experimenter to specify the number of steps on the slider, but small changes is the user interface of esmDesk and the input method library on esmMobile now allow the experimenter to specify the number of steps that is between the two extremities on the slider.

During the experiment several comments and future requests were made by the psychologists, these limitations are presented in more detail in the future work section in the conclusion chapter of this thesis.

7.6 Summary

With the experiment done in this thesis it was possible to let the stakeholders in the project try out the developed prototype software. The survey created, distributed and ran was fairly simple, although illustrated the main features of both esmDesk and esmMobile. Based on the feedback given, even though relatively short time was given to requirements acquisition, system design and implementation for this project, the outcome is very satisfactory when it comes to demonstrating the concept of using mobile devices in ESM based surveys.

Chapter 8

Evaluation

The main focus of this chapter is to determine whether the implemented systems conform to the requirements listed in chapter 3. Before evaluating the two systems, we start out by reflecting on the method used in the software development, followed by our experiences working with the middleware platform Argos.

8.1 Method

As presented in section 1.5 the method followed in this thesis has been a simplified version of the agile method *Scrum*. The idea behind agile methods is to plan a short iteration at a time and to include developers in the planning of the technical development. Short iterations decrease the risk of developing out of date products, and also produces fast feedback from the stakeholders in a project. Developing a prototype like we have done in this thesis is about demonstrating a concept for a set of stakeholders. Following the Scrum approach we have had a great opportunity to split the development into several stages covering different functionality. After each completed stage, it has been possible to collect impressions from the stakeholders through feedback given when demonstrating our work. Having such a close dialog with the stakeholders as we have had in this project is very valuable. Application development targeting a particular domain often requires domain expertise to identify the correct needs from different customers. Since we had people from different domains, both technical and psychological, we had the expertise required and they helped us a lot when identifying requirements, and provided valuable feedback during the meetings we had.

With the scrum approach requirements are identified as the development takes place. In our work we had to have a set of basic requirements before starting the actual development. People from Telenor, NST and department of psychology at University of Tromsø met with us to draw the basic boundaries of the prototype. From these guidelines we formulated a set of requirements which was presented a week later. These requirements made up the starting point of the development. During the development new requirements were identified by all parts involved in the project, some were included in the prototype while others were left out due to timing constraints and/or degree of importance to illustrate the concept. The final part of the project involved an experiment and two presentations for Telenor R&I where we got to demonstrate the prototype. Being able to run an experiment was very useful because we saw how the stakeholders actually interacted with the developed prototype.

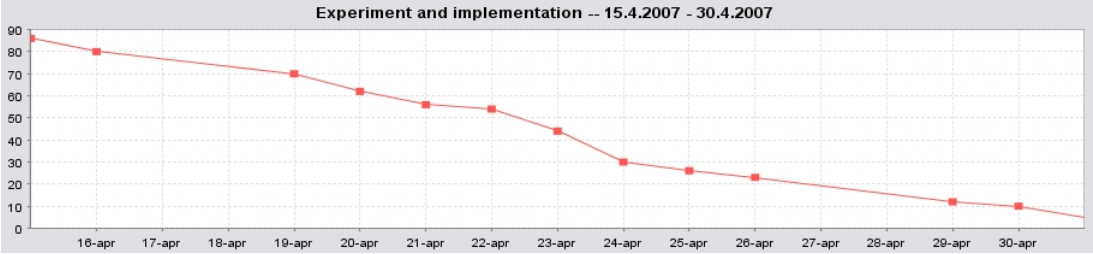


Figure 8.1: Burn-down chart for a sprint within this thesis

Organising the different requirements for the two systems was done using ScrumWorks[1]. The software provides a comprehensive user interface working with requirements and sprint details following the Scrum approach. As new requirements were identified, these were added to the repository of all requirements, and organised by the Scrum Master and the author according to their assigned priorities. For every sprint planning meeting requirements were taken from the repository and inserted into the area representing the next sprint. The most difficult part associated with using Scrum was estimating the required working hours for each task and backlog item. In the beginning, the estimates tended to be too low, consequently the number of tasks included in a single sprint were too many compared to the time spent on completing them. A total of eight sprints have been planned and carried out. There were a couple of sprints that had to be adjusted due to unexpected implementation problems. In those situations the Scrum Master together with the author determined what tasks to remove from the sprint backlog and back to the product backlog. With experiences in estimating the number of hours required for a particular task, a revision of the sprint was not required. One last thing worth mentioning about the ScrumWorks application is that it provides a burn-down chart for each sprint. The sprint period is shown along the x-axis, while the y-axis displays the hours required to complete the sprint. As tasks are completed, the number of hours left decreases, and at the end of a sprint the graph should be something like the one displayed in figure 8.1.

8.2 Argos Middleware Platform

From the outline in chapter 4 we saw that Argos is a personal middleware platform shipped with a set of services to get the developer started using it. Because of its lightweight and the set of systems service it was chosen as the platform for the desktop application development in this thesis. By exploiting Argos system services we were able to quickly integrate the use of persistent storage and web services into our application. Unfortunately, little or no documentation existed at the time we started our development using Argos. The main reason for this was that its design, implementation and documentation were part of another student's Master's Thesis. Without any documentation it was challenging to figure out the workings of Argos, but with help from the developer and other people with experience in using the predecessor to Argos, namely *A Personal Middleware System (APMS)*, we were able to quickly establish an understanding of the architecture and workings of Argos.

[1] http://www.scrumworks.com

For our development we deployed one system service (!!SMSservice) and two user services (esmDesk and esmService). A great advantage of Argos was the straight forward approach to creating and deploying services, and the fact that we are given permissions that an enterprise middleware system would not allow us. These permissions include the possibility of creating threads plus folders and files within the platform's file tree. When using some system services we found it a disadvantage that we had to start and stop the entire Argos for each time a new version of our user service was deployed into it. Argos itself supports hot deployment, but the system service !!hibernate did not, due to limitations in the Hibernate implementation. Working with several services within Argos we saw the need for a way of starting and stopping individual components, for example provided by a graphical user interface.

Building applications on top of Argos is valuable in many ways. First of all, developers are given a head start by the fact that Argos is shipped with a set of very useful system services needed by most applications. Secondly, Argos is component based and it is preferable to follow the Component-Based Software Development (CBSD) approach such that distinct parts of an application can easily be changed without having an impact of the entire system. This also opens for the easy integration of components written by other developers, again illustrated by the SMSservice developed in this thesis. A last point worth mentioning is the fact that Argos utilises JMX making it relatively straight forward for developers familiar with JMX to monitor components within Argos.

8.3 Functional Evaluation

Chapter 3 outlined the requirements for the two systems developed in this thesis. Roughly, the two systems together were to provide an experimenter with the ability to create, distribute and run surveys based on the Experience Sampling Method (ESM) approach. Like other parts of this document we have decided to separate the functional evaluation of the two systems esmDesk and esmMobile. The missing functionality of each system is identified based on the feedback given during the experiment ran as part of this thesis.

8.3.1 esmDesk

The functional requirements for esmDesk can be summarised into letting an experimenter create, distribute and view results of ESM based surveys. Stated in the design chapter section 5.1, the prototype software in this thesis is not a fully realisation of all the requirements.

Missing Functionality

During the experiment for this project the psychologists were able to use esmDesk. As they created the experimental survey they came across some features that they wanted the system to hold. One of the most significant requests was a feature where the system could store a set of properties related to a particular question in the repository. For example the system could store the previously used input method and other settings for the particular question. The motivation behind having such a feature is that most questions will have the same input method and time-out settings independent of the survey in which they reside. A second important request has to do with the specification of group properties. Currently, the experimenter can specify start time for a particular group, the number of repetitions and the delay between each repetition.

To allow for full utilisation of the interval-contingent protocol of ESM it should be possible to specify that a group of elements should be run at a specific intervals during the day. Say you have a group of elements related to eating lunch, such elements should typically be scheduled for execution between 1100 and 1300 every day, and maybe with random intervals within these hours. Just specifying seconds and minutes between each execution of a group can potentially result in the situation where a participant gets asked questions in the middle of the night.

Flexibility

One of the most important features of esmDesk is its flexibility during the preparation of a survey. Even though many surveys may have a set of elements in common, it is essential to give the experimenter free rein as he or she is putting together a new survey. When adding a question to a survey in esmDesk, the experimenter can specify several properties for that question. These settings improve the flexibility and usefulness of the entire application. Another feature worth mentioning in this context is the possibility of rearranging both elements within a group, and the groups representing the entire survey. With this property the experimenter can reuse an already distributed survey by for example changing the execution sequence of the groups to determine whether this has an impact on the answers given by a participant. With improvements in the prototype it might be an idea to allow the experimenter to open several surveys and copy entire groups into a new survey, this way decreasing the time spent on creating a new survey.

Wireless Advantage

In most of the ESM systems presented in chapter 2, we observed that they all required a device to be physically connected to the computer creating the survey, to transfer the actual survey and receive results. A functional advantage of our ESM system is the possibility of transferring surveys and results between esmDesk and esmMobile without physically connecting the devices to a computer. This improvement basically means that the experimenter is not troubled by transferring a created survey to each individual device. As the survey is created, the experimenter instructs esmDesk to manage the distribution of the survey. After completion of a survey, the results are automatically returned to the computer that distributed the survey. This feature implies that the experimenter never has to be in possession of the device to run a future survey. This holds as long as the mobile device is turned on, esmMobile is installed and the SIM-card is operable. Given these requirements the mobile device handles both the retrieval of a new survey and the submission of results from a completed survey.

8.3.2 esmMobile

esmMobile is a reference implementation for interpreting and running surveys created using esmDesk. Based on the design and implementation of the application its functionality can be summed up as retrieving surveys created in esmDesk, organise the contents of the survey persistently on the mobile device and run all elements of the survey. As the participants answer questions, these are stored and submitted back to esmDesk for further analysis.

Limitations

During the experiment with both systems, a stronger emphasis was put on the functionality of esmDesk because it actually defines the format in which the survey should be represented. The

device used in the development was a PDA supporting both landscape and portrait orientation of its screen. A change in orientation is done when the user devices to make use of the qwerty keyboard available on the device. In our experiment we saw that the reference implementation lacks support for handling a change in orientation. Because of a covet put forward by the psychologists, any future development of the prototype targeting devices with two orientations, significant effort should be put into handling both orientations, screen sizes and aspect ratios. If the Windows Mobile platform is chosen for further development it is recommended to look into the application block called Orientation Aware from Mobile Client Software Factory (MCSF), which can assist in the creation of user interfaces that support different orientations and screen sizes. As mentioned in chapter 7 the input method *Free Text* has incorporated the features of the Orientation Aware Control application block from MCSF.

From esmDesk the experimenter can configure a mobile device to either signal a beep or vibrate in front of a question prompted to the participant. In esmMobile the implementation of this functionality only works if the user of the mobile device has enabled the system sounds in the operating system. Such a limitation might benefit the participant in some situations. Say the participant is in a meeting where mobile phones are not allowed; hence he or she will switch to silent mode on the phone. When a question is prompted the device will *not* signal the participant, consequently an answer might never be collected. On the other hand, this limitation also makes it impossible to guarantee the ESM protocol signal-contingent. It is important to mention that the limitation is bound to the platform chosen in this development and since the client is loosely coupled from the creation of the survey in esmDesk the problem might be able to solve using another reference implementation.

Input Methods

The supported input methods are the ones listed in the requirements chapter, but during the experiment we saw that surveys were in most situations constructed with elements that required the multiple choice input method. Specifying what input method to use is relatively complex since it varies between the implementations that are to run the actual survey. Our approach to indicate in the XML representing the survey what input method to use for an element, has proven to work satisfactorily. By separating the input method into library files, we opened the possibility for adding additional input methods without having to change the source code of the client implementation. Using the plug-in framework designed for the input methods we mentioned that a reference implementation could request input methods that it does not contain. Such a feature puts an additional requirement on the desktop application. Knowledge about the kind of device asking for an input method would be required to be able to return the library targeting the correct platform. An approach to identifying variations in mobile devices can be found in [32].

The current implementation displays a logo together with the number of questions left in the survey and how much time a participant have left to answer the current question. To cope with the challenge of the small screen, it would be an improvement to allow the experimenter to choose whether this kind of information should be shown or not. Leaving out this information, provides a larger area available for the particular question and input method, meaning that the length of the question or the number of choices (using multiple choice) can be increased without

exceeding the boundaries of the screen.

Architectural approach

The architecture of esmMobile is based on the smart client approach where the client, in this case esmMobile, contacts the server, esmDesk, whenever it is finished running some task. The implications of choosing such an approach are that a particular application is needed for each operating system that is to run it. A platform independent alternative could be to implement the application using for example J2ME, unfortunately it is very common for each device vendor to provide their own SDK for developing Java on their devices. Hence, it is not so platform independent at all. The application developed also has to be installed before being able to run surveys in our context. We have tried to meet these limitations by developing a utility for esmDesk that allows an experimenter to issue the installation of the required software on a set of mobile devices before starting to create surveys. Theoretically, an experimenter never has to physically be in contact with the devices to be used in surveys. Our approach is based on sending an SMS message to instruct the participant to download and install the required software. Given that the participant is able to click on the link given in the SMS message, the software will be installed and is ready for use. Of course, this approach requires that the software downloaded by a client is compatible with the operating system running on the device. Again, considerable effort must be put into obtaining knowledge [32] about what kind of device that tries to download the particular software required to run surveys. Without the smart client approach it would not have been possible to persistently store the answers collected from a participant. Such a situation would require the answers to be submitted immediately as they were given. Because a mobile device is not guaranteed connectivity, this approach was abandoned at an early stage in the development phase.

8.4 Non-Functional Evaluation

Due to the time constraints of this thesis, the non-functional requirements have not gotten as much attention as the functional requirements. This section presents an evaluation of some important aspects of the non-functional requirements for esmDesk and esmMobile as a whole.

8.4.1 Interoperability

A very important concern in our development was to make the coupling between the application creating surveys and the one running them as loose as possible. By making a loose coupling our approach to ESM is not bound to a particular set of devices or operating systems. The entire survey and all its properties are expressed in the mark-up language XML, hence the only requirement put on a terminal to be used is an engine that is able to interpret the contents of the XML. However, we do not overcome the problem of having to implement an application for every type of device, but what we do is that the communication between esmDesk and a mobile application is not bound to a particular platform or operating system. The user interface required for running a survey must be implemented for each platform, consequently if a device does not have the input method required it might not be able to run the particular survey. For future development, a mechanism for handling missing input methods must be designed and implemented, possibly as a system service in Argos.

As a second attempt to provide interoperability the communication between esmDesk and esmMobile is mainly done using the web service technology. It is worth mentioning that this approach introduces additional overhead in two important areas. The first is the overhead of creating and parsing the SOAP messages exchanged in the technology, SOAP messages are structured in XML, meaning that we have an XML (the survey) representation in another XML (the actual SOAP message). Secondly, using XML increases the size of the original message on average by 400% [13], hence it might be an idea to introduce some technique for compressing the data sent between esmDesk and esmMobile.

8.4.2 Performance

During the experiment we were given feedback on the delay between an answer to a question was given until the next question appeared on the screen. The delay between each element in the experimental survey was set to zero seconds, and the physiologists expected another question to appear immediately after a previous question had been answered. The implementation of esmMobile needs to store the answer given by a participant before it determines the next element to run. At first glance it could be a solution to determine the next element to be executed the same time as the previously is shown, this way it would be possible to display the next element while storing the answer. Unfortunately, this is not attainable. The main reason is that a question could have dependencies. If an element has dependencies, it is impossible to predict if it is to be run or not. As a consequence of the relatively slow hardware available on mobile devices, we did not find an approach that could display questions immediately following each other. The unwanted delay varies determined by the set of elements that depends on the element just answered.

8.4.3 Usability

The user interface for esmDesk was given several complements during the experiment, and most of them were concerned with its easy layout and concept. The use of touch-screen for esmMobile was thought to be a very user friendly approach to ESM, but the experiment showed that not all parts of the user interface for our device was as user friendly as one would like. For example the slider input method was too small and very difficult to operate. Participants not familiar with this kind of device would have a problem understanding how to operate the slider. Another aspect is that most people does not have a touch-screen enabled mobile device, thus to run an ESM survey the experimenters would have to purchase a set of devices to be used. If a regular mobile phone were to be used, participants could potentially make use of their private mobile phone devices. With the experiment we observed that most questions were designed with answers on a scale from 1-7, meaning that an implementation using the regular number keys on the mobile device would be sufficient for the participant to be able to answer most questions.

The implementation of esmMobile allows the participant in any survey to use other functionality of the mobile device while participating in a survey. Hence, he or she might be interrupted while making a phone-call, writing a message or edition their personal calendar. If a person is making a phone-call and a question is prompted from esmMobile, the participant is not notified even if a signal or vibration has been set. Another issue is that the participant is most likely holding the device up against their ear, hence not seeing that a question is prompted. The time-out

for that particular question will probably run out, and the system registers the question as not answered due to time out. At this point it had been useful to be able to add extra information to why it was not answered. The system could for example be able to determine whether the user was in the middle of a phone call or not. In some situations the participant may not want to answer a question, and the option *ignore* allows him or her to do that. When experimenters are creating surveys, they should keep in mind that questions are not to be prompted too often. If so, they might interfere too much with the daily life of a participant, thus making them ignore all questions or some might be tempted to give up the wrong answers.

8.5 Other Area of Utilisation

The main focus for this thesis has been ESM based surveys. After giving two presentations of the work done to Telenor Research & Innovation both in Tromsø and Oslo, Norway, another area where the technique could be applied has been identified. The cost of launching new products is for most companies very high and in a situation where new products are not accepted by the consumers, the company looses a huge amount of money. During pilot testing of new products the ESM technique combined with sensors attached to a set of test-customers, could help a company determine people's reactions and impressions of the product to be released. If it turns out that the group of pilot customers does not approve the solution, the company can choose to give it up or try to improve it according to feedback received from questions asked using the ESM approach.

The fact that our approach also focuses on a wireless approach both for sending out surveys and retrieving results makes it very easy to initiate new surveys of any kind.

8.6 Summary

In this chapter we have given a functional and non-functional evaluation of the ESM prototype developed in this thesis. With reference to the system goal of section 3.1 we have fulfilled the requirements for this thesis. Using the prototype an experimenter can create and distribute a survey using esmDesk. The survey is run on devices that have installed esmMobile, which also makes sure to submit all answers back to esmDesk once a survey is finished. This way the experimenter is also able to view the results from participants in a survey. We mentioned several advantages with the prototype and the greatest is probably the fact that an experimenter can distribute surveys and retrieve results from devices without physically connecting them to the computer which they are working on.

It is important to emphasise that the design and implementation in this thesis is done as a prototype and as mentioned in section 1.3 prototypes can contain both knowledge and non-knowledge, even though the non-knowledge is removed from the implementation there might still be leads to were they have occurred. We have found prototyping to be a very efficient approach in developing new software concepts, especially since we have had the opportunity to work with the people that actually put forward a request for an ESM prototype.

Chapter 9

Conclusion

This chapter concludes the work done in this thesis by discussing our achievements with respect to the problem definition introduced in chapter 1.

9.1 Achievements

The problem definition of section 1.2 states:

> *The main focus in this thesis is to develop a software tool for simplifying the process of creating, running and collecting results from surveys based on the ESM approach. The development shall be done with help from the middleware platform known as Argos*[1] *to clarify how applications built on top of Argos can benefit from its features.*

We have developed a prototype application, esmDesk, allowing an experimenter to create and distribute surveys based on the ESM-approach to capture immediate user experiences. ESM defines three protocols called interval-, signal- and event-contingent, which an experimenter can exploit and combine in the application. During the creation of a new survey it is possible to set several properties related to the execution of the survey itself, and specific details of each element being part of the newly created survey. This way we achieve great flexibility. The experiment run with psychologists from Telenor and Department of Psychology, University of Tromsø , confirmed the high degree of usability achieved for esmDesk.

The second developed application is the engine, esmMobile, responsible for the interpretation of any survey created using esmDesk. The engine extracts the information contained in the XML file representing a survey, and stores it persistently before running it. The survey is run according to preferences set by the experimenter in esmDesk, and answers are persistently stored before they are returned to esmDesk as the survey completes.

A wide range of mobile operating systems and devices exist making it very difficult, in some situations impossible, to develop a single application to run on all types of devices. Since the work in this thesis involved a communication part between a desktop application and a set of mobile device, an important concern in our approach has been interoperability. Information sent and received by esmDesk is not bound to any mobile operating system or particular device,

[1] http://argos.cs.uit.no

all messages are expressed according to a well-formed XML-Schema. Consequently, the implementation of esmMobile is a reference implementation of an engine capable of interpreting the information expressed in messages sent by esmDesk.

We have seen that building an application on top of Argos is very useful because we can put more focus on the actual problem at stake, instead of technical sub-tasks that are common for most applications.

We believe the achievements stated here are coincident with the problem definition from section 1.2, hence we conclude that the requirements for this thesis are met.

9.2 Final Thoughts

The work done in this thesis has shown that communication between people from different domains can be challenging. Sometimes it is very difficult to translate the requirements from one domain down to the computer science domain. In many situations ideas put forward by people in a team turns out to be very different from what was actually asked for, because the interpretation of an expression is different between distinct domains. To prevent misunderstandings from happening, we tried to keep a close dialog with people from the psychology domain throughout our work.

esmDesk was built on top of a middleware platform which was part of another students Master's Thesis. Dealing with software being work-in-progress or not fully tested, introduce an additional concern developing applications depending on that software. We have learnt that it can be difficult to locate errors and/or false behaviour when they might reside in parts outside the application to be built. Before deciding to incorporate not fully tested software, sufficient documentation describing the software should be available.

9.3 Future Work

In this section we present issues that should be explored to improve our ESM prototype.

- *Additional input methods*
  Most new mobile devices are equipped with both a digital camera and a voice recorder. Allowing two new input methods based on these technological features would decrease the burden on the participant in addition to provide a more detailed answer. Experimenters in a survey aimed at people hit by Diabetes Type 2 might be interested in what kind of food the participants consume. Instead of describing the food eaten using free text, the participant could simply take a photo or record a description of it.

- *Emulator support*
  After completing the creation of a survey, it would be very helpful for the experimenter to test the survey using an emulator built into esmDesk.

- *Advanced interval specification*
  For the interval-contingent protocol of ESM it would be very useful to be able to specify weekdays, months and at what hours during the day a question shall be asked by the system. This way the participant is not interrupted during nighttimes etc.

- *Remembering properties*
  When adding a question to a survey it would be helpful to be able to store the properties set by an experimenter the last time this question was added to a survey.

- *Decouple esmDesk user interface from Argos*
  Currently, the user interface of esmDesk is deployed as a user service in Argos. Decoupling it from Argos would make it possible to launch the application from another location using for example Java WebStart. Then the user interface could communicate with Argos using JMX.

- *Survey scheduling*
  If the SMS service for Argos has failed it could be possible to schedule the distribution of surveys. This way the system would send out a notification to the mobile devices as soon as the service is up and running again.

- *Sensors*
  Another Master's Thesis written by Mats Mortensen at University of Tromsø handles the communication with sensors for mobile devices. Integrating the work done in his thesis with the work done here opens the possibility of retrieving context information in an ESM based survey.

- *Security*
  Due to timing constraints security has not been an area of concern in our development, future improvements of the prototype should devote time to handle different security issues, such as separating the answers from the participant who actually gave them.

9.4 Conclusion

Throughout this thesis we have seen that immediate user experiences can be captured by exploiting the approach known as Experience Sampling Method (ESM). By combining the method with today's modern mobile technology it is possible to ask participants questions at specific intervals during the day, after certain events or following a particular signal given by a mobile device. The prototype designed and implemented is built on top of Argos and allowed us to put more focus into the business logic of the application because of the easy integration of services that solved technical sub-tasks. The development done in this thesis proofs the concept in which an experimenter specifies all properties of an entire survey, distributes it out to a set of participants and waits for the results to arrive.

Bibliography

[1] Tamlin Conner. Experience sampling resource page. Technical report, Department of Psychiatry, University of Connecticut Health Center, 2006. http://psychiatry.uchc.edu/faculty/files/conner/ESM.htm.

[2] G.L Clore M. D. Robinson. Belief and feeling: Evidence for an accessibility model of emotional self-report. *Psychological Bulletin*, 2002.

[3] S. S. Intille J. Ho. Using context-aware computing to reduce the perceived burden of interruptions from mobile devices, 2005.

[4] Phillip G. Armour. The five orders of ignorance. *Communications of the ACM*, 43(10):17–20, October 2000.

[5] M. Satyanarayanan. Fundamental challenges in mobile computing. In *Principles of Distributed Computing '96*, 1996.

[6] M. Beedle K. Schwaber. *Agile Software Development With Scrum.* Pearson Education, 2002.

[7] E. Diener C. N. Scollon, C. Kim-Prieto. Experience sampling: Promises and pitfalls, strengths and weaknesses. *Journal of Happiness Studies*, 2003.

[8] Tamlin Conner. Experience sampling methods: the theory and practise of measuring behaviour in situ, 2004.

[9] Chen Hsiang. Digitization of the experience sampling method: transformation, implementation, and assessment. *Social Science Computer Review*, 24, 2006.

[10] S. Lucy H. Weiss, D. Beal. Construction ema studies with pmat: The purdue momentary assessment tool, 2004.

[11] Mr Mehl, Jw Pennebaker, and Dm Crow. The electronically activated recorder (ear): a device for sampling naturalistic daily activities and conversations. *Behavior Research Methods, Instruments, and Computers*, 33, 2001.

[12] C. Kinsman J.P. McManus. *Visual Basic .NET Developer's Guide to ASP .NET, XML and ADO.NET.* Addison Wesley Professional, 2002.

[13] Arun Nagarajan Anbazhagan Mani. Understanding quality of service for web services. *SOA and Web Services*, Jan 2002.

[14] Daniel Winkowski Michael Cokus. Xml sizing and compression study for military wireless data. In *XML 2002*. deepX, 2002.

[15] M Honkala and P Vuorimaa. A configurable xforms implementation. In *Proceedings. IEEE Sixth International Symposium on Multimedia Software. Miami, FL, USA. 13 15 Dec. 2004.*, Proceedings.-IEEE-Sixth-International-Symposium-on-Multimedia-Software. 2004: 232-9. IEEE Computer Soc, Los Alamitos, CA, USA, 2004.

[16] World Wide Web Consortium. Browser statistics, 2007. http://www.w3schools.com/browsers/browsers_stats.asp.

[17] J Wusteman. About xml: from ghostbusters to libraries - the power of xul. *Library Hi Tech*, 23, 2005.

[18] T Armstrong, O Trescases, C Amza, and Lara E de. Efficient and transparent dynamic content updates for mobile clients. In *MobiSys2006. The Fourth International Conference on Mobile Systems, Applications and Services. Uppsala, Sweden. 19 22 June 2006.*, MobiSys2006.-The-Fourth-International-Conference-on-Mobile-Systems,-Applications-and-Services. 2006: 56-68. ACM, New York, NY, USA, 2006.

[19] Ian Sommerville. *Software Engineering*. Addison-Wesley Publishers Limited, 7 edition, 2004.

[20] Dahai Li Luiz Fernando Capretz, Miriam A. M. Capretz. Component-based software development. In *The 27th Annual Conference of the IEEE Industrial Electronics Society*. IEEE, 2001.

[21] Hong Mei et al. Qianxiang Wang, Feng Chen. An application server to support online evolution. In *International Conference on Software Maintenance (ICSM02)*. IEEE, 2002.

[22] IBM. Ibm soa foundation: providing what you need to get started with soa. Technical report, IBM, 2005.

[23] Sayed Hashimi. Service-oriented architecture explained. Technical report, O'Reilly Windows devcenter.com, 2003.

[24] Dan Peder Eriksen. Design and implementation of the second generation of apms middleware. Technical report, University of Tromsø, Norway, 2006.

[25] Elena Malykhina. Analysis: How smartphone platforms compare. *InformationWeek*, 2007.

[26] iAnywhere Solutions. Smart client architectures for the mobile developer. Technical report, iAnyware Solutions, 2006.

[27] Trace Galloway. Principles of xml schema design. In *XML Conference and Exposition 2002*, 2002.

[28] Ayesha Malik. Create flexible and extensible xml schemas. Technical report, IBM DeveloperWorks, 2002. http://www-128.ibm.com/developerworks/library/x-flexschema/.

[29] S.B. Navathe R. Elmasri. *Fundamentals of Database Systems*. Pearson Addison Wesley, fourth edition edition, 2004.

[30] Talin. A summary of principles for user-interface design. Technical report, Viridia.org, 1998.

[31] Telenor Mobile. Content provider access, protocol specification and how to get started. Technical report, Telenor Norway, 2003.

[32] Vander Alves. Identifying variations in mobile devices. *Journal of Object Technology*, March 2003.

Appendix A

Functional Requirements

This appendix contains the functional requirements identified for both esmDesk and esmMobile. In the description of each requirement the terms *shall*, *should* and *could* are used to indicate the importance of an identified requirement. *Shall* is used for requirements identified as mandatory, *should* indicates recommended but optional requirements, and finally *could* is used to provide illustrative examples of how requirements might be rather than indicating mandatory or recommended requirements. For both systems the term *element* is frequently used. An element can be either a question or a measurement method (attached sensor).

A.1 esmDesk

The tables in this section contains the identified requirements for each use case as described in section 3.3.1. All requirements in this section are given a unique identifier that starts with Functional Requirement esmDesk (FRD) followed by a number. When referring to the term *system* in this subsection we mean esmDesk.

Use case D-1: Create Survey

Requirement ID	Description
FRD-100	The system shall allow the experimenter to create an ESM-based survey.
FRD-101	A survey shall include: • A title. • Author - the name of the creator of the survey. • Description - a one sentence description of the survey. • Date - the date the survey was created. • Status -has the survey been distributed or not. • Elements - a set of elements constituting the survey.

continued on next page

continued from previous page

FRD-102	The system shall support the three ESM protocols. These are interval-, signal- and event-contingent as explained in section 2.2.
FRD-103	The system should allow the experimenter to select between two methods of gathering information from participants. The first is the traditional approach where the participant answers a particular question. The second is automatically reading information from for example sensors attached to the device.
FRD-104	The experimenter should be able to define a welcome and an end message for the current survey. The welcome message could typically be displayed to the participant before the first element of the survey, and the end message could be the last information given to the participant at the end of a survey.
FRD-105	The experimenter of a survey shall be able to select whether the answer/measured result from an element is to be returned immediately to esmDesk, or after all elements of a survey has completed.

Table A.1: Requirements: Create Survey

Use case D-2: Add Question

Requirement ID	**Description**
FRD-106	The experimenter shall be able to add a question to a survey.
FRD-107	The experimenter shall be able to create a question to be part of a survey.
FRD-108	When a question has been created, it should be stored in a repository such that an experimenter can make use of this question in the creation of any future survey.
FRD-109	It shall be possible to add and remove questions from the repository.
FRD-110	When adding a question to a survey, the experimenter shall specify the input method and allowed timeout for that question. Timeout could be the portion of time the participant is given to complete the answer of the added question.
FRD-111	In addition to specifying input method and timeout, the experimenter should be able to set whether a signal shall trigger the participants attention. A signal could be: • A beep • A vibration

continued on next page

continued from previous page

FRD-112	The experimenter shall specify the action to take if a participant does not reply within the given timeout. The actions available could be either to throw the question away or later prompt it to the participant again.
FRD-113	The experimenter shall be able to set dependencies for a question. A dependency could be answers from other questions in the survey, specific values read from sensors or event dependencies.
FRD-114	A question can only depend on an element that is prior to this question in the execution of the survey.

Table A.2: Requirements: Add Question

Use case D-3: Select Input Method

Requirement ID	Description
FRD-115	The experimenter shall be able to select how the participant should answer a created question.
FRD-116	All available input methods shall be organised in a repository. From this repository the experimenter shall be able to view example illustrations of the different input methods as the could be displayed on a mobile device.
FRD-117	The system should support the following input methods: • Choose *one* item from a list of alternatives. • Choose *one or more* items from a list of alternatives. • Input free text. • A slider.

Table A.3: Requirements: Select Input Method

Use case D-4: Add Measurement

Requirement ID	Description
FRD-118	The experimenter shall be able to select among different automatic measurement methods to include in the survey.
FRD-119	All measurement methods shall be kept in a repository.
FRD-120	It should be possible for an experimenter to add and remove measurement methods from the repository.

Table A.4: Requirements: Add Measurement

Use case D-5: Group Elements

Requirement ID	Description
FRD-121	It shall be able to group elements together.
FRD-122	The experimenter shall be able to add new groups and delete already created groups. When a group is selected for deletion, all its elements shall be removed from the survey too.
FRD-123	The experimenter shall be able to drag an element from its respective repository and drop it into the wanted group for that element.
FRD-124	A group shall have the following properties: • Start - date and time when to run the elements of this group. • Sequence - fixed or random element execution. • Intervals - number of times to execute elements of the group. • Delay between each element.
FRD-125	The order in which the groups are displayed to the experimenter shall reflect the sequence in which the groups are executed in a survey. The experimenter shall be able to reorganise this sequence.
FRD-126	As a consequence of reorganisation of the groups, the system should remove no longer valid dependencies and make sure that the start time of the moved group does not conflict with the prior groups start and total execution time. This is to prevent collisions.

Table A.5: Requirements: Group Elements

Use case D-6: Save Survey

Requirement ID	Description
FRD-127	The experimenter shall be able to save a survey. Saving a survey means to store all the properties, questions measurement methods stored for a created survey such that it can be retrieved for later use.

Table A.6: Requirements: Save Survey

Use case D-7: Open Survey

Requirement ID	Description
FRD-128	It shall be possible to open a previously created survey using esmDesk.
FRD-129	The experimenter shall be allowed to have multiple surveys opened at a time.
FRD-130	When a survey is opened the experimenter shall be able to change all its contents apart from the date the survey was created and the name of the author who created it.

continued on next page

continued from previous page

FRD-131	If a survey has been distributed the experimenter shall be able to see when it was distributed.

Table A.7: Requirements: Open Survey

Use case D-8: Distribute Survey

Requirement ID	Description
FRD-132	The experimenter shall be able to distribute a created survey out to a set of participants.
FRD-133	The system should keep track of which participants have received and started the survey, and which have not.
FRD-134	The system should provide information on why a participant has not started or received a distributed survey.

Table A.8: Requirements: Distribute Survey

Use case D-9: Select Participants

Requirement ID	Description
FRD-135	The system shall maintain a repository of participants that have registered for taking part in surveys created using esmDesk.
FRD-136	When a survey is selected for distribution, the experimenter shall select participants from the repository.
FRD-137	The experimenter should be able to invite participants that are not registered in the repository.
FRD-138	For each participant the system shall store the following information: • Name • Age • Phone number • Gender • Comment

continued on next page

continued from previous page

FRD-139	The information about each participant shall be stored in a repository.
FRD-140	It should be possible to change the type of information stored for each participant.

Table A.9: Requirements: Select Participants

Use case D-10: Delete Survey

Requirement ID	Description
FRD-141	The system shall allow the experimenter to delete a previously created survey.

Table A.10: Requirements: Delete Survey

Use case D-11: View Results

Requirement ID	Description
FRD-142	The system shall organise the results received by participants of a survey.
FRD-143	When results from all participants are received, the system should notify the experimenter such that he/she can view the results.
FRD-144	The results shall be organised in such a way that they easily can be imported into external tools for further analysis.

Table A.11: Requirements: View Results

Use case D-12: Install esmMobile

Requirement ID	Description
FRD-145	The system shall provide the experimenter with an option of sending out the esmMobile software to a set of mobile devices.

Table A.12: Requirements: Install esmMobile

A.2 esmMobile

The tables in this section contains the identified requirements for each use case as described in section 3.3.2. All requirements in this section are given a unique identifier that starts with Functional Requirement esmMobile (FRM) followed by a number. When referring to the term *system* in this subsection we mean esmMobile.

Use case M-1: Run Survey

Requirement ID	Description
FRM-100	The system shall allow for the participant to take part in one or more surveys created using esmDesk.

continued on next page

continued from previous page

FRM-101	The participant shall not be presented with more than one question at a time during a survey.
FRM-102	The system shall **not** notify the participant every time it runs parts of the survey that does not require answers from him/her. Such a part could be when the system reads out or asks for a value from a sensor attached to the mobile device.

Table A.13: Requirements: Run Survey

Use case M-2: Answer Question

Requirement ID	Description
FRM-103	The system shall support all the input methods defined in FRD-117.
FRM-104	Each time a question is presented, the participant shall get an estimate on how many questions are left of the current survey.
FRM-105	The participant shall be presented with the number of seconds left to answer the currently displayed question. When there are no more seconds left, the question shall automatically disappear from the screen of the device and register that the question timed out.

Table A.14: Requirements: Answer Question

Use case M-3: Store Answer

Requirement ID	Description
FRM-106	Each answer given by a participant must be persistently stored by the system. This must be done for the system to be able to send results back to esmDesk when the survey is finished.
FRM-107	The participant should be able to select whether answers given in a survey shall be deleted or not when they are submitted to esmDesk.

Table A.15: Requirements: Store Answer

Use case M-4: Receive Data

Requirement ID	Description
FRM-108	The system shall be able to receive data sent by esmDesk. The data could be a new survey to participate in or messages to be displayed to the participant.
FRM-109	The system shall differentiate between a message meant for the participant, and the actual survey when it is distributed.

Table A.16: Requirements: Receive Data

Use case M-5: Send Data

Requirement ID	Description
FRM-110	The system shall be able to send data to esmDesk.
FRM-111	The system shall maintain the address of the esmDesk that created the survey a participant is currently taking part in.

Table A.17: Requirements: Send Data

Use case M-6: Installation

Requirement ID	Description
FRM-112	Any device used must be able to download the system from a location given by esmDesk. The location will typically be given through an SMS.

Table A.18: Requirements: Installation

Appendix B

esmDesk - Package Diagram

Due to the complexity and large number of classes in esmDesk we decided to illustrate the design by using UML package diagram. The package diagram organises classes in a system into groups, and shows the dependencies between them. This way it makes it easier to see the architecture of the software application.

Figure B.1 displays the package diagram for esmDesk. As mentioned in section 5.3.1, the architecture is split into three layers; *ui*, *logic* and *persistence* layer. The persistence layer is handled entirely by esmService, hence not depicted in the figure below. Everything that has to do with the user interface is handled in the ui packet or any of its children, whereas operations that is not directly related to the ui is handled in logic.

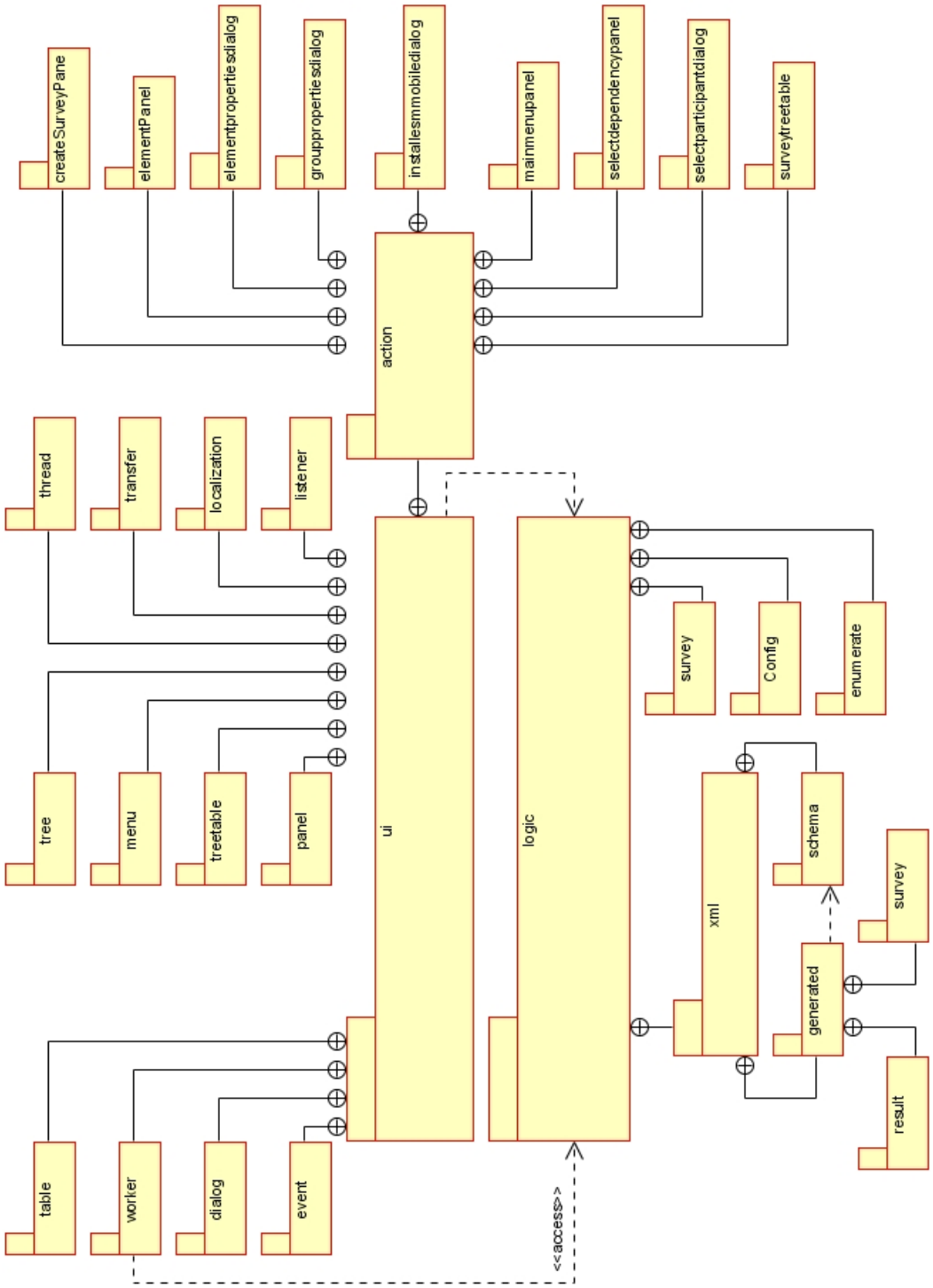


Figure B.1: esmDesk - Package Diagram

Appendix C

XML Schemas

This appendix includes a visual representation of the two XML-schemas that has been defined for the ESM software in this thesis. A survey expressed in XML is valid as long as it conforms to the schema in figure C.2, whereas answers in a survey is valid according to the schema in figure C.1.

Both schemas can be found in the CD-ROM following this document under the names *surveySchema.xsd* and *answerSchema.xsd*.

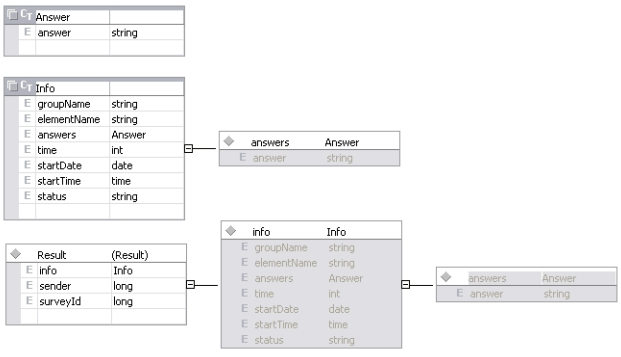


Figure C.1: XML Schema representing answers in a survey

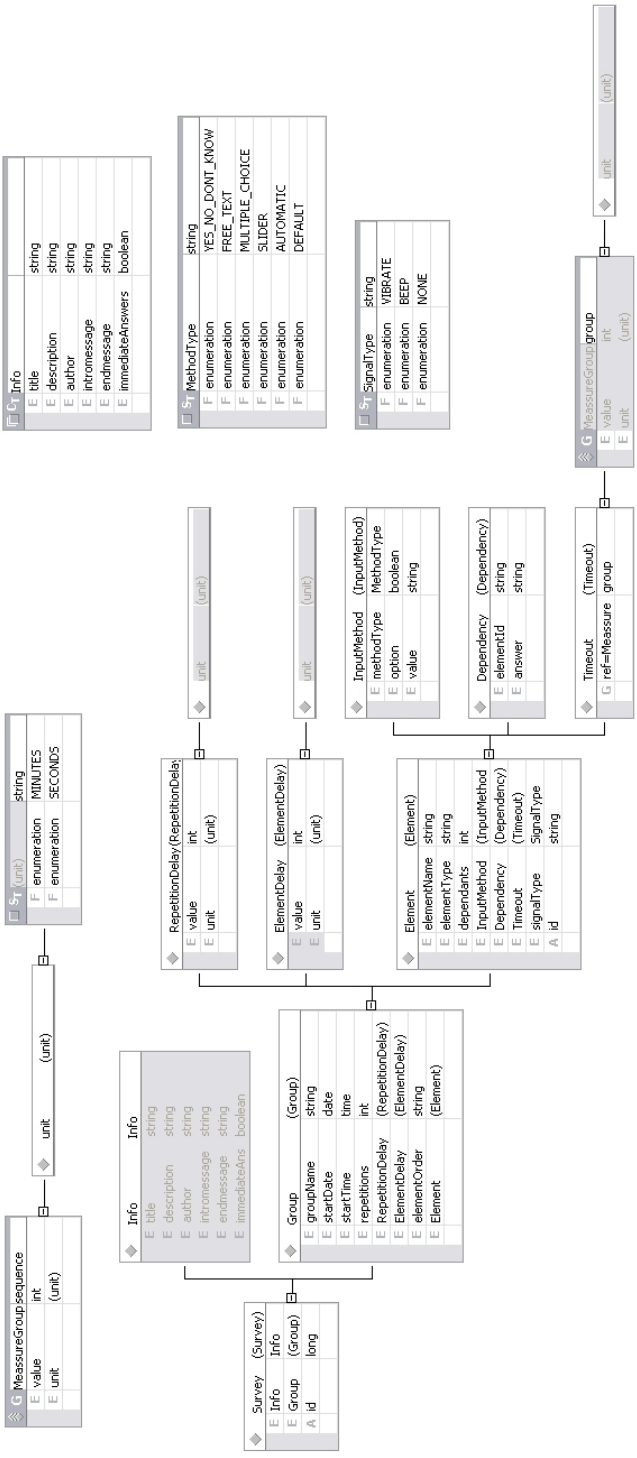


Figure C.2: XML Schema representing a survey

Appendix D

esmService - Diagrams

D.1 ER Diagram

Figure D.1 depicts the ER diagram for the persistent storage in esmDesk. All information about a single survey is stored in the entity *Survey*, including the XML representation of the survey. Each survey can have a number of participants, and a participant can be part of one or more surveys. This is achieved by a many-to-many relationship between these two entities. For every participant many results are received which can contain one or more answers. One-to-many relationships connect these entities.

D.2 Class Diagram

Figure D.2 displays the classes of esmService. Apart from DBComponent and its exception classes the rest are classes which maps to the different entities in figure D.1. Hibernate handles the mapping of these classes down to tables in the Derby database.

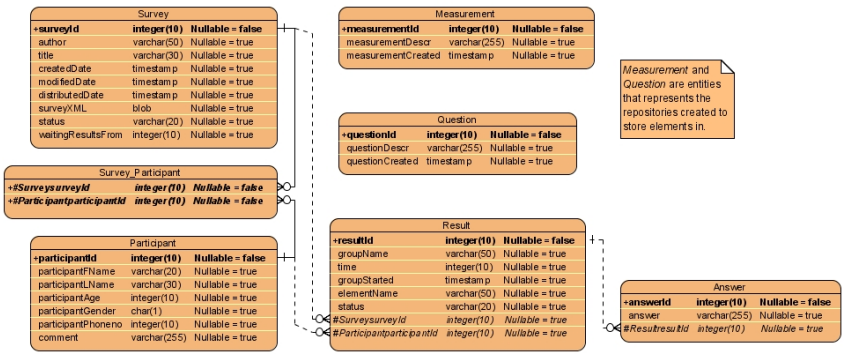


Figure D.1: esmService - ER Diagram

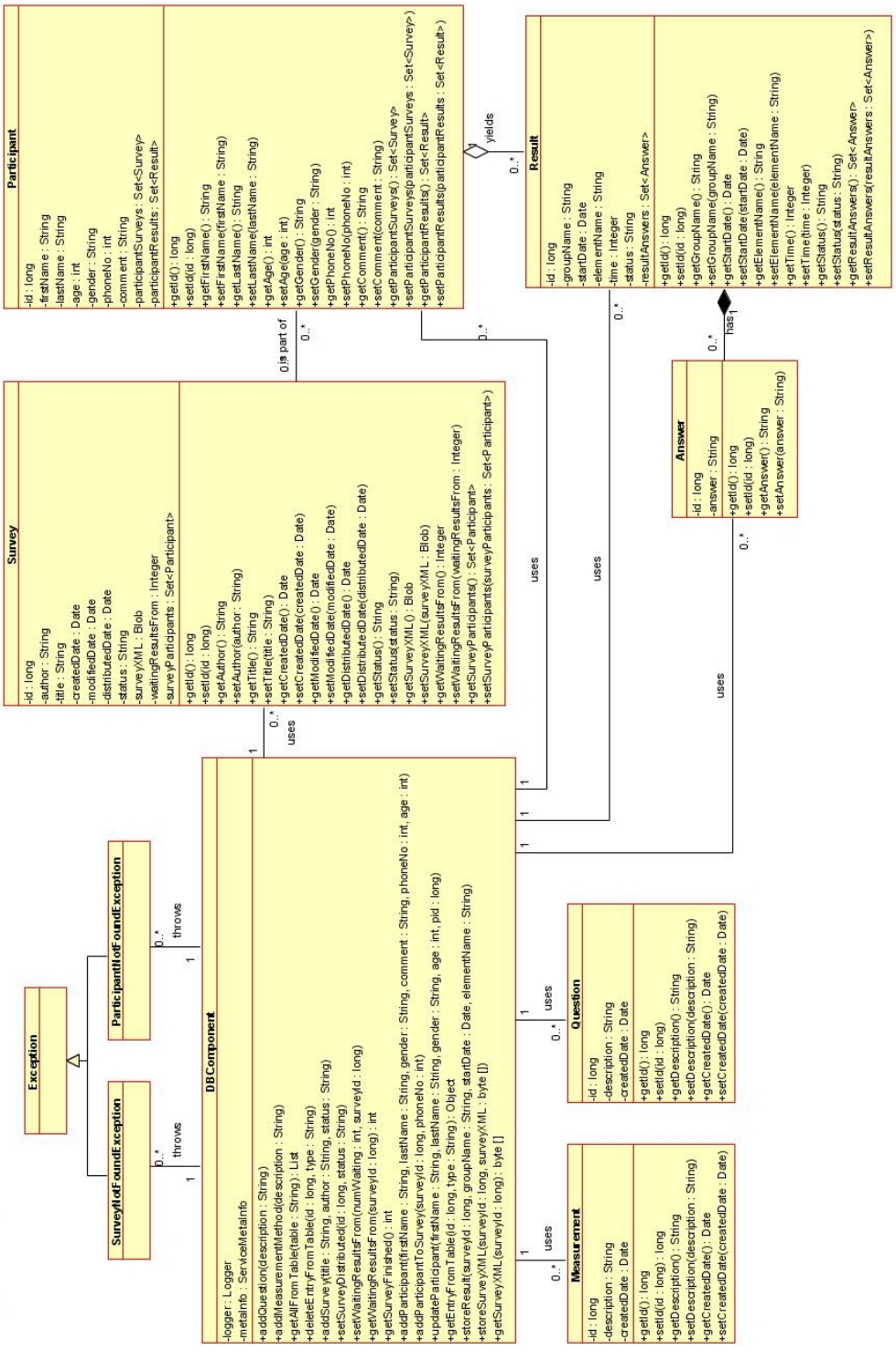


Figure D.2: esmService - Class Diagram

Appendix E

esmMobile - ER Diagram

Figure E.1 displays the ER diagram for the persistent storage of esmMobile. The design of this diagram reflects the design of the XML Schema that is defined in figure C.2. The reason for this is that when the mobile application receives a new survey, esmMobile immediately parses the XML file and stores it according to the ER diagram. Worth mentioning is that some columns of selected tables does not have the preferred data type, one example is the column *hasRun* in Group should be boolean type but is integer. This is not due to bad design, but limitations in the mobile database. The database used for this thesis is Microsoft Mobile SQL Server 2005 Mobile Edition[1].

[1] http://msdn2.microsoft.com/en-us/sql/aa336364.aspx

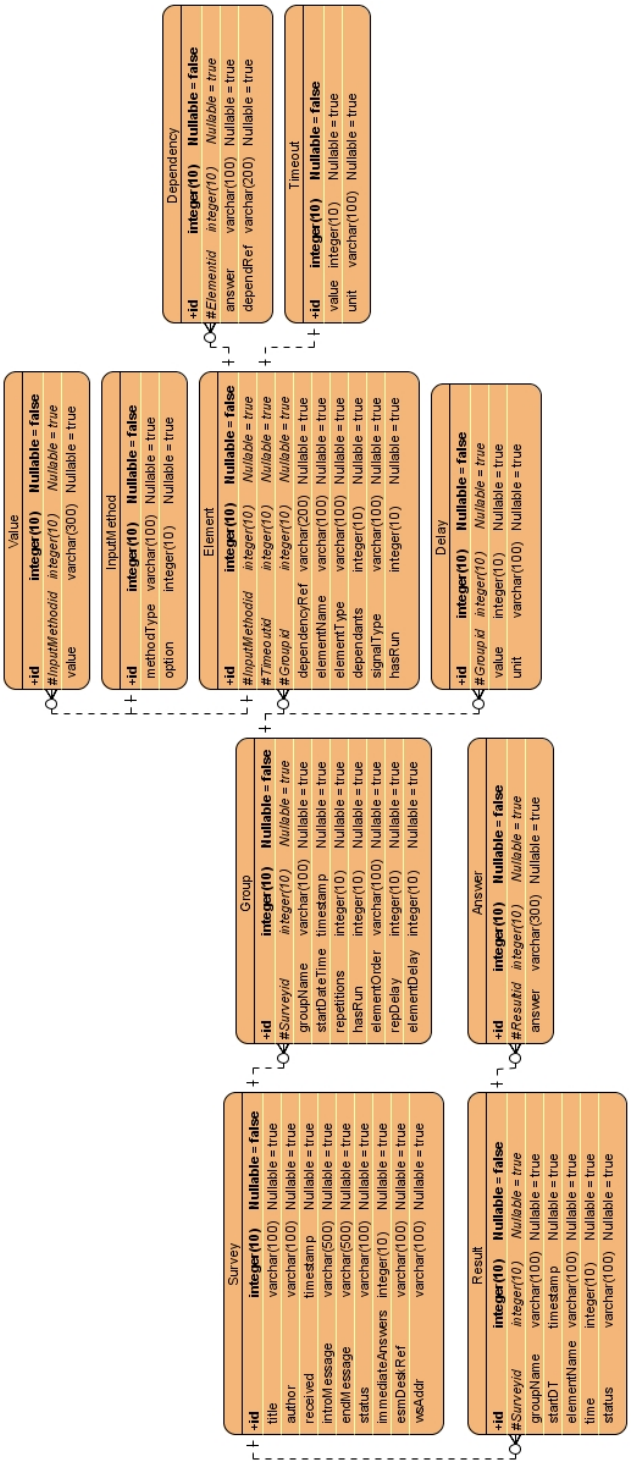


Figure E.1: esmMobile - ER diagram

Appendix F

Experiment

This appendix includes the XML file generated by esmDesk during the experimental survey done in the thesis. The XML file was interpreted and ran by a device of type HTC TyTn[1] with Microsoft Windows Mobile 5.0 Pocket PC[2] edition installed. To get syntax highlighting in this appendix the *listings* package of LaTeX has been used, which does not support special UTF-8 encoded characters. As a consequence the Norwegian characters æ, ø and å are not visible in the attached XML file. That said, the goal with this appendix is to show the structure of the generated XML file, and not its actual content.

[1] http://www.europe.htc.com/products/htctytn.html
[2] http://www.microsoft.com/windowsmobile/default.mspx

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Survey xmlns="http://ablab03.cs.uit.no/esmdesk" id="1">
  <Info>
    <title>Trivsel</title>
    <description>Opplevelser i fest og fritid</description>
    <author>Joar V</author>
    <intromessage>
      Vi ber deg lese sp rsm lene grundig. Svar s  godt du kan.
    </intromessage>
    <endmessage>
      Takk for at du har deltatt i denne unders kelsen -
      flaxlodd kommer i posten !
    </endmessage>
    <immediateAnswers>false</immediateAnswers>
  </Info>
  <Group>
    <groupName>Start</groupName>
    <startDate>2007-04-25</startDate>
    <startTime>09:27:20</startTime>
    <repetitions>1</repetitions>
    <RepetitionDelay>
      <value>10</value>
      <unit>SECONDS</unit>
    </RepetitionDelay>
    <ElementDelay>
      <value>10</value>
      <unit>SECONDS</unit>
    </ElementDelay>
    <elementOrder>SEQUENTIAL</elementOrder>
    <Element id="77503d8a-4e06-4a74-80c1-3d65d1b7401b">
      <elementName>
        Jeg samtykker i at jeg vil v re med i
        denn unders kelsen
      </elementName>
      <elementType>QUESTION</elementType>
      <dependants>8</dependants>
      <instruction></instruction>
      <InputMethod>
        <methodType>MULTIPLE_CHOICE</methodType>
        <option>false</option>
        <value>ja</value>
        <value>nei</value>
      </InputMethod>
      <Timeout>
        <value>1</value>
        <unit>MINUTES</unit>
      </Timeout>
      <signalType>BEEP</signalType>
    </Element>
  </Group>
  <Group>
    <groupName>Swels</groupName>
    <startDate>2007-04-25</startDate>
```



```
    <startTime>09:34:30</startTime>
    <repetitions>1</repetitions>
    <RepetitionDelay>
      <value>10</value>
      <unit>SECONDS</unit>
    </RepetitionDelay>
    <ElementDelay>
      <value>10</value>
      <unit>SECONDS</unit>
    </ElementDelay>
    <elementOrder>SEQUENTIAL</elementOrder>
    <Element id="3449f2f3-4f3d-4eed-becb-b0e2ce242e54">
      <elementName>
        P  de fleste m ter er livet mitt n r idealet mitt
      </elementName>
      <elementType>QUESTION</elementType>
      <dependants>0</dependants>
      <instruction></instruction>
      <InputMethod>
        <methodType>MULTIPLE_CHOICE</methodType>
        <option>false</option>
        <value>1 Helt uenig</value>
        <value>2 </value>
        <value>3</value>
        <value>4</value>
        <value>5</value>
        <value>6</value>
        <value>7 Helt enig</value>
      </InputMethod>
      <Dependency>
        <elementId>
          77503d8a-4e06-4a74-80c1-3d65d1b7401b
        </elementId>
        <answer>ja</answer>
      </Dependency>
      <Timeout>
        <value>1</value>
        <unit>MINUTES</unit>
      </Timeout>
      <signalType>NONE</signalType>
    </Element>
    <Element id="2444cd25-6e0d-4020-92dc-67045e5d894e">
      <elementName>
        Hvis jeg kunne leve livet p  nytt,
        ville jeg nesten ikke forandret p  noe.
      </elementName>
      <elementType>QUESTION</elementType>
      <dependants>0</dependants>
      <instruction></instruction>
      <InputMethod>
        <methodType>SLIDER</methodType>
        <value>1(helt uenig)</value>
        <value>helt enig 7</value>
      </InputMethod>
```

```
      <Dependency>
        <elementId>77503d8a-4e06-4a74-80c1-3d65d1b7401b</elementId>
        <answer>ja</answer>
      </Dependency>
      <Timeout>
        <value>1</value>
        <unit>MINUTES</unit>
      </Timeout>
      <signalType>NONE</signalType>
    </Element>
  </Group>
  <Group>
    <groupName>Bet</groupName>
    <startDate>2007-04-25</startDate>
    <startTime>09:42:58</startTime>
    <repetitions>1</repetitions>
    <RepetitionDelay>
      <value>10</value>
      <unit>SECONDS</unit>
    </RepetitionDelay>
    <ElementDelay>
      <value>10</value>
      <unit>SECONDS</unit>
    </ElementDelay>
    <elementOrder>SEQUENTIAL</elementOrder>
    <Element id="92efbbe1-de77-4d89-a293-e6aeb6990aba">
      <elementName>Glad</elementName>
      <elementType>QUESTION</elementType>
      <dependants>0</dependants>
      <instruction>
        S nn til vanlig, hvor ofte opplever du f lgende f lelser:
      </instruction>
      <InputMethod>
        <methodType>MULTIPLE_CHOICE</methodType>
        <option>false</option>
        <value>1 Aldri</value>
        <value>2</value>
        <value>3</value>
        <value>4</value>
        <value>5</value>
        <value>6</value>
        <value>7 Alltid</value>
      </InputMethod>
      <Dependency>
        <elementId>77503d8a-4e06-4a74-80c1-3d65d1b7401b</elementId>
        <answer>ja</answer>
      </Dependency>
      <Timeout>
        <value>20</value>
        <unit>SECONDS</unit>
      </Timeout>
      <signalType>BEEP</signalType>
    </Element>
    <Element id="918019ce-047f-41d2-af67-bd16aa4740f7">
```



```
      <elementName>Redd</elementName>
      <elementType>QUESTION</elementType>
      <dependants>0</dependants>
      <instruction></instruction>
      <InputMethod>
        <methodType>MULTIPLE_CHOICE</methodType>
        <option>false</option>
        <value>1 Aldri</value>
        <value>2</value>
        <value>3</value>
        <value>4</value>
        <value>5</value>
        <value>6</value>
        <value>7 Alltid</value>
      </InputMethod>
      <Dependency>
        <elementId>77503d8a-4e06-4a74-80c1-3d65d1b7401b</elementId>
        <answer>ja</answer>
      </Dependency>
      <Timeout>
        <value>20</value>
        <unit>SECONDS</unit>
      </Timeout>
      <signalType>BEEP</signalType>
    </Element>
  </Group>
  <Group>
    <groupName>ESM1</groupName>
    <startDate>2007-04-25</startDate>
    <startTime>09:54:30</startTime>
    <repetitions>1</repetitions>
    <RepetitionDelay>
      <value>5</value>
      <unit>MINUTES</unit>
    </RepetitionDelay>
    <ElementDelay>
      <value>10</value>
      <unit>SECONDS</unit>
    </ElementDelay>
    <elementOrder>SEQUENTIAL</elementOrder>
    <Element id="5173afc9-cd54-4466-beec-916b4f84d6fc">
      <elementName>Hva gj r du akkuart n ?</elementName>
      <elementType>QUESTION</elementType>
      <dependants>0</dependants>
      <instruction></instruction>
      <InputMethod>
        <methodType>FREE_TEXT</methodType>
        <option>true</option>
      </InputMethod>
      <Dependency>
        <elementId>77503d8a-4e06-4a74-80c1-3d65d1b7401b</elementId>
        <answer>ja</answer>
      </Dependency>
      <Timeout>
```

```
      <value>5</value>
      <unit>MINUTES</unit>
    </Timeout>
    <signalType>BEEP</signalType>
  </Element>
  <Element id="93ff6d37-931a-486c-9e2c-25380c01fbd3">
    <elementName>
      Hvordan er dine evner i forhold til den situasjonen
      du er i akkurat n ?
    </elementName>
    <elementType>QUESTION</elementType>
    <dependants>0</dependants>
    <instruction></instruction>
    <InputMethod>
      <methodType>MULTIPLE_CHOICE</methodType>
      <option>false</option>
      <value>1 For d rlige</value>
      <value>2</value>
      <value>3</value>
      <value>4</value>
      <value>5 Mer enn bra nok</value>
    </InputMethod>
    <Dependency>
      <elementId>
        77503d8a-4e06-4a74-80c1-3d65d1b7401b
      </elementId>
      <answer>ja</answer>
    </Dependency>
    <Timeout>
      <value>20</value>
      <unit>SECONDS</unit>
    </Timeout>
    <signalType>BEEP</signalType>
  </Element>
  <Element id="d51b537e-b999-495d-a3c2-4cbe34250351">
    <elementName>Tilfreds</elementName>
    <elementType>QUESTION</elementType>
    <dependants>0</dependants>
    <instruction>Hvordan f l e r du deg akkurat n ?</instruction>
    <InputMethod>
      <methodType>SLIDER</methodType>
      <value>1 Slett ikke</value>
      <value>Absolutt 5</value>
    </InputMethod>
    <Dependency>
      <elementId>
        77503d8a-4e06-4a74-80c1-3d65d1b7401b
      </elementId>
      <answer>ja</answer>
    </Dependency>
    <Timeout>
      <value>20</value>
      <unit>SECONDS</unit>
    </Timeout>
```



```
    <signalType>BEEP</signalType>
  </Element>
 </Group>
</Survey>
```

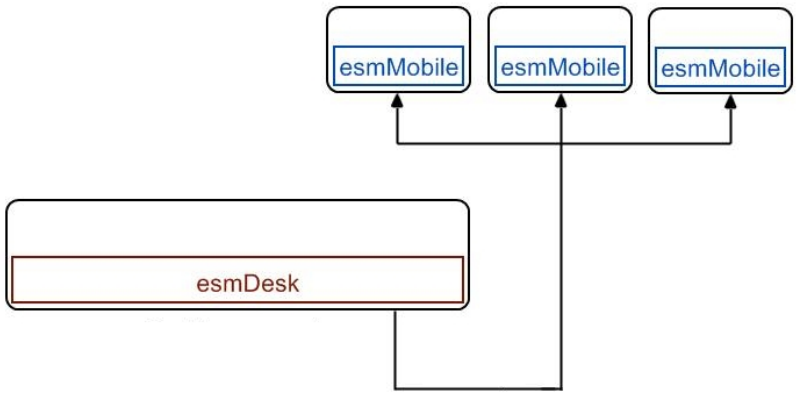
Appendix G

User Guide

This appendix includes a simple user guide written for esmDesk and esmMobile.

esm*Desk* & esm*Mobile*

User Guide



30.04.2007

Espen Ekvang

Preface

The user guide for esmDesk and esmMobile tells you how to benefit from the two systems in creation and realisation of any survey following the Experience Sampling Method (ESM). Both software systems have been developed as part of a master's thesis written by Espen Ekvang spring term 2007.

Contents

Chapter 1

Getting Started

Together esmDesk and esmMobile provide you as an experimenter with a tool that enables you to:

- Create a survey based on the ESM approach.
- Distribute the survey out to a set of participants running esmMobile.
- View the results from surveys created and distributed using esmDesk.

1.1 Prerequisites

Before setting up esmDesk and esmMobile make sure that the computer to run esmDesk has the following installed:

- Java 2 Standard Edition version 6 or later, obtainable from [1].

The device to run created surveys must have:

- Windows Mobile 5.0 Pocket PC operating system
- .NET Compact Framework version 2.0 or later
- Microsoft SQL Server 2005 Mobile edition

All prerequisites for the mobile device apart from the operating system itself are included on the CD-ROM following this document. The prerequisites are contained on the CD-ROM as cabinet (CAB) files, meaning that they will automatically install themselves on any mobile device as long as the file is transferred to it and started. To transfer a CAB file to a mobile device, connect the device to a computer with Active Sync[2] installed. Once the device is connected right-click on the Active Sync icon in the system tray of your Windows operating system and click *explore*. A window opens and a set of files are displayed. These files are the items on the mobile device. Now open the folder on your computer containing the CAB file you want to install. Next, drag the CAB file from the computer onto any folder on the mobile device. Ensure that the transfer is successful. The last step is to locate the file on the mobile device using the device. Once located, click on it using the stylus or keys on the mobile device, and you will see that it starts to install the application contained within the CAB file.

[1] http://java.sun.com/javase/downloads/index.jsp
[2] http://www.microsoft.com/windowsmobile/activesync/activesync45.mspx

Chapter 2

Using esmDesk

The CD-ROM following this document contains a zip filed called esmDesk_bin.zip which includes the application and the platform on which it is to run (Argos). Save and extract the zip file to a directory of your preference. After the file has been extracted you get a folder named Argos. Using the command prompt of Windows XP (start -> run -> cmd), navigate into the Argos folder and type the following:

java -jar Argos.jar

The output in the command prompt should be something like the screen dump from figure 2.1. When starting the application it will install itself in the system tray of your operating system as depicted with the red ellipse in figure 2.1. Clicking this icon maximises or minimises the application dependent of its current state.

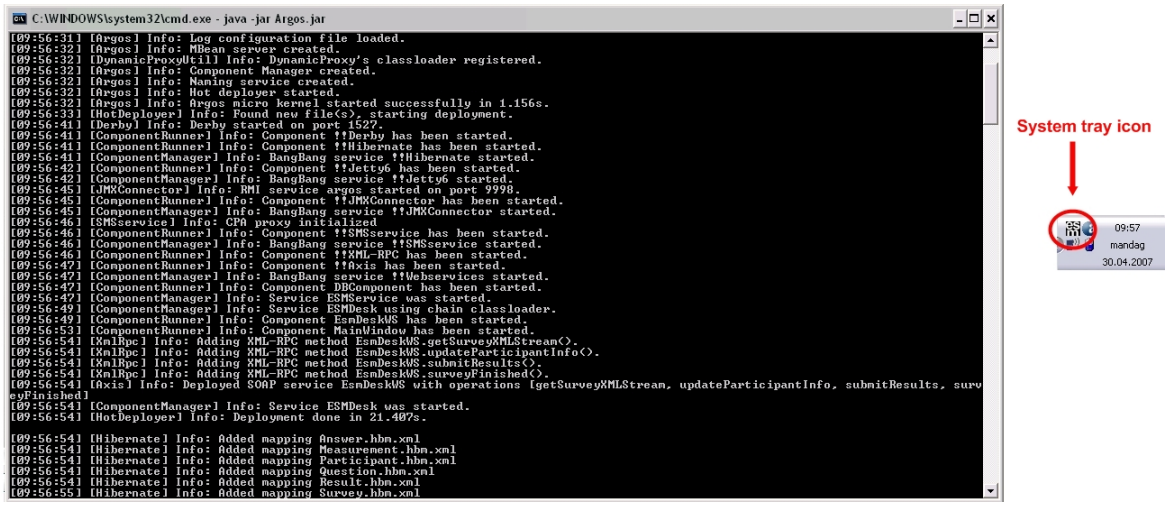


Figure 2.1: Starting up Argos and esmDesk

2.1 Main Menu

The first time esmDesk starts up it will require you to enter the IP address of the computer running Argos along with the port number on which Axis is running in Argos, usually port 8080. Later, if the IP address or port number changes, use the *Settings* button of the main menu.

After specifying the configuration, the main menu of esmDesk is loaded as shown in figure 2.2. A total of six buttons constitutes the functionality of esmDesk. Two buttons are disabled in the figure, which will be the case first time the application starts up. Details on how these buttons are enabled follows later on in this document. This section covers the different parts of the menu. To be able to distribute a survey esmMobile needs to be installed on a set of device, therefore the installation of esmMobile is first mentioned here.

Figure 2.2: Main menu esmDesk

2.1.1 Install esmMobile

esmDesk provides a simple feature for installing esmMobile on a set of devices, by clicking the *Install esmMobile* button of the main menu in esmDesk a dialog appears as in figure 2.3. Input the phone numbers of all devices that are to install esmMobile. After specifying all phone numbers click the *Install* button.

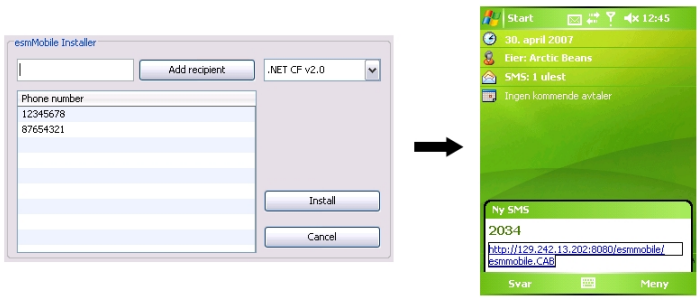


Figure 2.3: Install esmMobile

Installation is basically a two-step procedure. The first step is done in esmDesk and causes the system to send an SMS to the mobile device. In figure 2.3 a mobile device is shown to the right, and you can see that the SMS message have been received. Click on the link in the message and follow the instructions to install esmMobile. Once the installation is completed, it is important to click on Start -> Programs on the mobile device. In the list of Programs there shall be an icon with

the text *esmMobile* underneath. Click this icon to start the mobile application. Now your are ready to create and distribute surveys using esmDesk.

2.1.2 Create Survey

From the main menu of esmDesk click the button "Create new survey" to start creating your first survey. Before starting to create the survey a small dialog appears where you should fill in some information about the title of the survey, a small description of the survey and finally your name. Once you have done that, the system displays the survey creator to you.

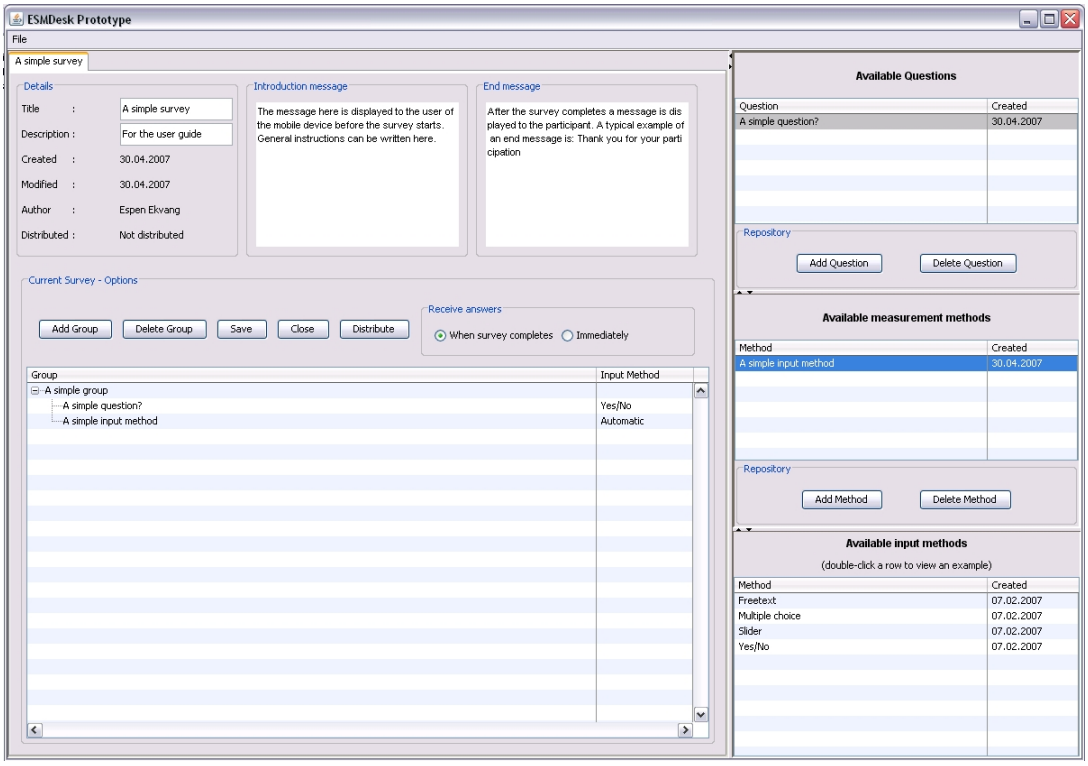


Figure 2.4: Survey Creator

The survey creator contains all pieces of information related to the survey you are going to create. At the top left corner there is a grouping named *Details* containing some characteristics of the survey. That is its title, author, when it was created and whether it has been distributed or not. If the survey has been distributed, the date of distribution is listed under the details section. The area to the right of the details panel is meant for an introductory message displayed to the participant on survey start-up. The end message field is equivalent to the introduction field, only difference is that it gets displayed at the end of the survey.

The right part of the application shown in figure 2.4 displays three tables. Each table represents a persistent repository where different elements can be stored. The top most repository is for questions used in any survey, the middle table is for measurement methods such as counting steps, measure blood sugar etc. The bottom table is an attempt to provide examples on how the different input

methods could look on a mobile device running a survey. With the two tables at the top you as an experimenter can add and remove elements from the repositories. To add an element click the add-button below the preferred table, enter information about the element and click ok as shown in figure 2.5.

Figure 2.5: Add a new question to the repository

To remove an element, first click on the element you want to remove, then select the remove bottom from the preferred table. To view the input method examples double-click on an item in the input method table and one of the dialogs displayed in figure 2.6 will be shown to you.

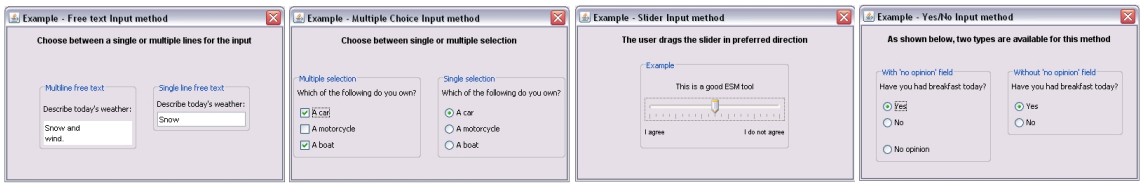


Figure 2.6: Input methods - examples

Create a Group

Any surveys created using esmDesk is compound of a set of elements and groups. A group is a collection of elements, where they share a set of preferences for that group. Before you can add elements to a survey, a group is needed. The large table with the columns *Group* and *Input Method* (see figure 2.4) is the visual representation of all the groups an elements in the survey. There are two possible approaches to creating a new group:

- Right-click in the area of the table representing the survey and click *Add Group*.
- Click the button *Add Group* from the buttons above the survey table.

Both approaches causes the system to display a dialog where you must specify the preferences for the new group. The dialog enables you to set the following preferences for the new group (see figure 2.7):

- A name for the new group.
- The start time for the first element of this new group.
- How many times the elements within this new group shall be repeated.
- The delay between each repetition of the group
- The delay between each element inside this group
- The execution order of the elements can be either sequential or randomised.

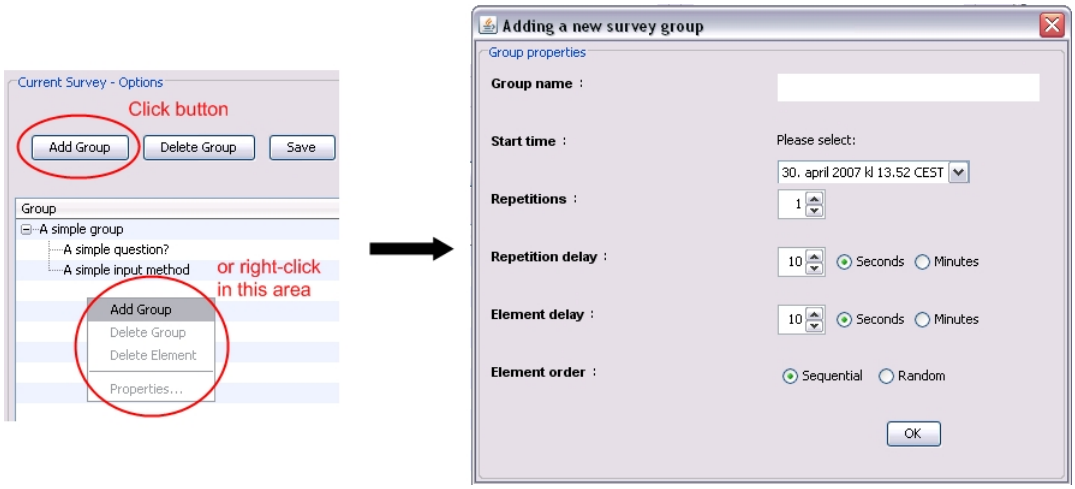


Figure 2.7: Adding a group

The properties of the group can be changed any time during the creation of the survey by first clicking on the particular group in the survey table, followed by a right-click to get the context menu. From that menu you can select *Properties* to view and/or edit the details for the selected group. The context menu also provides an option for deleting groups and elements. Note that if you select to delete a group, all its elements will be deleted as well.

Add an Element to a Survey

Once a group has been created, elements can be added to it. Adding an element to a survey is done using the repositories to the right in the survey creator. The application has a drag-and-drop feature that lets you drag elements from the different repositories onto a group in the table representing the survey. To add a question to the survey, you first click on the element you want to add in the repository. If the element you want is not in the repository you have to add to the repository before adding it to the survey. Once you have clicked on your element, drag it from the repository and onto the group in which you want it to be in the survey table, an illustration of this is provided in figure 2.8.

Dropping an element in a group causes a dialog to open (see figure 2.8). In this dialog all the preferences for the particular element must be specified. The different settings are split into *Input Method*, *Dependencies*, *Timeout*, *Instruction* and *Signal*. The first of these, the input method, specifies how the participant should answer the new question to be added to the group. A combo box gives you the opportunity to view examples of the different input methods until you have decided the best suited for your question. After a decision has been made on the input method, click the OK button to the right of the combo box. Dependent on what input method you have selected new properties will appear. The four input methods available are:

- **Free text**, let the participant answer using his/hers own words. By clicking the check-box *Multiline* for this method the participant gets more than one line to write their answer.
- **Multiple choice**, the participant is able to select among a set of available options. The options are specified by you using the *Add answer* button appearing when this method is

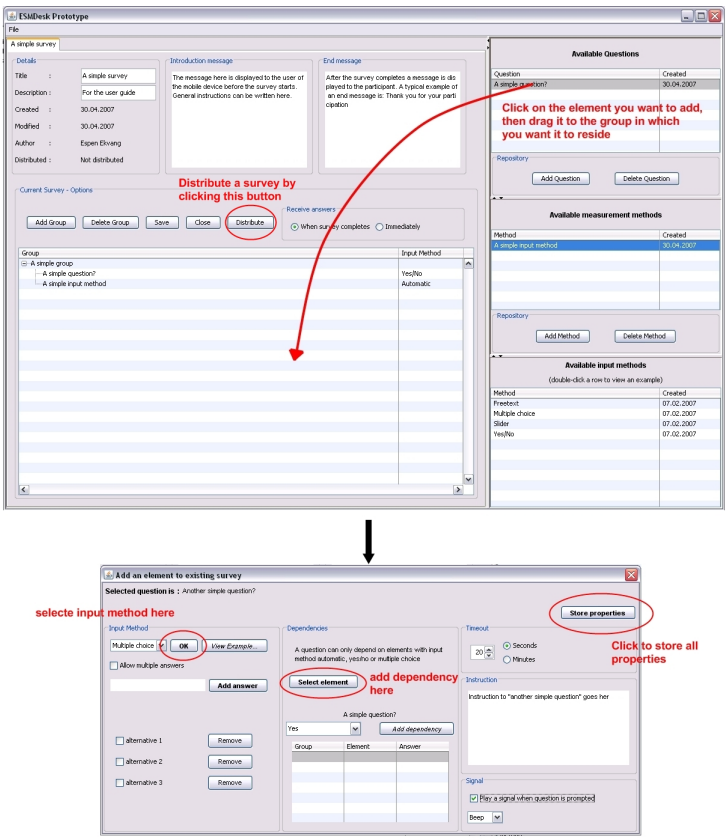


Figure 2.8: Add an element to a group

selected. If the check-box *Allow multiple answers* is selected the participant can select more than one answer. The figure 2.8 includes an example where multiple choice has been selected.

- **Slider**, specify the left and right extremities using the *left* and *right* buttons appearing when this method is selected.

- **Yes/No**, the participant is allowed to answer yes or no to an answer. If you check the option *Add no opinion field* the participant will also be able to answer no opinion.

Setting the time out for a new question means specifying how many seconds the participant is given to answer the question. If the participant does not answer within the time out specified here, esmMobile registers that the user never answered the question. Keep in mind that it cannot be assumed that the participant keeps the mobile device in his/hers hand all the time. Allow for the device to be in a pocket or so by setting the time out to a reasonable amount of time. Along with specifying a time out you can also set whether the device should play a signal or vibrate as the question is prompted, using this option can help to get the attention of the participant.

Because the mobile device very likely has a small screen we encourage you to create your questions as small as possible. In situations where you need to provide detailed information to a participant

before a particular question is asked, please use the instruction field of the question properties. Here you can write everything the participant must be aware of before answering the question.

The last option for a new question has to do with dependencies. A dependency is a condition that must be fulfilled in order for another question to be run. A simple example illustrates this. Say that you have a question A, "Do you have a car?" and you want the question B, "What type?" only to be asked if the participant answers yes on the first question. With dependencies this is possible. Figure 2.8 shows what button to click to add a dependency. Clicking that button displays another dialog containing a list with elements scheduled to run before this one. Click the element you want to depend on from this list and click OK. Now you will see something like what is displayed in figure 2.8 for the dependency section. The available options for the selected dependency appear in a new combo-box. Select the answer your require for the dependency element and add the dependency by clicking "Add dependency". To remove a dependency, right-click in the list of dependencies, a context menu appears where you can select "Remove dependency".

After specifying all your properties for the new element, click *Store properties* and the element is added to the position in the table you dropped it into. NOTE: If you drop the element at the bottom area as illustrated in the figure, the element will be inserted at the end position in the last group. The information provided in this section is basically what you need to create a full survey. If you want to save and distribute the survey on a later occasion, click the close button and the survey is saved and you can open it at a later point in time. The properties of an element can be changed the same way as properties for a group, it is also possible to delete an element within a particular group.

2.1.3 Open Survey

Opening an existing survey is an option from the main menu of esmDesk and is only enabled if there are any surveys in the system to be opened. By clicking the *Open Survey* button a dialog appears with a list of all available surveys stored with esmDesk. Select one of the surveys listen and click *Open...* Now the survey creator will be displayed as when you create a new survey, the only difference is that it will contain all the information you stored from the last time you opened the survey.

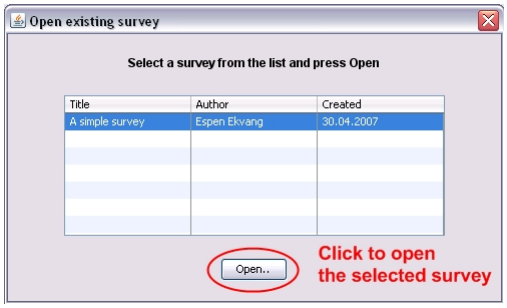


Figure 2.9: Open an existing survey

2.1.4 Distribute a Survey

Distributing a survey involves notifying all participants that a new survey is available. From the survey creator there is a button called *Distribute* which you click when you consider yourself done with the creation of a survey, and your are ready to distribute it out to the set of devices (see figure 2.8). Note that since the tool is based on Experience Sampling Method there is no need for you to wait until the start date of the first element, esmMobile is responsible for starting the element on the correct time. Therefore when you are ready click the *Distribute* button and select the participants for the survey you have created.

Figure 2.10 shows the steps involved in the selection of participants for a survey. When you have clicked the *Distribute* button a dialog representing the participant repository is displayed. From this repository you can choose to add/remove participants from the repository and select the participants which are to be part of the survey you want to distribute. To add a participant to the particular survey make sure that the check box to the left of the name is checked as the bottom image of figure 2.10 depicts. After esmDesk has distributed the new survey out to the set of selected participants, the main menu of the application is displayed.

2.1.5 Get Results

Results from a survey are automatically transferred back to esmDesk and when results are back from a survey the button *Get Results* from the main menu will be enabled. Once this button is enabled you can choose to dump these results to a file. When clicking the *Get Results* button a dialog similar to the one displayed in figure 2.9 is displayed, instead of opening a survey you select the survey from which you want the results. After selecting the survey you have to specify a location and name for the text file that should contain the results from the survey.

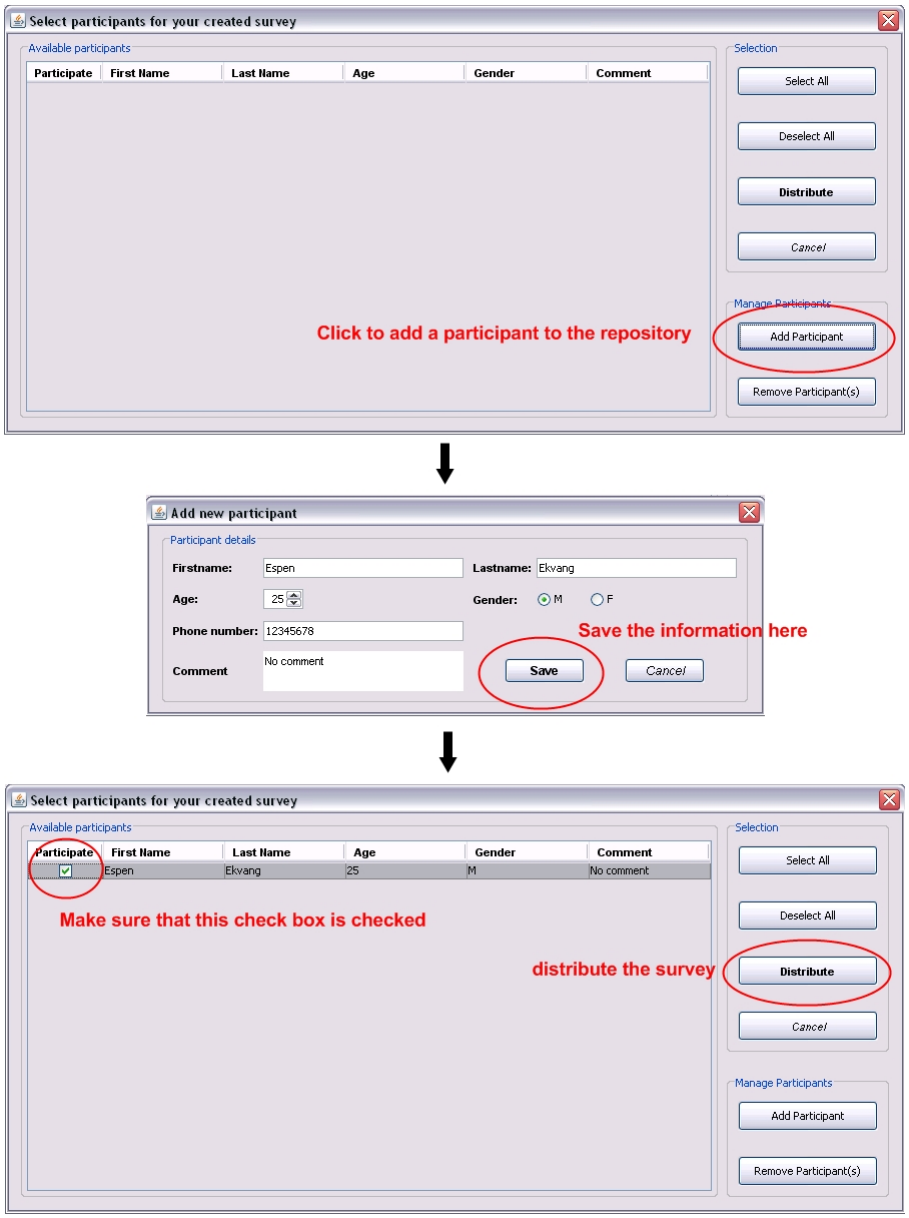


Figure 2.10: Distribute a survey

Chapter 3

Using esmMobile

esmMobile is the mobile application actually running surveys created using esmDesk. As mention in section 2.1.1 esmDesk includes and option for remotely installing the mobile application. After installing the application you start the application by clicking its icon in the list of programs on the mobile device.

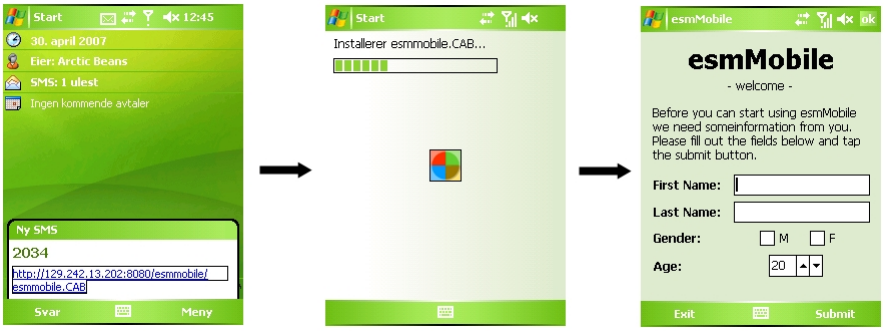


Figure 3.1: Starting up esmMobile

Figure 3.1 shows the sequence of screen dumps for the mobile device from the receipt of an SMS sent by esmDesk until the application is installed and started. The start-up screen shown in the right most part of the figure only appears the first time esmMobile starts. It is meant for the participant, alternatively the experimenter can fill out the form before handing the device out to the participant. From that point on the application will always run in the background. If the device is turned off and on again, the application will automatically starts itself. The rest of the operations for esmMobile are done automatically when messages and surveys are received.

The different input methods available from esmDesk are illustrated through the screen dumps from a mobile device in figure 3.2.

Figure 3.2: Input methods available

Appendix H

CD-ROM

The CD-ROM following this thesis contains the source code for the implemented applications. The content of the CD-ROM is organised into three main folders, *src*, *bin* and *doc*. Please refer to the User Guide for the prerequisites for esmDesk and esmMobile.

The *src* folder contains the source files for the different applications within this thesis. It is recommended to use the folder structure as given in the *src* directory for further development, mainly because the ANT build files for the projects assume this structure. For further development of esmDesk, esmService and !!SMSservice using the editor Eclipse[1] is preferable. The mobile development is done using Microsoft Visual Studio 2005[2] and it is also assumed that the computer to be used for the development of esmMobile has Mobile Client Software Factory (MCSF) from Microsoft Patterns & Practices[3] installed and set up correctly.

Using the prototype as-is, is possible using the files in the *bin* folder. The CAB file can be transferred to any mobile device running Microsoft Windows Mobile 5.0 using for example ActiveSync. *esmDesk_bin.zip* contains the Argos middleware platform with all services needed to run esmDesk deployed in it.

The pdf version of this document is included in the *doc* folder.

[1] http://www.eclipse.org
[2] http://msdn2.microsoft.com/en-us/vstudio/aa718668.aspx
[3] http://msdn2.microsoft.com/en-us/downloads/default.aspx