NTNU
Norwegian University of
Science and Technology

InstantSocial

social networking in mobile ad-hoc environments

Henrik Halvorsen

Master of Science in Computer Science

Submission date: July 2009

Supervisor: John Krogstie, IDI
Co-supervisor: Svein Hallsteinsen, SINTEF
Jacqueline Floch, SINTEF
Jiang Shanshan, SINTEF

Norwegian University of Science and Technology
Department of Computer and Information Science

Problem Description

With InstantSocial, we intend to investigate how new capabilities of mobile devices can be used to transpose Internet social networking trends and principles to a mobile ad hoc environment. In the context of InstantSocial, the European project MUSIC will be used to adapt / re-adapt an ad hoc server infrastructure in response to context changes by coordinating the different available devices and their resources. The main objective of this assignment is to develop a prototype of the InstantSocial application in order to assess the realism of this application idea and the suitability of the MUSIC platform in this context. InstantSocial is representative of a class of similar applications related to ad hoc social networking in mobile ad hoc environments, f. ex. multi user games or chat rooms, which need many similar mechanisms. As far as possible the design of the application prototype should seek to separate such more general parts from the parts specific to InstantSocial, with the aim to lay the foundation for an application framework for this class of applications.

Assignment given: 15. January 2009
Supervisor: John Krogstie, IDI

Abstract

This report covers the research, design and prototype implementation of a social application for mobile ad-hoc networks, InstantSocial. The main goals of this project has been to look at this exciting field and examine how the European scientific collaborative project MUSIC can be used to develop such an application.

The project has been conducted using a Design Science approach. First the field of interest, existing similar applications and technology was examined to get a good view of the current state-of-the-art and choose a suitable prototype application. Then the design of the application was constructed, emphasizing the important features outlined in the project goals. Finally a prototype was developed and tested as a means to prove the correctness and applicability of the design.

The results of the project was somewhat divided. Although the developed prototype had limited functionality, mostly because of the current limitations of the used framework, the tests that were performed where positive. Not all of the requirements of the system was successfully implemented, but this was due to limited time available or limitations in the currently available version of the framework rather than a shortage of the design.

We conclude that the presented design and approach to context-adaptation is very plausible as a way in which the MUSIC project can be used to develop social application for mobile ad-hoc networks, but that much more work and testing is required before such an application can be fully realized.

Preface

This report was written as a master thesis project at the Norwegian University of Science and Technology (NTNU) in a collaboration with SINTEF. The report was written in its entirety by the master graduate participating in the project - Henrik Halvorsen.

First and foremost I would like to thank my teaching supervisor at the university, John Krogstie, and the supervisors at SINTEF Trondheim Svein Hallsteinsen, Jacqueline Floch and Shanshan Jiang. Further I would like to thank for all the help from the various people that have lent a hand throughout the difficult parts of the project - Erlend Stav, Nearchos Paspallis, Mohammad Ullah Khan and Jorge Lorenzo Gallardo. I would also like to thank the writers whose works are referenced in this report for their important knowledge and work in this field.

Henrik Halvorsen

Oslo, July 15, 2009

Contents

List of Figures

List of Tables

Chapter 1

Introduction

> Do you suppose I could buy back my introduction to you?
>
> Groucho Marx

This chapter contains the introductory parts of the report. First a short general introduction to the project is presented. Then the research questions of the project is discussed, followed by a presentation of the contribution of this project to the scientific field and a short readers guide.

1.1 Project Introduction

Modern devices have evolved beyond the stationary computers of the past, and is rapidly moving to ever smaller, more mobile and interconnected devices. As mobility is ever-increasing, new interesting forms of communication are emerging. A very interesting such new communication form is the mobile peer-to-peer ad-hoc network, in which connections between devices are dynamically formed "on the go", and no central control is necessary. However, the hardware resources of the devices must often be sacrificed in the struggle to become ever smaller and more mobile. More on this can be found in section 2.1.

Another interesting effect of the new mobile forms of computing is the increased emphasis on the context of the devices, users and applications, in particular the variations in context caused by mobility. As users take their devices and applications with them as they move about, the context in which the applications are run varies than ever. As an example, varying degrees of illumination, network capacity and even the presence of other users can, and should, change the way in which the applications behave. This is explained in greater detail in section 2.2.

Developing software for such environments poses new challenges, and new approaches and supporting technologies are appearing with explicit support for the implementation of context awareness and adaptation. One interesting example is the MUSIC framework, being developed by the EU funded MUSIC project. This is introduced in section 2.6.

At the same time new interesting forms of applications are emerging, evolving beyond the old patterns of applications of the past. A very interesting and ever more popular such form of application is the social application. The definition of social application is somewhat vague, but the main point of interest is that these applications rely on and allow the users to provide content in a way not seen before. As might be expected, this allows for opportunities and ways of collaboration that has not been possible in earlier, more static forms of applications. More on this is found in section 2.4.

The goal of this project is to combine the interesting fields of mobile devices, context-adaptation and social applications and propose a prototype application which combines these elements. The focus will be on investigation the applicability of the MUSIC framework to social applications for mobile ad-hoc environments and to identify architecture and design elements which could be reusable in similar applications.

1.2 Problem Description

This section contains the project description of the project as indicated by the master thesis contract, as well as the choices made in the interpretation and approach to this.

1.2.1 Project description

This is the project description as agreed on between the supervisor and student of this thesis and presented in the master thesis contract:

With InstantSocial, we intend to investigate how new capabilities of mobile devices can be used to transpose Internet social networking trends and principles to a mobile ad hoc environment. In the context of InstantSocial, the European project MUSIC will be used to adapt / re-adapt an ad hoc server infrastructure in response to context changes by coordinating the different available devices and their resources. The main objective of this assignment is to develop a prototype of the InstantSocial application in order to assess the realism of this application idea and the suitability of the MUSIC platform in this context. InstantSocial is representative of a class of similar applications related to ad hoc social networking in mobile ad hoc environments, f. ex. multi user games or chat rooms, which need many similar mechanisms. As far as possible the design of the application prototype should seek to separate such more general parts from the parts specific to InstantSocial, with the aim to lay the foundation for an application framework for this class of applications.

1.2.2 Interpretation and Approach

The project description introduces a number of elements to this project. It is clear that a focus is on the InstantSocial application.

As the goal in the description is to develop a prototype to assess the realism of the application and suitability of the platform, this becomes the chief approach of this project. However, as the time available to this project is likely too short for a prototype with all functionality to be developed, the focus is placed on the indicated adaptation and context changes. Rich functionality is not deemed as important, as long as the design and prototype can support the indicated Internet social networking trends and principles on a mobile ad hoc environment.

1.3 Research Questions

The research questions of this thesis has been chosen in a collaboration with the supervisors and aims to ask the questions that is most likely to both be answerable in the short timespan available and which will also produce the most interesting results. The resulting questions has been derived from problem description in 1.2, and can be found in table 1.1.

DESIGNATION	DESCRIPTION.
RQ1	How can the MUSIC framework be utilized for developing social networking in mobile and ad-hoc environments?
RQ1.1	What functionality of the MUSIC framework is applicable for developing ad-hoc networking?
RQ1.2	What architecture is best suited for supporting such an application?
RQ1.3	What are other typical applications related to ad hoc social networking, and how can the architecture be designed to best separate these parts so that potential reuse is maximized?

Table 1.1: Research Questions

:

1.4 Contribution

As will be shown in this report, the project undertaken during this thesis is in a novel field. Although we will see that social applications and context-sensitivity have begun to receive some attention from scientific and professional communities, the application and design presented in this report is, in the unique coupling of social applications and context sensitivity, an interesting new development indeed.

The main contribution of this report is the design and prototype of the proposed system, InstantSocial. This application is built on existing scenarios and ideas of the various European scientific and professional communities behind the Self-adapting applications for Mobile Users In ubiquitous Computing environments (MUSIC) initiative, but this is the first time such an application has been tested properly and developed by a developer external to the initiative. Although the prototype developed is only an early prototype, this is a valuable first step in this exciting, emerging field.

1.5 Guide to this Report

Following is a short overview of this report, with a short description of what each chapter and section contains.

Chapter 1 contains the introduction to the report, with research questions and research approach.

Chapter 2 contains important background information on the subject. This is important for understanding the later chapters.

Chapter 3 contains the research approach of the project, both background information on relevant theory and choices made and plan for the research.

Chapter 4 contains the design of the proposed application, from high-level architecture to the use of the technology and frameworks.

Chapter 5 contains the details of the implementation phase, in which the prototype is built. It is focused on the choices and deviations from the design made during implementation

Chapter 6 contains the results of the implementation, with fulfillment of requirements, test results and experiences obtained during implementation.

Chapter 7 contains the conclusion, which tries to answer the research questions proposed in the beginning, based on the results obtained.

Chapter 8 contains a look ahead at the future work of this field after this project is completed.

Chapter 2

Background

> To steal ideas from one person is plagiarism, to steal ideas from many is research
>
> Unknown

This chapter contains the background information researched through the prestudy section of the project. It is an introduction to the terms and fields used and discussed later in the report, and covers some of the existing work in this field. Mobile computing, Context-awareness, a proposed w3c ontology and other social applications that cover some of the areas of this project will be examined. We will also give a short description of the MUSIC framework, which can be used to develop such applications.

2.1 Mobile, Pervasive and Ubiquitous Computing

Ever since the beginning of the computer sciences, the trend has pointed towards ever smaller and more mobile devices. The gargantuan early room-sized mainframes have largely given way to personal terminals. These again are being challenged by laptops, and even handheld devices. In addition, thousands of tiny computers surround us every day, in everything from cars to washing machines. Today, most phones can run simple programs, and network availability and performance are ever-increasing. This has led to the emergence of several new, interesting fields in computer science.

Although there are some differences in various literature, and the various terms in this field is sometimes used somewhat overlapping, a good introduction and definition of important terms in this area is given in [LY02]. This article to the field introduces pervasive, mobile and ubiquitous computing in a well-defined manner, and is the basis of the definitions given in the next paragraphs.

Mobile computing simply implies that a computing device is mobile, and can be moved about with little difficulty [LY02]. Moving from traditional computing into the mobile realm means limiting device size, while still retaining as much as possible computational power, battery lifetime etc.

Pervasive computing indicates that a device uses the environment around it to some extent [LY02]. This includes, but is not limited to, using information provided by other devices in the environment and detecting and responding to other devices present. Naturally, a high level of pervasiveness cannot be accomplished by a single device alone, it is imperative that conditions in the environment is established to support the device.

The most interesting such new computer field is that of the *ubiquitous computing*. Ubiquitous computing is the sum of both mobile and pervasive computing, and indicates that a device is both highly mobile and reacts to it's environment in an interesting way [LY02]. Ubiquitous computing opens new possibilities for interactions, computational patterns and indeed wholly new applications that would not be possible on older, "classic" computing devices.

These new possibilities enables a whole new way of using the devices and applications, which is now beginning to become apparent to researchers and potential users alike. It also, however, introduces some new difficulties. As devices move in and out of reach of each other and other embedded devices, providing a stable, usable network communications implementation becomes increasingly tough.

Mobile, pervasive computing naturally differs from computing on traditional, stationary terminals on many points [FZ94]. Because of their mobile nature, mobile devices have always aimed to be small, light and highly portable. Because of this, screen size, battery lifetime, computing power and other hardware capabilities have always been limited.

An interesting and promising element of mobile applications are the possibility to form *ad-hoc networks* . An ad-hoc network is a decentralized network that dynamically detects and determines participant nodes. An example of this is two mobile devices that both run some application. When these devices detect each

other, they can form a network, sharing services, information and more. As more devices comes into range, they can dynamically join the network, as well as leave without causing collapses or severe problems for the network. These networks are typically distributed and not reliant on a server or centralized topology, and exist only for the time they are needed, and are then stopped.

2.2 Context-aware and self-adaptive Applications

Context-awareness and adaptation is a field of much interest in pervasive computing. As mentioned in section 2.1, pervasive computing is concerned with the interaction between the device and the environment. As we will see in this section, context is a central concept in this field.

2.2.1 Context

Context can be seen as all information concerning the situation of a person, place or object which is relevant to the interaction between a user and an application [DA00], although several other definitions have been proposed with some differences.

It is apparent that this definition of context allows for a very wide variety of context elements. Context concerning the situation of a person can be anything from the fact that the user is busy with some other task (like a meeting) and does not wish to be disturbed, or that the user is blind and thus cannot use a purely screen-based interface.

Context concerning place can vary from the (now popular in a number of devices) location information that can be retrieved from Global Positioning System (GPS), network coverage, temperature and a lot of other potentially interesting bits of information.

Context related to relevant objects naturally includes the device in which the application is running. This can be information that differs between various devices that can potentially run the application, such as screen resolution available. This can be important, as the application could potentially use this information to choose a graphical interface that matches the available screen resources, and is unlikely to change much over time (as long as the same device is in use). Also, it can be context that could change over time on the single device, such as battery power.

Interestingly, the definition can also include other devices (or other objects, for that matter), if these are relevant to the interaction. For instance, for a server-based application, the availability of a server to connect to would be very a interesting context. For a peer-to-peer application, such as the one researched in this project, the availability and statistics of available peers would be a similarly interesting context.

2.2.2 Context-aware Applications

An application that somehow senses it's context, either through sensors or other means, is considered context-aware [BNSW94]. This can vary from a user entering the context information manually to a sensor system where the user doesn't see or notice anything changing at all. It is, however, likely that potential users will find the latter a better solution so as to minimize their work.

Naturally, what context elements are considered interesting will vary a lot depending on the application. Since the definition of context in section 2.2.1

allows for a huge amount of very different context elements it is important to develop a common approach to deal with such information. We will see some of the work done in this area later.

2.2.3 Self-Adaptive Applications

A self-adaptive application is an application that is aware of it's context and adapts to this. Note that many definitions of context-aware applications includes self-adaptation. Self-adaptation includes limiting functionality when a mobile device is running low on battery, changing display options to better fit different lightning conditions and changes depending on the location of the device, to name a few. Because of the limited screen sizes, computation power, battery life and storage of mobile devices, context-adaptation is an important means to maximize the usability of these devices.

2.3 W3C Delivery Context Ontology

The world wide web consoritum (www.w3c.org) is currently working on an ontology of delivery context, which aims to provide a formal model of the environment in which devices interact [Fon09a]. This ontology is a possible future W3C recommendation, and is currently in it's second draft. The first draft for this document was published in 2007.

The document covers a lot of information regarding the delivery context of devices, many of which are not interesting to this project. However, some of the information in the ontology is potentially covered in, or of interest to, the work done in this report, and they have therefore been studied and covered in this section.

One thing of particular interest to this report is the ontology's coverage of devices' hardware capabilities. The main overview of this can be found in figure 2.1, with shows details covered in the ontology available on the W3C homepage. Note that this table has been simplified somewhat for readability. The full original can be found at [Fon09b].

As is obvious, this ontology covers many different aspects of the hardware. Although it only covers the hardware statistics of a device and not dynamic "current" values, it's a small job to identify which of the properties have such potential to change. As an example, "supportsVoiceRecognition" is unlikely to change much according to the context of the device, while memory, battery and CPU have both a maximum value (which is what is indicated in the ontology), but also have a "current" value that will change over time. Memory and CPU load could vary depending on the situation and use of the device, and the battery charge will typically be depleted over time. This is of much interest to a context-sensitive application.

NAME	DESCRIPTION.
batteries	This property represents the batteries in a device.
bluetoothSupport	This property represents the support for Bluetooth available on the device.
builtInMemory	This property represents the memory built into the device and which is not removable during normal operation.
cameras	This property represents a camera associated with a device .
display	This property represents a display associated with a device.
extensionMemory	This property identifies additional memory that is provided to a device, typically in the form of removable memory cards
inputCharacterSets	This property defines the character sets supported by the device for input.
inputDevices	The input mechanisms supported by a device
networkSupport	This property represents the network support available on the device.
numberOfSoftKeys	This property represents the number of input keys on the device whose function can be controlled programatically.
outputCharacterSets	This property defines the character sets supported by the device for output.
primaryCamera	The primary camera of the device
primaryCPU	This class represents the main CPU for a device.
supportsAudioOutput	This property defines whether or not the device supports the ability to output audio beyond the basic capability for supporting voice calls.
supportsVoiceRecognition	This property specifies whether or not the device supports voice recognition.
textInputType	This property specifies the text input type supported by the device.

Table 2.1: W3C Ontology Device Hardware, Simlified

2.4 Social Applications

The term Social Application indicates a type of application that supports human social and/ or collaboration abilities [Coa03]. The term is not well-defined, and a number of different definitions emphasizes different aspects. However, most concur that the central idea in social applications is to work with groups of people, and interactions between these ([Web04], [Coa03]). Typical for such applications is their reliance on groups of people, meaning that a "critical mass" of users is necessary for the application to make sense for the users.

Many websites popular nowadays can be thought of as social applications, including examples such as Facebook (www.facebook.com), YouTube (www.youtube.com) and Flickr (www.flickr.com). Typical for these services is that they are reliant on the users to provide content, while they provide the tools for people to be together and share content. Another popular term for this is "web 2.0". All these examples have, however, been developed primarily with the typical desktop user in mind. Providing such services on a mobile devices to entertain and support people on the move is an interesting challenge. This is one possible application of the results of this project/ thesis.

2.4.1 Existing Work and Applications in the Area of Research

There exists some current and in development applications that touch various topics covered by this project. Mostly, these are mobile social applications with little context sensitivity. Some of these will be examined here to give a sense of the "state-of-the-art".

Mobile social applications have been examined in [CtHS06]. This workshop on mobile social software touches social software, mobile devices and location services. However it is only concerned with research topics, not actual implementations of applications. Context is discussed briefly, but the only focus is on location and it's importance to mobile applications.

Dodgeball

Dodgeball (www.dodgeball.com) is a good example of several more or less similar mobile social applications starting to emerge and become more increasingly popular. Dodgeball is an application for meeting people with heavy emphasis on using location. The idea is that you inform the application of your location, and the application checks this against your friends' location. You can this way be informed of nearby friends, and vice versa. There is also some functionality for meeting new people, including a "crush" feature for meeting potential future partners.

The service is heavily server-focused, and requires the user to send his/ her location manually by SMS to the Dodgeball server. The service can be "started" and "stopped", and for camera phones the server can send photos of the nearby people you might know/ want to meet etc.

Dodgeball is included here as an example, several other quite similar applications exist. Sadly, Dodgeball was shut down by it's developer, Google, in the beginning of 2009. This seems due to cut-downs in Google because of the worldwide economic crisis [Boc09]

Mobile Web Server

Mobile Web Server (see [Tea09]) is an application for the Nokia S60 and similar mobile devices that lets you share information and content from your mobile device. Essentially, it lets you access information on your mobile device easily from a computer through a web browser. You can browse contacts, update your calendar, send SMS messages and share pictures, amongst other things.

Not only can you access your mobile device, it is also possible to give different levels of access rights to your friends and contacts. This leads to a number of interesting possibilities, such as sharing the camera of the device with your contacts, enabling them to take pictures through your device. With the proper level of rights they can also access the location information of your device (using GPS), and see the network availability and mode of the device at any time. Checking this information, a friend can avoid calling if the device is in "meeting" mode, or if there is not sufficient signal strength for proper communication.

Mobango

Some applications for file-sharing on mobile devices exists, and Mobango is a good example of such an application. Mobango (www.mobango.com) is a file-sharing application for mobile devices (also usable from computers) that lets you upload, share and download files from the Internet. The application has support for friends and groups, but does not appear to use any context information. It does not use location and always downloads data from the central servers.

tunA

tunA (http://web.media.mit.edu/ stefan/hc/projects/tuna/) is an application in development that aims to allow users to stream music between mobile devices on ad-hoc networks. Although the project has received some interest, not much information is yet available on the design and technical choices.

2.4.2 InstantSocial Idea and Scenarios

The idea for the InstantSocial application, which is the purposed artifact for this research project, is a multiuser application which dynamically forms ad hoc communities of neighboring mobile devices and establishes a server infrastructure supporting sharing of multi-media content. It uses and adapts to context information and changes, as is apparent in the following scenario, based

on an earlier scenario from [LFS08] :

Paul is visiting a large rock festival. During a Bjork concert, he is not able to take a good shot, others could have done better.

Back at the tent, Paul is willing to share his pictures with others. He starts the InstantSocial application and his PDA notifies him about the presence of a media sharing group. Present in the group is a single other user, Adam. Adam is currently using his device for other work, and thus his device does not have a lot of resources available.

Paul happily joins, gives high priority to this application, and a moment later his display shows a selection of pictures, each representing a collection of shots matching his interests. He browses through the content, selects the ones he likes, and begins to download them to his device.

After a while another user joins the group, Bill. Bill has a high-end mobile device and is not doing much other work, so he his device has a lot of available resources. Paul browses the additional pictures provided by Bill.

After a while, Adam turns off his device to conserve power, leaving the group. Other than that his pictures disappear from the selection, this does not affect the other users.

When Paul is done with the browsing he lowers the priority of the application so he can listen to some MP3. He still leaves his media (and CPU power) available for others to engage in the InstantSocial platform.

Some songs later, he decides to save some battery for phone calls and indicates his wish to leave the group. The PDA asks him to wait a few seconds. After getting the OK, he quits.

2.4.3 Common Features of Social Applications

As indicated by the introduction to the matter and the various examples provided, there are some features that are common for social applications. The most important such features is summarized in the following list.

- Social Applications focus on groups of people, and collaboration between users.
- Social Applications focus on user created content, and try to provide easy tools for the users to share and access such content.
- Social Applications are reliant on a user base. With no or too few users providing content, the applications are of little interest to users.
- Has to tolerate users joining and quitting dynamically without disturbing other users to much.

Additional Features for Mobile Ad-Hoc Social Applications

The above features are for all social applications. When looking at such applications running on mobile ad-hoc networks, these features will also be

important for a successful application:

- Should be able to run on very resource constrained device and take advantage of available resources in the network to enrich the functionality of the tiniest devices.
- Should tolerate that users need to temporarily switch to other applications during a session.

2.5 OSGi™

OSGi™is a Java platform allowing applications to define, share and dynamically reuse various components [All09]. It allows different modules and applications to define services, which can be used by other modules easily. The components can share functionality through services while hiding their implementations. The OSGi™framework includes a service registry, which greatly simplifies the sharing of services between the various modules. The platform also supports the most popular service discovery protocols, including Service Location Protocol (SLP) (see [EGD99]). This enables publication, discovery and use of services hosted in the OSGi™by remote computers using standard networks.

2.6 the MUSIC framework

Self-adapting applications for Mobile Users In ubiquitous Computing environments (MUSIC) is a research project aimed to support development of applications for mobile devices, with emphasis on context-awareness, self-adaptation and distributed, mobile systems [Tec09].

MUSIC is a joint research project between important European universities, research organizations and IT companies, including SINTEF, Hewlett Packard and Telecom Italia [Tho08].

The MUSIC project is a comprehensive scientific undertaking, and comprises of modeling languages, model specification and validation tools and a comprehensive open-source software development framework amongst other things [Tho09]. The project has produced a number of scientific papers, with many more planned in the future.

The MUSIC middleware framework monitors and adapts to context changes, and reconfigures the application in accordance with the changing context. MUSIC is based around self-adaptation, which means the application should adapt itself fluidly, without requiring specific actions by the user for this.

2.6.1 Generic Middleware-supported Adaptation

The MUSIC framework is based around the idea of using architecture models for runtime adaptability [JFG06]. The advantages of using models of the architecture to adapt to changes are many, including reduced complexity, good scalability, easier development and increased reuse of adaptation mechanisms. Of course, this means the architectural model must include the details of the context changes and dependencies, the varying Quality of Service (QoS) levels and resource needs. This information is is added in the form of property predictor functions and used in a utility function, a function which calculates the best adaption based on the available data. The middleware tries to maximize the utility for the application at all times, choosing those component instances providing the most utility at a given time, varying on the context. The tools and modeling language provided through the MUSIC project are designed with this approach in mind and simplifies this process greatly, with the added benefit of clarifying these issues at an early stage of development.

2.6.2 MUSIC service approach to context adaptation

The service approach is one way the MUSIC framework supports context adaptation. This entire section is a short simplification of the information found in [Ita09]. The interested reader is strongly encouraged to study this document further for better understanding of the MUSIC framework.

MUSIC uses an architecture model of the running application with different cooperating components. These components have different implementation and runtime variants, which the MUSIC middleware changes between depending on

which fits best at a given time. These components can be either local implementations or services offered by other devices/ webservices. Parameterization can be used to instigate change in the interior of a module, varying the properties and dependencies of the modules.

The components provide services, in accordance with the Service-Oriented Architecture approach. This is therefore central to the service approach to context adaptation. The components in the system collaborate by using and providing services to each other. The OSGi^{TM}platform (see section 2.5) is used for the sharing of services.

2.7 Relevant parts of MUSIC

For the proposed InstantSocial application, several parts of the MUSIC framework is of interest. From the scenario provided, it is apparent that there are several modes of operation for the application, depending on the context. As explained in section 2.1, one of the challenges of mobile devices is to retain the best performance while battery power, screen sizes and other resources are limited. A good way to maximize the performance under such hard conditions is the approach introduced in the scenario, which moves some of the heavy workload from resource-poor devices to other nearby devices with more resources available. The MUSIC middleware-supported adaptation is one possible and seemingly well-fitting way to enable this.

This adaptation also seems well-suited for use with the service approach for context adaptation. Identifying different components with their services and different implementations of these, the wanted different behaviors should be achieved by the middleware. The provided tools and modeling languages should also ease the identification, modeling and development of the various context and adaptation information.

As is indicated by the common features of social applications listed in 2.4.3, a central feature of social applications is the focus on users, and user-provided content. This is likely to provide a changing context, where availability of other users and the level which a user shares or browses content can be seen as changing contexts. The MUSIC modeling tools should make early identification and support for such variations easier, and will be helpful in the design and development of the prototype application.

Chapter 3

Research Approach

This chapter covers the research approach taken in this project. It will both provide some background information on relevant theory as well as explain and try to justify the choices of research approach.

3.1 Design Science

The Merriam-Webster Online Dictionary, [MW09] lists a number of uses for the word *design*, including:

"to create, fashion, execute, or construct according to plan"

"to conceive and plan out in the mind"

"to devise for a specific function or end"

Merriam-Webster Online Dictionary, [MW09] lists a number of uses for the word *science*, including:

"the state of knowing : knowledge as distinguished from ignorance or misunderstanding"

"a department of systematized knowledge as an object of study"

"something (as a sport or technique) that may be studied or learned like systematized knowledge"

Design science is an approach that seek to create and evaluate artifacts that solves some problem in information research [HMPR04]. An artifact is understood as a working design [VKV08]

There are some different opinions on what the important outputs of a design science approach should be. A good definition is provided in [VKV08] and presents five important outputs. A listing of these can be found in table 3.1, which is borrowed from that book.

OUTPUT	DESCRIPTION
Constructs	The Conceptual vocabulary of a domain
Models	A set of propositions or statements expressing releationships between constructs
Methods	A set of steps used to perform a task - how-to knowledge
Instantiations	The operationalization of constructs, models, and methods
Better Theories	Artifact construction as analogous to experimental natural science

Table 3.1: Design Science Output, taken from [VKV08]

It is no surprise that different research projects and their artifacts produce these outputs at different weights and importance. While one artifact might first and foremost be used to explore and reason on the relationships between various

of it's constructs another might lead to better theories in the field of research, for example by proving that a developed theory does not hold in some practical way.

For this thesis, the most important output will be the models and instantiations. The conceptual vocabulary of the domain covered has received much attention and research in the last years (see chapter 2), so no new discoveries here is expected in this project. As for the methods, the goal of this project is to use excising methods and constructs. The result and experiences from using this could, however, give some valuable feedback on these. Better theories for the field is considered somewhat outside the scope of this project, as the main focus will be on the models and instantiations. These could lead to new and better theories, but this is more likely to appear in further work building on this thesis, and focusing more on the theory of the field.

3.2 Evaluation Approach

An important part of design science is the evaluation of the proposed artifacts. The importance of this is stressed in [HMPR04], where the writers name a number of possible and appropriate ways to evaluate the artifacts of a design science project. The different methods are Observational, Analytical, Experimental, Testing and Descriptive. In this project, the evaluation will be done mainly by Experimental and Testing means. The perhaps most important result of this report will be whether the proposed design is operating as intended when tested with different data. Simulation and Black Box testing will be the means used to reach this goal. There is also a short Descriptive evaluation provided in the form of a scenario, but this is taken from earlier work in this field and cannot be seen as a new discovery.

3.3 Project Research Design

The research approach for this project will be to use the design science approach to create the design and prototype implementation of a typical social application for use in mobile ad-hoc environments, with support for adaptation to changing context and communication between peers. This should provide important insight into the area of research and should provide the answers to the research questions.

3.3.1 Choice of InstantSocial as Prototype Application

The concept of the InstantSocial application is introduced in section 2.4.2. In this project, the InstantSocial application is used as an example of a typical mobile social application because of the mobile nature of this application and the excellent match with the features presented in sections 2.4.3 and 2.4.3. This match is shown in table 3.2.

FEATURE	INSTANTSOCIAL MATCH.
Social Applications focus on groups of people, and collaboration between users	In the InstantSocial application, the users are central. As the application is entirely peer-to-peer, the users become critical
Social Applications focus on user created content, and try to provide easy tools for the users to share and access such content	The purpose of the InstantSocial application is sharing content between users. In the presented scenario, the content shared is pictures, which will also be the chief focus of the prototype.
Social Applications are reliant on a user base. With no or too few users providing content, the applications are of little interest to users	This is extremely important in the InstantSocial example. Without local, connected users to provide content, the application provides nothing for the end user. Therefore, critical mass is critically important.
Has to tolerate users joining and quitting dynamically without disturbing other users to much.	This is also present in the InstantSocial scenario description.
Should be able to run on very resource constrained device and take advantage of available resources in the network to enrich the functionality of the tiniest devices.	The InstantSocial scenario uses mobile devices collaborating.
Should tolerate that users need to temporarily switch to other applications during a session.	The InstantSocial example includes users working on other applications, so this is applicable to the InstantSocial application .

Table 3.2: Features of Social Application and InstantSocial Match

In addition to the good fit with the social application features, the InstantSocial scenario clearly presents heavy use of context adaptation, mobile ad-hoc networks and various devices sharing the job of computation. This excellent fit with the project goals presented in section 1.1 is the reason InstantSocial is chosen as the prototype application for this project.

Chapter 4

Design

> Software and cathedrals are much the same - first we build them, then we pray
>
> Unknown

This chapter contains the design of the proposed application, from requirements and high-level architecture to lower-level design. A short discussion of technology choices is made, and the context-sensitive design information is discussed.

4.1 Identified Requirements

This section contains the functional requirements for the project, as identified from the presented scenario and objectives of the application. All requirements are given indicators of priority and difficulty, to better prioritize what requirements are important and plausible to implement in the limited time avaliable.

4.1.1 Functional requirements

FR1 The user should be able to set a priority to the application, determining its relative importance to the user compared with other applications running concurrently, and consequently influencing its share of the available resources.
Priority - LOW *Difficulty* - MEDIUM

FR2 The user should be able to join groups based on available groups and his interest.
Priority - MEDIUM *Difficulty* - MEDIUM

FR3 The user should be able to browse content available in his joined group(s).
Priority - HIGH *Difficulty* - MEDIUM

FR4 The user should be able to download selected content from his joined group(s).
Priority - HIGH *Difficulty* - MEDIUM

FR5 The user should be able to share his content with the joined group(s).
Priority - MEDIUM *Difficulty* - EASY

4.1.2 Non-functional requirements

NFR1 The application should be able to run on typical mobile units. See 4.1.3
Priority - HIGH *Difficulty* - MEDIUM

NFR2 The application should be fully distributed. No central server should be required.
Priority - HIGH *Difficulty* - HIGH

NFR3 The application should be dynamically self-adaptive to react to the numerous possible changes in the context. See 4.1.3

Priority - CRITICAL *Difficulty* - HIGH

NFR4 The application should make use of the MUSIC framework to better enable context-awareness.
Priority - HIGH *Difficulty* - MEDIUM

NFR5 The application should be able to run along with other applications, and in the background.
Priority - HIGH *Difficulty* - LOW

4.1.3 Clarifications

This section contains clarifications to the requirements presented above, which provides more details on these requirements.

Typical Device

With typical device, we define as a minimum the ability to run the Connected Device Configuration (CDC). The typical specification provided by Sun Microsystems ([Mic03]) is:

- 32-bit microprocessor/controller
- 2 MB of RAM
- 2.5 MB of ROM

Context

For a discussion on the relevant context, refer to section 4.5.

4.2 Choice of Top-Level Architecture

A number of different top-level architectural approaches for the application and requirements are possible. This section will discuss some possible solutions and justify the one chosen.

4.2.1 Adaptation Master

One possible solution to the adaptation of the numerous devices connected is to let one of the devices act as a "master", controlling how every device is adapting. This is, clearly, not a good solution as this master provides a single point of failure which does not fit the scenario or problem description well. Also, this is unlikely to scale well.

4.2.2 Server Infrastructure

It is possible to use a classical server infrastructure to provide a functional application, but this does not fit the scenario, problem description or identified requirements well.

4.2.3 Ad-hoc peer-to-peer

An ad-hoc peer-to-peer architecture might provide the most challenging approach, but it is also a much better fit to the problem description, requirements and the scenario. This solution requires dynamic joining, sharing of functionality and a peer-to-peer solution for sharing the content. The MUSIC framework supports this mode well, and this solution is chosen and used in the architecture in section 4.4.

4.3 Choice of Technology

As is apparent from the scenario presented in section 2.4.2, the application contains a high use of context information with ad-hoc networks in a way not seen before. For the development of this application we have chosen the MUSIC framework, covered in section 2.6.

4.3.1 Chosen use of the MUSIC framework

The MUSIC framework is very comprehensive, not all parts are applicable to the application of this project. Also, the MUSIC project is not completed, so not all parts of the framework are yet available/ complete.

Service approach to context adaption

One of the approaches for context adaption supported by the MUSIC framework is through the use of services. By using a service oriented approach, adapting to changes in the context can be as easy as connecting to a different service. This is the chosen approach of this design, as it provides an easy, flexible way to change the provider of functionality. Other possibilities exist in the framework, but this is chosen for it's good tutorial support and good fit to the nature of the presented problem.

4.3.2 Programming Language

The choice of the MUSIC framework forces the choice of programming language, as the only currently available implementation of the framework supports Java. As a high-level language with good support for services, modular development and with very good tool support, this language is seen as well-fitting for this project. However, the design of the system has aimed to be largely language-independent, and if a future version of the framework is created for another language, the design should require minimal changes.

4.3.3 Services and Service Discovery Protocol

The preferred, and only currently supported, service method of the MUSIC framework is the OSGI approach to plugin development with service hosting and consumption. See section 2.5 for information on this.

For service discovery, the MUSIC framework supports both the upnp and SLP protocols. As it was believed that availability for SLP was better than upnp on limited mobile devices, this was chosen as the protocol of choice for the application. However, this does not affect the design much, and changing this is a simple matter.

4.4 High-level Architecture

This section covers the high-level architecture for the proposed service-based system. This has been partially based on the limited architecture proposed in [LFS08]. The architecture was designed after the decision to use the MUSIC framework, and reflects the possibilities and constraints of this choice.

4.4.1 Top-Level Device Deployment and Collaboration

The overall idea of the design solution is that multiple devices will be running their own, local instance of the application. These instances will use the MUSIC framework to discover and communicate with each other.

The application is divided into layers (see section 4.4.2). The layers provide a set of services to the layer above it, and these services are also published through MUSIC, using SLP through the OSGi^{TM}framework (see section 5.1.2) to negotiate the provided properties of each service. Thus, an application layer on a device can use the services of a lower layer on a nearby device, should this be more effective than to run the layer locally. This will enable resource-poor devices to utilize the services of other devices nearby with more available resources, providing better performance. Of course, this service utilization is completely dynamic, and adapts as the context (including provided services) changes. The various layers are provided in the next section, 4.4.2.

4.4.2 Layered Design

The system is divided in three layers, the Browser Proxy, Presentation and Content Repository layers. Each layer provides a service to the layer above it, and each layer can have different variations depending on the context. With the exception of the Browser Repository layer, the layers can be run locally on a device or on another, nearby device as decided by MUSIC at runtime. A system overview can be found in figure 4.1.

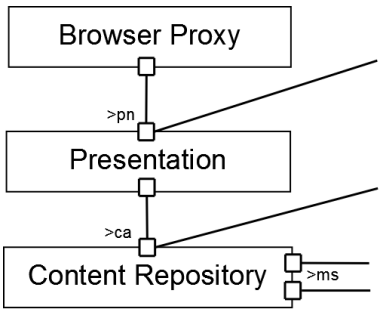

Figure 4.1: System design overview

Figure 4.1 identifies the different services offered between the layers. The presentation layer offers the Presentation Service (PN) to the browser proxy. The content repository offers the Content Access Service (CA) to the presentation layer. As can be seen in the figure, these services can be used by the local

device, or other devices (as indicated by the lines from the side). Also, the content repository uses the Membership Service (MS) to connect to other content repositories, exchanging information as a means to provide distributed file access.

Content Repository

The content repository is the bottommost layer in the system, and provides the Content Access Service (CA) service to the presentation layers connected to it. The content repository also connects to other content repositories using the ms Membership Service (MS), and together the content repositories are responsible to keep a level of replication to the data. This should ensure that the failing/ dropout of one such repository is not fatal to the availability of content in the system.

The content repository must keep a record of all content available throughout the network, and provide this content to the connected presentation layers, should they request it. Thus, effective communication between the content repositories in this "‘content repository ring" becomes important for the performance of the system.

Presentation

The presentation layer feeds the information that is to be displayed to the browser proxy. Thus, it provides the essential job of monitoring the available content and selecting what content is to be presented to the user. The presentation layer offers the Presentation Service (PN) to the browser proxy, which can be local or on a connected, nearby device. This is simply delivered as a XHTML (or similar) list of content to display.

A high-level view of the presentation layer can be seen in figure 4.2. As indicated, the layer is composed of two modules, the content generator and the thumbnail generator.

The content generator is the main part of the layer, and provides the Presentation Service (PN) to the connecting browser proxies. It also uses the content repositories' Content Access Service (CA) to access the content and the logic to decide what to present to the user, based on user preferences and group membership.

The thumbnail generator is a much smaller module, that simply deals with generating thumbnails for the pictures that are to be presented. This is done internally in the layer as a service, and the thumbnail generator can have different variations, providing different levels of quality at different costs of resources.

Browser Proxy

The browser proxy is the topmost layer, and it's main purpose is to present information to the user. Because of the wide availability of simple browsers, familiarity to most users and ease of development and modification, the Browser Proxy will display information as a website. This will enable ease of development,

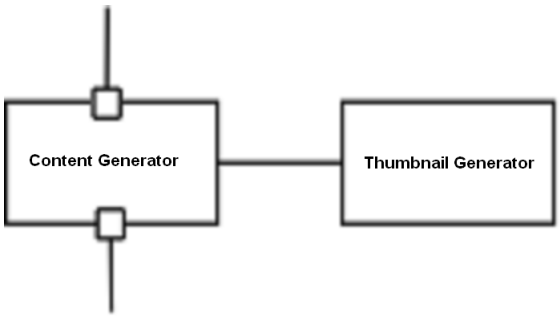

Figure 4.2: Presentation layer internal structure

and should also ensure that at least a fair amount of modern mobile devices are supported without much fine-tuning to different screens and UI interfaces. However, the design also supports having multiple variations of this module, allowing different presentation forms based on context information such as available power, available screen or other things. This could allow the system to display fewer colors if low on battery which could save energy if the device uses organic light-emitting diode (OLED)s, for instance.

4.4.3 Modes of Realization

With the layered architectural approach, and the possibilities of each layer being run either locally or on a remote machine, this leaves the system with a few different realizations of the system depending on how many layers are run locally.

isFull realization

The isFull realization is the mode in which the largest part of the system is running on the local client machine, and the realization which is the most resource-dependent. The isFull realization instantiates the browser proxy, presentation layer and content repository on the local device, connecting itself to a membership service of another, nearby content repository. It will then join the "content repository ring", providing the content to nearby presentation services, and the membership service to other devices that might want to create and connect their own content repository. An overview of the isFull realization can be found in figure 4.3

isMini realization

The isMini realization instantiates the browser proxy and presentation layers on the local device, but is dependent on a remote content repository to provide the content requested by the presentation layer. As the content repository is likely the most resource-heavy of the layers, the isMini realization should help resource-poor devices attain better performance by using the functionality of other, more

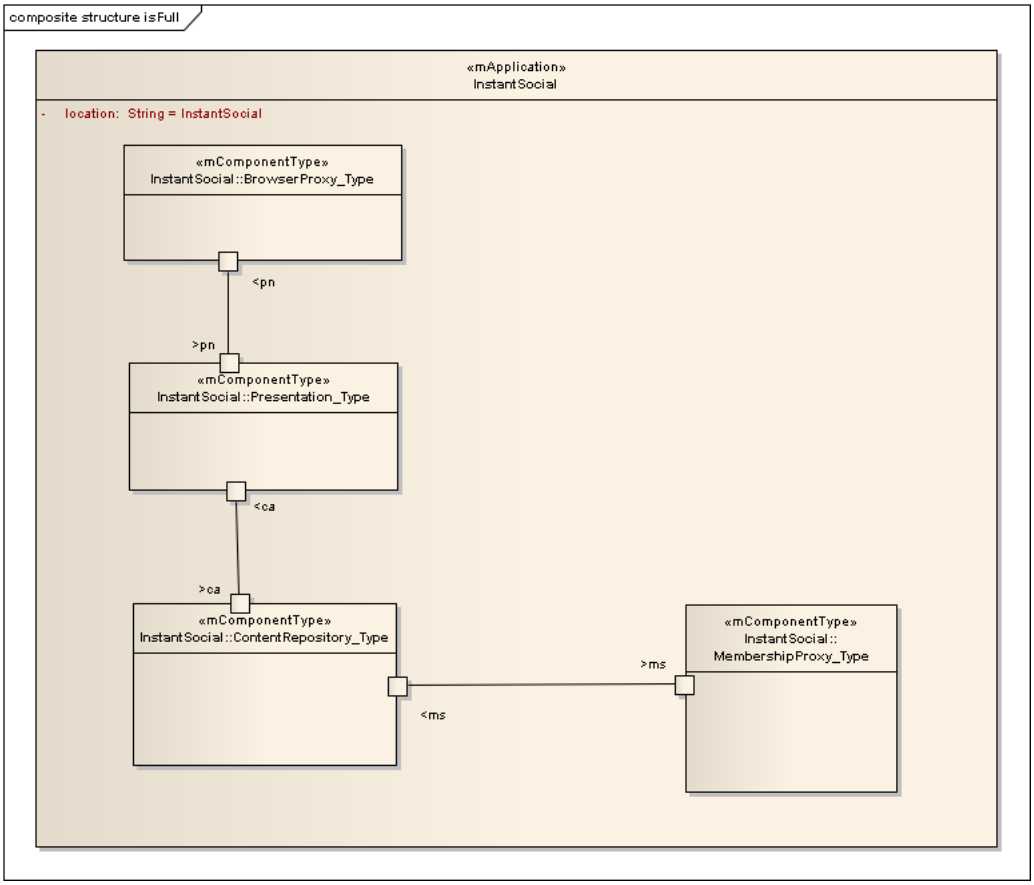

Figure 4.3: isFull

resource-strong devices nearby. An overview of the isMini realization can be found in figure 4.4

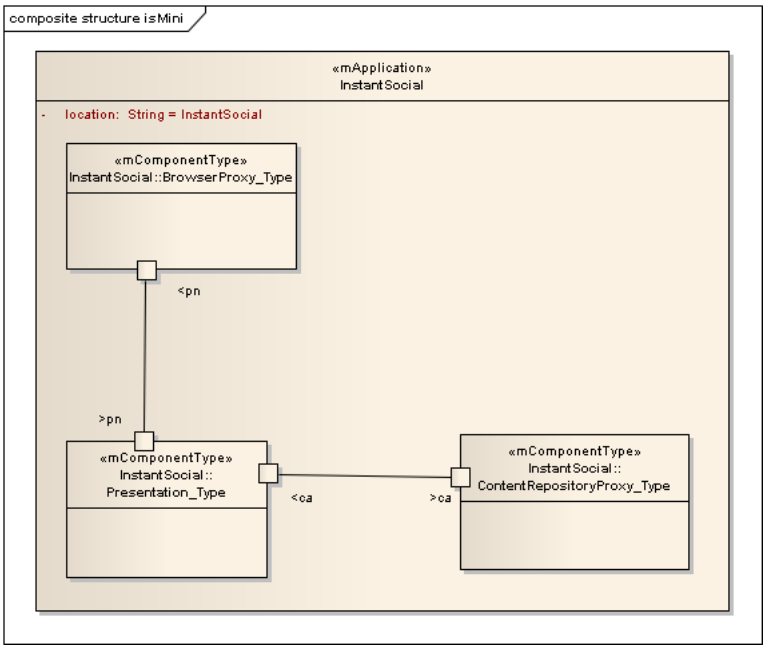

Figure 4.4: isMini

isLeech realization

The isLeech is the simplest of the operational realizations, only instantiating the browser proxy on the local device, and connecting to a presentation service of a remote device to provide all content and provide the functionality. This is the least resource-dependent of the realizations, but as the device must speak to a remote device every time something is to be shown or changed, performance will likely be much poorer than the other realizations. An overview of the isLeech realization can be found in figure 4.5

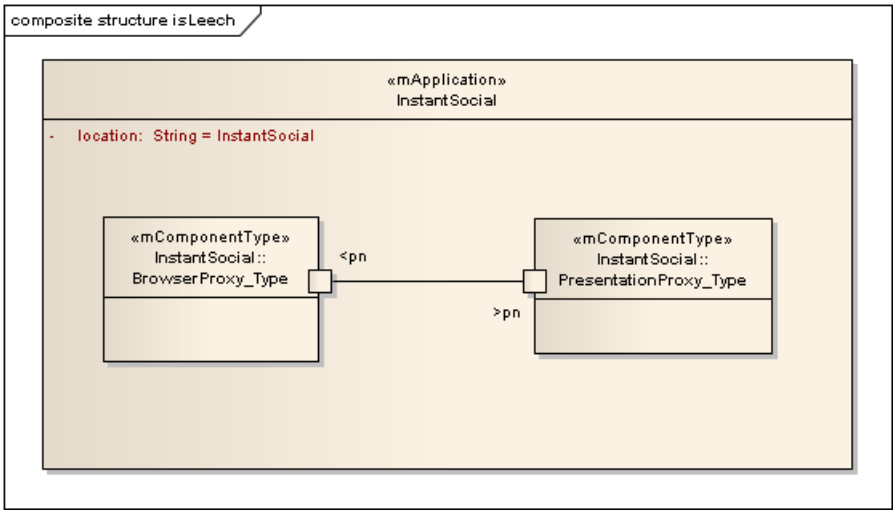

Figure 4.5: isLeech

4.5 Context Changes

This section covers the various context elements available to the system, and the variations that can exist within the system to adapt to the changes in this context.

4.5.1 Context

There are a number of relevant context for the system to monitor and adapt to. Some of these will be more important than others, and this will be reflected later in the utility function of the adaption reasoning. As is apparent, these contexts often work against each other, meaning the choice of what to prioritize and how to adapt to this becomes a complex problem.

Device resources

The main group of context to take into account is the resources available to the system on the device. This includes:

- Computational power: This is an important hardware constraint, and a measure of how much workload the device can handle before long delays ruin the performance.
- Available memory: This is also an important hardware constraint. Running out of memory could seriously hamper performance, or even stop the application from working properly.
- Battery power: This is very important on a mobile device. If the system uses too much power, the battery could run out and force the entire device to shut down, not just the InstantSocial application!

It is important to note that this does not only cover what is potentially available on the device, but also how much resources the user is willing to commit to the application. There could be big differences in how much battery a device has and how much the user wants to use for this application. For instance, what if the user knows he is at rock festival and won't be able to recharge his mobile device for a week?

Communications bandwidth

Available communication bandwidth is also an important context. If there is little bandwidth available, communication with other devices will probably take much time, and the client might be inclined to do more work locally.

There might also be a difference here between what is available and how much the user wishes to commit to the application. What if the user wants to reserve most of the bandwidth for surfing the web at the same time as using the application?

Group presence

Another important context in the application is group presence in the system. This will be incredibly important in determining what content to access and view.

4.5.2 Adaptation

There are a number of possible ways the application can adapt to these changes withing the MUSIC framework. These are the important ones for this implementation.

Distribution of computations among peers

An obvious way to adapt to context is to change the distribution of computations. That is, if one device has little available resources, it can contact another device to do the computations. This should ensure the device low on resources does not exert itself too much, however it might of course hamper performance in some cases.

Adaptation of duplication of computations

When performance is important, a possible solution could be to duplicate the computations. If more than one device is set to perform the computations one can always use the fastest response and forget about the rest, possibly increasing performance. Also, if one device fails during the computation, another will probably finish before it so the client doesn't have to wait while the computation starts all over.

Component realizations

MUSIC supports changing the realizations of parts of the system at runtime to adapt to context changes. This means that another variation of a component that better fits a given context can be swapped in when the context changes. As an example, a thumbnail editor that doesn't produce as good-looking thumbnails, but uses less resources, can be used when resources becomes more scarce.

4.5.3 User profile

As discussed, the context is not only concerned with what resources are available, but also with what the user wishes to commit. For this, the MUSIC framework's user profile can be used, where the user can set up their preferences to the various contexts.

4.6 Properties, Predictors and Utility of the design

As the proposed design uses the MUSIC framework for adapting to the context changes, a number of choices had to be made concerning how this was best achieved. As it would be difficult to use all the various context elements presented in 4.5 directly, the approach presented in [LFS08] is used. This article, in which the implementation of InstantSocial is introduced, contains an approach to the properties, predictors and utility function which was deemed very appropriate for the prototype.

4.6.1 Used Properties

In the MUSIC framework, properties can be declared and used however the designer wishes. [LFS08] proposes to use two simplifications of the context of the system.

Resource Utilization

The resource utilization is an abstraction of the resource context elements that is presented in 4.5.1. Resource utilization is simply a number between 0 and 100, where a resource utilization of 100 means all the resources of a device is used, and no more resources are available for the application. A resource utilization of 0 means that no resources are used, and that much resources are potentially available.

The gathering of data and abstraction from the various resources of a device to the resource utilization property is done using a specially constructed Resource Utilization Sensor. Such specialized sensors are well-supported in the MUSIC framework.

Availablity

The availability property is another abstraction, representing a prediction of how fast and how reliable a service or module will respond to queries.

4.6.2 Property Predictors

The property predictors are used to calculate the properties for each of the realizations of the application. This will then be used to calculate the utility (section 4.6.3). Each realization has a property predictor for each of the properties, but some of them are somewhat similar.

isFull Predictors

Resource utilization for the isFull realization is the resource utilization of the local device, read through the Resource Utilization Sensor. This is the most important resource utilization in this case, as all the layers are running locally.

Availability for the isFull realization is a bit more interesting. This is based on the number of content repository instances available in the connected devices. The more devices, the better the availability as each content repository is likely to serve less presentation layers. The availability is calculated using table 4.1

Number of Instances	Availability.
< 1	30
1	70
2	90
> 2	100

Table 4.1: Mapping from number of content repositories to availability

isMini Predictors

Resource utilization for the isMini realization is the larger of the resource utilization of the local device (as with isFull) or the resource utilization of the device providing the content repository service. This context property is defined as a property of the service, and the framework reads and responds to changes in this.
The availability of the isMini realization is similarly the availability of the content repository service. This is provided through the service, and is calculated on the remote device similarly to the isFull predictor (see table 4.1)

isLeech Predictors

Resource utilization for the isLeech realization is provided by the presentation service, and is calculated on the remote provider of the service similarly to the isMini realization. Availability is also read directly from the presentation service properties, and is also calculated similarly to the isMini realization on the remote device.

4.6.3 Utility Function

As introduced in section 2.6, the utility function is the MUSIC approach to choose between different realizations of the application. The utility function uses the properties provided by the property predictors (see 4.6.2) of each realization to calculate which is the best suited to run at a given time. The utility function was developed from the utility function proposed in [LFS08], and is presented in formula 4.1.

$$(avy + (100 - rut))/2 \tag{4.1}$$

In the formula, avy means availability and rut means resource utilization.
This means that the availability and the (inverse) resource utilization is equally weighted and important, and that a higher availability and lower resource utilization is preferable. This is in line with the definitions of these properties

- a high availability and low resource utilization should indicate a higher chance of utilizing resources well across the devices and getting good performance.

Chapter 5

Implementation

> Writing the first 90 percent of a computer program takes 90 percent of the time. The remaining ten percent also takes 90 percent of the time and the final touches also take 90 percent of the time.
>
> N.J. Rubenking

This chapter contains the details of the implementation part of the project. This includes how the proposed design was implemented to test it, including details on the software, tools and frameworks used for implementation and testing. The chapter also covers the changes and simplifications that were made from the full design for different reasons.

Because of the nature of the project, the design and prestudy phases were the most relevant and effective at answering the research questions. However it is a known fact that what works well on paper doesn't always work well in practice, and since the behavior of the system is dependent on the correct behavior of the utility functions, which is difficult to prove with theory alone, it was decided that a small prototype implementation would be necessary. The prototype was developed as closely to the design as possible, but some simplifications were made due to time limitations and the available resources.

5.1 Software, tools and middleware used

This section contains the tools and middleware used to implement the prototype. This includes what versions are used.

5.1.1 Java

As covered in the Design chapter, Java is the only currently supported language of the MUSIC framework. The latest release available at the time of development, Java SE 6, was used. This release supports all the technical requirements of the design.

5.1.2 OSGi and Equinox

Both the prototype application and the TestEnvironment (see 5.3) uses OSGi™(see 2.5). The prototype is built on the MUSIC framework, which in turn is built on OSGi™. The TestEnvironment uses the the Equinox OSGi™implementation to communicate with the prototype. This was found to be very handy in development of the services of the application.

5.1.3 Eclipse Classic

The most important tool used in the implementation phase was the Eclipse Classic IDE, which includes the Plug-in Development Environment which was used for all implementation. The Eclipse IDE is a much-used Java development environment, and supports all the development requirements of the project. It also includes Equinox, as mentioned above in section 5.1.2.

The current release of the MUSIC framework and much of the developer documentation for it is also streamlined for use of the Eclipse IDE.

5.1.4 MUSIC Framework

As the MUSIC framework is still in development, the latest available release version is 0.3.0. As the version number indicates, this is still an early version, and there is still much functionality proposed in the documentation that is not yet available in the implementation of the framework. For the prototype implementation, the most important such shortcomings are covered in section 5.2.

5.2 Implementation Changes From the Design

As discussed in 5.1.4, the MUSIC framework is currently under development, and not all the proposed functionality is yet supported. This has necessitated that the developed prototype has some reduced functionality, and some parts of the system has been implemented in a simpler version only intended for testing. Some simplifications have also been made because of the limited time available for implementation, so that areas considered unimportant for the primary research areas covered in this report.

5.2.1 Prototype Device

Although the focus of the proposed application is mobile devices, the choice was made early to develop the prototype to run on a typical laptop computer. This choice was made to minimize the time for development and ease the testing. Deciding not to use actual mobile devices meant that typical hardware problems with such devices, such as wireless connection problems, could be avoided during testing.

5.2.2 Browser Proxy

Because of the decision to not develop the prototype on a mobile device, the browser proxy layer was implemented using a standard Java SWING-based GUI rather than using a web browser as indicated in the design. This because the presentation form was not deemed very important to the prototype, and a browser-based user interface would take more time to develop. It could also not be guaranteed that such an interface working properly on a laptop computer would work on the much more limited browser of a mobile device.

5.2.3 Thumbnail Generator

The thumbnail generator of the presentation layer was dropped due to time concerns. It was not deemed important to the prototype, and the thumbnails shown where generated in advance, instead.

5.2.4 Service Hosting

Because the MUSIC framework does not yet support hosting services to other devices, some major simplifications had to be made to the implemented prototype. However, the framework does support consuming services hosted by other applications, so this part of the functionality has been developed as designed with only minor changes. Because the design uses hosting of services for a number of purposes however, some changes had to be made to this part of the prototype.

Hosting Services for Other Users

The design of the system uses services as a way for a resource-strong device to offer functionality to devices with fewer resources, or who are less willing to use them. This includes the Presentation Service (PN) and Content Access Service (CA) services. Since these services cannot yet be advertised and provided to other devices, and using some different form of offering this functionality to other devices is not possible without deviating seriously from the original design, this part of the system has been removed entirely from the prototype.

This is of course an important change, and has some implications. Most importantly, testing context adaptation when devices connect to each other is not possible at the current time. This is the main reason why the Testenvironment (see 5.3) was developed. Because the application can consume services from outside the MUSIC framework with ease, this enables a single device to be tested against this "fake" environment with good results.

The Membership Service (MS) of the Content Repository

Another service that cannot be provided to other devices is the MS service, which is very important for the correct behavior of the Content Repository. Because service hosting is not yet supported in the framework, the prototype cannot provide this service, and is only using the membership service of the Testenvironment (see 5.3). Because of this, new devices joining cannot join the Content Repository Ring through the prototype device, and this is a prominent reason for the changes to this module (see 5.2.5 for more on this).

5.2.5 Fully Distributed Context Repository

The Content Repository module is an important part of the proposed application, and the distributed nature of this repository is an interesting research area in itself. However because the implementation and behavior of this distributed repository is not very central to the research questions of this report, and since the current version framework makes it impossible to implement the Membership Service (MS) properly, this part of the implementation has been severely simplified.

Because the Content Repository cannot be fully implemented, and most of it will reside on the Testenvironment anyway, this part of the system was done in a simplified manner, which eased the testing and lessened development time. In the tested prototype the Testenvironment simplifies the distributed repository so that every node has all the information, and if the prototype device decides to implement it's own local Content Repository it too will have all the pictures used stored locally and just receive the filenames for them. This means the trouble with transfer of files is negated, but since it receives the names of the files from the MS it is believed to be an acceptable simplification.

5.2.6 Resource Context Sensor

The context sensor developed for the prototype was severely simplified. A fully functional sensor, registering for instance the memory, CPU load and battery remaining of the device and reporting this unseen to the user, would be the best way to realize this part of the context in a proper setting. However, as the prototype was developed with only testing in mind, it was deemed too difficult to get the correct resource load for the various tests using this approach. Thus, a much simpler sensor was created in which the tester can choose directly the level of resource utilization he wants to simulate. This proved to be very helpful during the testing. The resource sensor can be seen in figure 5.1.

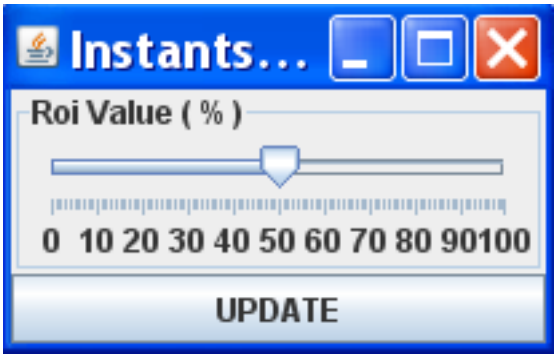

Figure 5.1: The resource utilization sensor

5.3 The TestEnvironment

The TestEnvironment was developed for testing the prototype, because of the limitations and simplifications outlined in 5.2. The Testenvironment is a quite simple Java application, using SLP to advertise and offer the services that a full InstantSocial application would do. Because it's just developed for testing a single device, the TestEnvironment is very simple, and does not have much internal functionality. For the prototype it simulates a single remote device very well, so it is just like connecting to another fully functional device. Figure 5.2 shows the locations and communication of the prototype and TestEnvironment. Note that each TestEnvironment only simulates a single device, for testing with more than one remote device additional instances of the TestEnvironment was run on separate computers.

Unfortunately the TestEnvironment had some difficulties properly transferring the pictures over the services. Although the prototype could perfectly well connect to the services over the network and receive the properties associated with it, the actual transferring of the pictures caused errors on the TestEnvironment. As this transfer used a high number of various components (the MUSIC framework, the Equinox OSGi^{TM}implementation, SLP and more), debugging this error was found to be very time-consuming. As the most important tests, the adaptation tests, could be performed perfectly well without the actual transfer of the pictures, it was decided to not invest more valuable time on this problem. Also, this was an error with the TestEnvironment alone and thus was not seen as a significant problem.

The use of the services itself (for testing the fulfillment of the requirements) was tested on a slightly modified version of the prototype that did not use the TestEnvironment but rather ran the services locally.

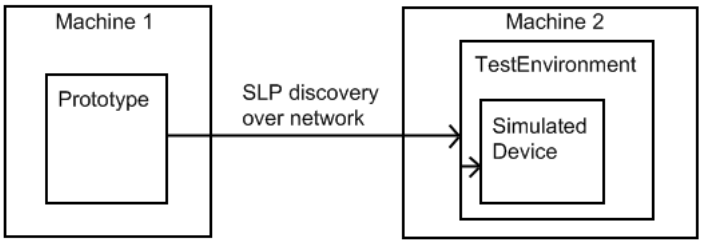

Figure 5.2: Prototype and TestEnvironment communication

Chapter 6

Results and evaluation

> There is no such thing as failure. There are only results
>
> Anthony Robbins

This chapter contains the results of the implementation and design phases. Focus is on the produced prototype and the tests and results of this. Some experiences from the implementation and results from using the design is also presented.

6.1 Prototype

As detailed in section 5.2, the functionality of the implemented prototype has been cut somewhat because of time limitations and the current version of the framework.

The implementation phase of the project has resulted in a functional prototype that can be tested against most of the requirements of the project. However, as is stated in 5.2, the prototype cannot operate on it's own, and requires the Test Environment that was developed for this purpose. Furthermore, an error in the service hosting of the TestEnvironment meant that the prototype presented in this section could not properly access the TestEnvironment to retrieve the pictures. Therefore, the prototype shown here is running the content locally. The prototype version running with the services, which is used in section 6.3, has the exact same modules. Thus this error is simply an error in the TestEnvironment, and is deemed of little importance.

An example of the main interface of the application can be seen in figure 6.1. This screen shows the thumbnails of all pictures available in the currently connected Content Repository ring. A user can click a single image to transfer it to the device and show the full picture.

Figure 6.1: Instant Social Thumbnail Browser

When a user chooses to view the full picture, the screen changes to that seen

in figure 6.2. Only the single picture is shown, and at this point the entire picture has been transferred to the user's device.

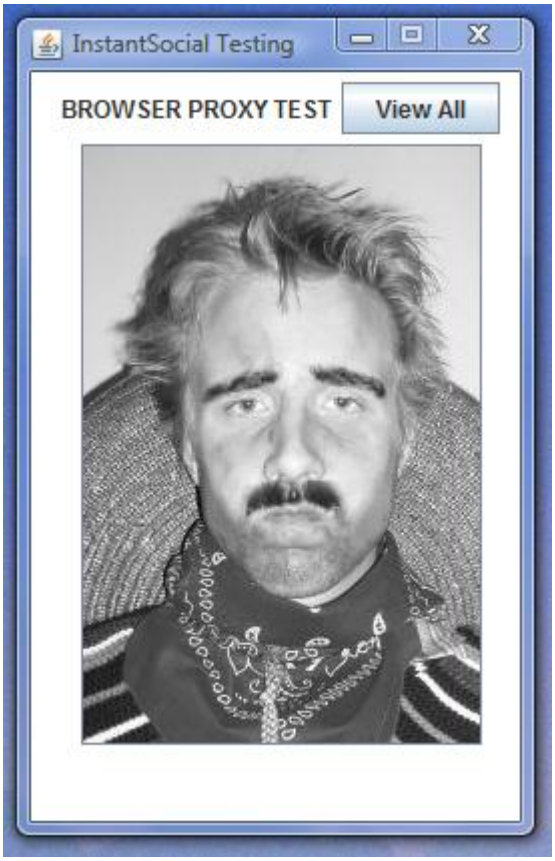

Figure 6.2: Instant Social Picture View

As is obvious from the screen shots the interface is consistent regardless of the current mode of operation of the application. Such internal details should not distract the user too much while using the application.

6.2 Fulfillment of Requirements

This section takes a quick look back at the requirements presented in 4.1, and whether or not the developed prototype fulfills these.

6.2.1 Functional requirements

FR1 The user should be able to set a priority to the application, determining its relative importance to the user compared with other applications running concurrently, and consequently influencing its share of the available resources.
Status: - Partially implemented. This is supported by the MUSIC framework, but due to the limited time available and the version of the framework used, implementation and testing of this was deemed outside of the scope of the prototype.

FR2 The user should be able to join groups based on available groups and his interest.
Status - Not implemented. Due to the limited time available, this was deemed outside of the scope of the prototype. There is only one group available, and the user automatically joins this group.

FR3 The user should be able to browse content available in his joined group(s).
Status - Implemented. Although there is only one group which is automatically selected.

FR4 The user should be able to download selected content from his joined group(s).
Status - Not implemented. Browsing is all that is possible at this time.

FR5 The user should be able to share his content with the joined group(s).
Status - Not implemented. Because of the limitations of the available version of the framework, this was not possible in a proper way.

6.2.2 Non-functional requirements

NFR1 The application should be able to run on typical mobile units. See 4.1.3
Status - Partially implemented. This has not been tested. The prototype was designed to run on a typical laptop for simplification of the testing, but the frameworks used supports running on mobile devices, with the possible exception of the user interface.

NFR2 The application should be fully distributed. No central server should be required.
Status - Partially implemented. Again, this was not possible due to limits in the framework. The design is however fully supporting this.

NFR3 The application should be dynamically self-adaptive to react to the numerous possible changes in the context. See 4.1.3
Status - Implemented. This is proved through the testing in 6.3

NFR4 The application should make use of the MUSIC framework to better enable context-awareness.
Status - Implemented. More on this in 6.5.1.

NFR5 The application should be able to run along with other applications, and in the background.
Status - Implemented. This is supported automatically by the MUSIC framework.

6.3 Context Adaptation Test Results

This section contains the tests that were performed on the context adaptation of the prototype, with the results of each test. As each test is somewhat specialized and limited in size, no good conclusions can be drawn without looking at them all. Such an overall conclusion can be found later (see chapter 7).

Because of a problem with the service hosting of the TestEnvironment, the version of the prototype that performed these tests was not entirely similar in behavior to that which is presented in section 6.1. The prototype version used in these tests had difficulties getting proper access to the content, but this did not impede the tests of the behavior of the context adaptation.

Each test is presented here with a short textual description of their purpose and how they were carried out. The tests were designed to match with the scenario presented in section 2.4.2, or a case of similar nature to the scenario. Each description indicates what part of the scenario is relevant. As each test has a final "shutdown" step and this can be tied to Paul leaving the group, that link to the scenario has been omitted from the description to avoid needless uninteresting repetition.

The setup of the tests can be found in figure 6.3, which shows the various machines, simulated devices and TestEnvironments. Three separate computers were used in the testing, one running the prototype and the two others running TestEnvironments and simulating remote devices. Since the a big point of the context adaptation is that it is invisible to the user, the prototype was configured to print debug information to the console, including every adaptation and change in realization, and this was the basis for the results of the tests.

All start conditions and the context variables of both the local device and remote devices simulated in the test environment is listed for each test, and walk through of steps taken and results. Finally, a short textual summary of the results of each test is presented.

6.3.1 Test 1

The first and most simple test is with a device having much available resources connecting to an environment where there is only one other device to connect to, and this device has limited available resources. This is a basic startup situation, which is similar to the situation in scenario when Paul has a resource-strong device and only Adam is in the group. The devices used and their locations can be seen in figure 6.4

Description

The local resource usage is set to be very small, while the only remote device (Adam) has a large resource usage and low availability. The start conditions of this test can be found in table 6.1. This simple test is only to ensure that the initial context reasoning is correct. The test is terminated after the startup. The steps and results of this test can be found in table 6.2

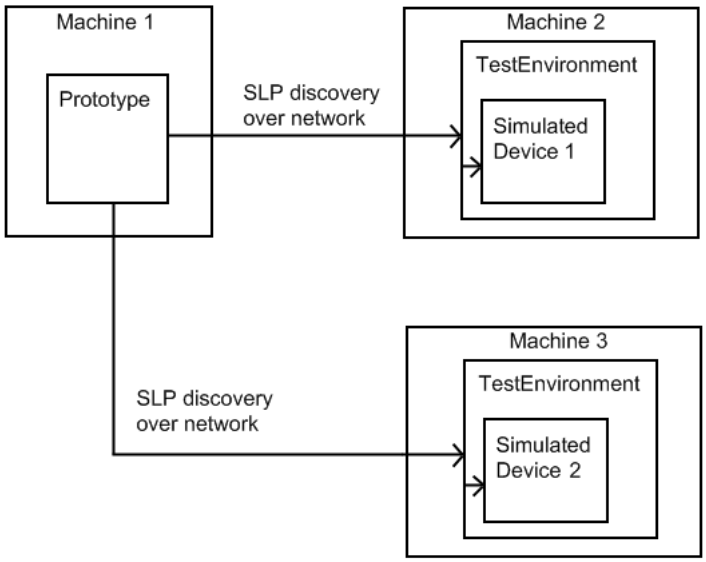

Figure 6.3: Test Setup

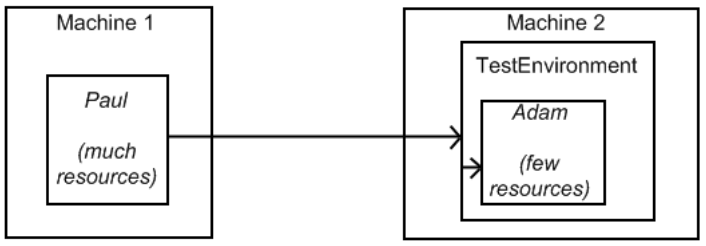

Figure 6.4: Test 1 devices

DEVICE	SETTINGS	DESCRIPTION.
Remote 1 *Adam*	ms.noi = 1 , ms.rut = ca.rut = pn.rut= 90, ca.avy = pn.avy = 10	This is the only remote device, with very limited resources
LOCAL *Paul*	crut = 10	This is the local device, with much available resources

Table 6.1: Test 1 starting conditions

STEP	RESULT.
Start the test environment and prototype with the initial values	Application starts in isFull mode
Terminate the test environment and prototype	Termination successful

Table 6.2: Test 1 Steps and Results

Results

The first test was successful. Because of the properties provided by the remote device and the resource utilization provided by the local context sensor, the utility for the isFull realization is much better than the other realizations, and this is selected.

6.3.2 Test 2

Test 2 is quite similar to Test 1, but the start conditions of the local and remote devices are reversed. This is tied to the situation in the scenario where Paul lowers the priority of the application to listen to music, and Bill's device is the only other remote device left in the group, having much available resources. The devices used and their locations can be seen in figure 6.5

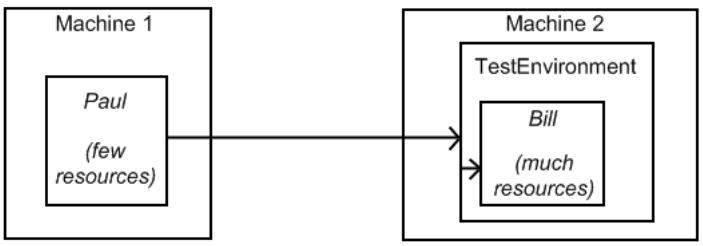

Figure 6.5: Test 2 devices

Although the prototype did not include such user-defined resource usage, it can be simulated by setting a high resource utilization. This results in the device

not being likely to use much more resources, although this "cheat" is not a very good simulation of the situation in the scenario.

Description

The device running the prototype, having very little available resources, connects to an environment where there is only one other device, which has plenty of available resources. The start conditions of this test can be found in table 6.3. This test is, like Test 1, only to ensure that the initial context reasoning is correct and is therefore also terminated after the startup is tested. The steps and results of this test can be found in table 6.4

DEVICE	SETTINGS	DESCRIPTION.
Remote 1 *Bill*	ms.noi = 1 , ms.rut = ca.rut = pn.rut= 10, ca.avy = pn.avy = 90	This is the only remote device, with much available resources
LOCAL *Paul*	crut = 90	This is the local device, with little available resources

Table 6.3: Test 2 starting conditions

STEP	RESULT.
Start the test environment and prototype with the initial values	Application starts in isLeech mode
Terminate the test environment and prototype	Termination successful

Table 6.4: Test 2 Steps and Results

Results

The second test was run successfully. Similarly to the first test, the provided properties and context values makes the isLeech mode the very much preferred one. The middleware correctly chooses this because of the utility function.

6.3.3 Test 3

Test 3 is intended to check whether the reasoner of the prototype can choose correctly between using the different services of a simulated remote device. Not presented directly in the scenario, this is a very important test of the correctness of the adaptation nonetheless. The devices used and their locations can be seen in figure 6.6

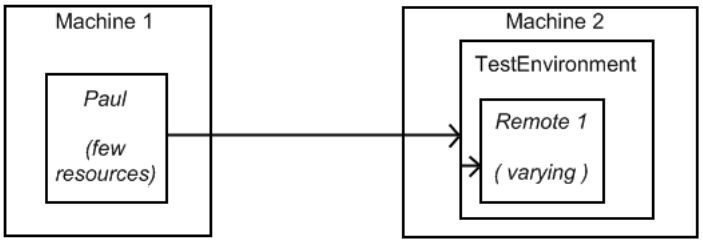

Figure 6.6: Test 3 devices

Description

The local device will have very high resource utilization, so it should choose to use a remote service to provide the needed functionality. The remote device will have differing values for the different services, which should instigate the prototype to choose one of them. During the test, these values will then be changed so that each service in turn should provide the best fit. The initial conditions of the test can be found in table 6.5. The steps and results at each step can be found in table 6.6.

It's worth noting that this test sets the values of the services in a way that would not be possible in a full implementation. This is because the way the context properties for the presentation layer, for instance, is defined. Since this, in a full implementation, returns avy and rut similar to that of the content repository it is connected to, these values would not be different if a full implementation was used. The test environment has no such limitations, however, and allows any values to be assigned as these properties at runtime. This simplifies this test.

DEVICE	SETTINGS	DESCRIPTION.
Remote 1	ms.noi = 1 , ms.rut = 10 ca.rut = pn.rut = 90, ca.avy = pn.avy = 10	This is the only remote device, with much available resources. Initially, the MS service has much lower resource utilization than the other services.
LOCAL	crut = 90	This is the local device, with little available resources

Table 6.5: Test 3 starting conditions

Results

The third test was completed, albeit with some difficulties. In the final test, the TestEnvironment was restarted on each step, each time the properties was

STEP	RESULT.
Start the test environment and prototype with the initial values	Application starts in isFull mode
Stop Remote 1, Set remote 1 values ms.rut = 90, ca.avy = 90, ca.rut = 10, Restart Remote 1	Application changes to isMini mode
Stop Remote 1, Set remote 1 values ca.avy = 10 ca.rut = 90, pn.avy = 90, pn.rut = 10, Restart Remote 1	Application changes to isLeech mode
Terminate the test environment and prototype	Termination successful

Table 6.6: Test 3 Steps and Results

changed on the Remote device. This was not the original plan, but because the TestEnvironment, which was simulating the remote device was somewhat simplified, it had to be restarted when the properties were changed and new services hosted.

With the added restarts of the final test, the context reasoning and adaptation worked very well, performing the anticipated adaptations. The prototype changed for a moment to the isNotAvailable mode while the TestEnvironment was down, as it could not detect any services to utilize.

6.3.4 Test 4

The fourth test is designed to test that the application can properly identify new services as they become available, and properly adapt to this change in the environment. This is tied to the situation in the scenario when Bill joins, but for more thorough testing, this test also includes the first remote device (Adam) joining after Paul. Also, the scenario indicates that Bill's device is a powerful high-end mobile device. To indicate this, the local device has been given medium resources, even though it's running at high priority . The devices used and their locations can be seen in figure 6.7

Description

The prototype is initially started alone, with no services to connect to and a medium resource utilization. Then a device with very limited resources publishes it's services. Finally, a device with much available resources joins and publishes it's services. The initial conditions of the test can be found in table 6.7. The steps and results at each step can be found in table 6.8.

Results

The fourth test was completed successfully. As there are no membership services available at the beginning of the test, the application starts in the

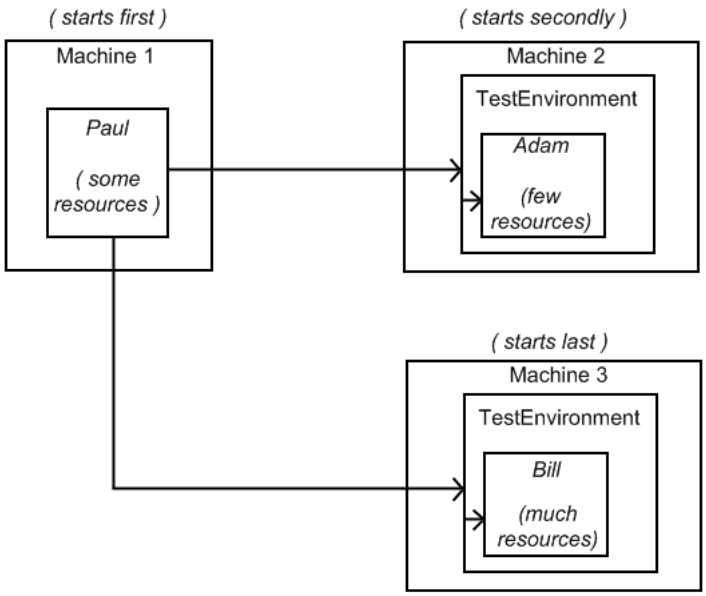

Figure 6.7: Test 4 devices

isNotAvailable mode, which is just a message informing the user that there is no other devices to connect to. When the first device (Adam) becomes available, with limited resources, the device changes to the isFull mode and subscribes to the membership service of this device. When the final device becomes available (Bill), with much free resources, the local device changes to the isLeech mode and connects to this device's presentation service.

6.3.5 Test 5

The fifth test is similar to Test 4, except that it tests if the application can successfully connect to services already running before the application is started. This is tied to the start-up of the application in Paul's scenario, when Adam's device is already present in the group. Again, to make the it more thorough, the test also includes Bill's device already on the group. Thus it can also be tested that the adaptation in this situation is the correct one. Also again, the local resource usage is set to medium to indicate Bill's powerful device. The devices used and their locations can be seen in figure 6.8

Description

The simulated remote devices are started first. One of the devices has little available resources and low availability, the other has much available resources and high availability. After they are fully up and running, the prototype is started. The initial conditions of the test can be found in table 6.9. The steps and results

DEVICE	SETTINGS	DESCRIPTION.
Remote 1 *Adam*	ms.noi = 2 , ms.rut = ca.rut = pn.rut= 90, ca.avy = pn.avy = 10	This is the first remote device, with very limited resources
Remote 2 *Bill*	ms.noi = 2 , ms.rut = ca.rut = pn.rut= 10, ca.avy = pn.avy = 90	This is the second remote device, with much available resources
LOCAL *Paul*	crut = 50	This is the local device, with some available resources

Table 6.7: Test 4 starting conditions

STEP	RESULT.
Start the prototype with the initial values	the prototype starts in the isNotAvailable mode
Start the first remote device with the initial values	prototype switches to the isFull mode (connected to Remote 1)
Start the second remote device with the initial values	prototype switches to the isLeech mode (connected to Remote 2)
Terminate the test environment and prototype	Termination successful

Table 6.8: Test 4 Steps and Results

at each step can be found in table 6.10.

DEVICE	SETTINGS	DESCRIPTION.
Remote 1 *Adam*	ms.noi = 2 , ms.rut = ca.rut = pn.rut= 90, ca.avy = pn.avy = 10	This is the first remote device, with very limited resources
Remote 2 *Bill*	ms.noi = 2 , ms.rut = ca.rut = pn.rut= 10, ca.avy = pn.avy = 90	This is the second remote device, with much available resources
LOCAL *Paul*	crut = 50	This is the local device, with some available resources

Table 6.9: Test 5 starting conditions

Results

The test was a success. The prototype successfully discovers the already running services and connects to the better of the available remote devices.

6.3.6 Test 6

This test is to test if the application can successfully adapt to the situation when a remote device drops out and no longer provide a given service. This is somewhat

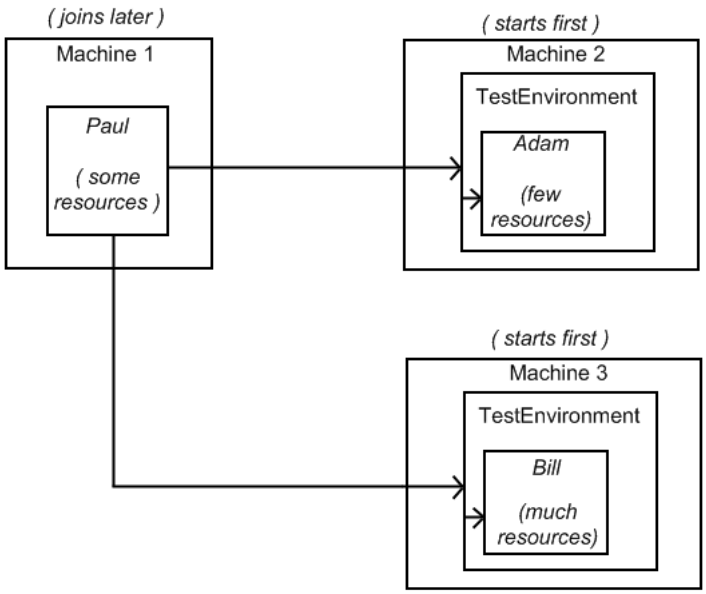

Figure 6.8: Test 5 devices

STEP	RESULT.
Start the remote devices with the initial values	The remote devices start successfully
Start the prototype with the initial values	The application starts in the isLeech mode (connected to Remote 2)
Terminate the test environment and prototype	Termination successful

Table 6.10: Test 5 Steps and Results

tied to the situation in the scenario when Adam quits the group. However, to make the test more thorough, Bill's device is also included in the test, dropping out. The devices used and their locations can be seen in figure 6.9

Description

All the simulated remote devices and the prototype is started at the beginning of the test. When the prototype is running properly remote 2, having the better availability, is shut down. After the prototype has reconfigured, remote 1 is shut down also. The initial conditions of the test can be found in table 6.11. The steps and results at each step can be found in table 6.12.

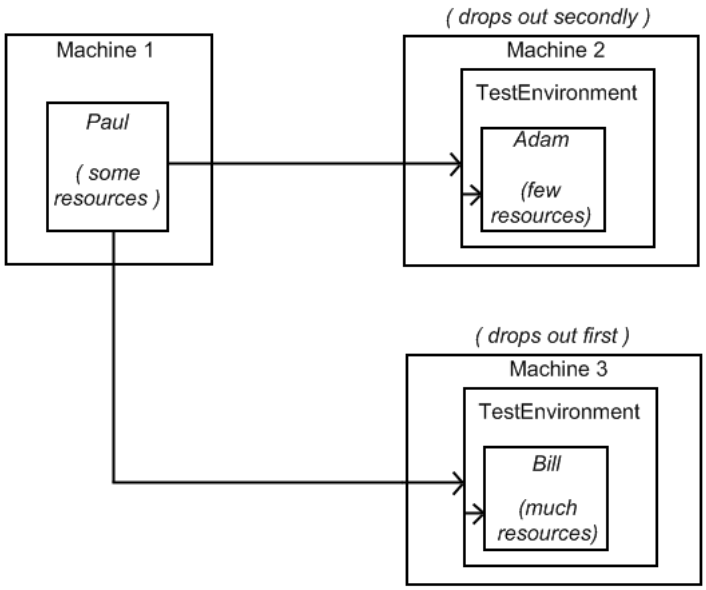

Figure 6.9: Test 6 devices

Results

The test was a success. Although the SLP discovery uses some time to detect that the services are missing (approximately 6 seconds), it does successfully reconfigure the prototype when this is detected.

DEVICE	SETTINGS	DESCRIPTION.
Remote 1 *Adam*	ms.noi = 2 , ms.rut = ca.rut = pn.rut= 90, ca.avy = pn.avy = 10	This is the first remote device, with very limited resources
Remote 2 *Bill*	ms.noi = 2 , ms.rut = ca.rut = pn.rut= 10, ca.avy = pn.avy = 90	This is the second remote device, with much available resources
LOCAL *Paul*	crut = 50	This is the local device, with some available resources

Table 6.11: Test 6 starting conditions

STEP	RESULT.
Start the remote devices and the prototype with the initial values	The prototype starts in the isLeech mode (connected to Remote 2)
Shut down Remote 2	The prototype switches to the isFull mode (connected to Remote 1)
Shut down Remote 1	The prototype changes to the isNotAvailable mode
Terminate the test environment and prototype	Termination successful

Table 6.12: Test 6 Steps and Results

6.4 Evaluation of Design Solution

The design for the application proposed in this thesis was presented in chapter 4. Although some changes were made during the implementation (see section 5.2), these were mainly simplifications due to limits in the available framework, and the overall architectural direction of the design was followed in our prototype.

This tries to evaluate the proposed design, using the research questions presented in section 1.3.

6.4.1 How can the MUSIC framework be utilized for developing social networking in mobile and ad-hoc environments?

The design in chapter 4 provides a possible use of the MUSIC framework for developing such applications. Through the fulfillment of the requirements in section 6.2 and the adaptation tests in 6.3 it is apparent that this approach is appropriate.

6.4.2 What functionality of the MUSIC framework is applicable for developing ad-hoc networking?

The proposed design uses many features of the MUSIC framework, see section 4.3.1 for details on this. Although the framework version available did not provide all the features that are to be implemented the adaptation tests in section 6.3 show that the functionality of the framework that was used, especially the generic middleware-supported adaptation using property predictors and utility functions, was very applicable for such applications.

6.4.3 What architecture is best suited for supporting such an application?

Although the prototype was somewhat limited, the functionality that was included was well-supported by the chosen design. The adaptation tests were successful, with only minor problems experienced. There were no big changes or difficulties because of the proposed design, which was also experienced as quite easy to understand and implement.

6.4.4 What are other typical applications related to ad hoc social networking, and how can the architecture be designed to best separate these parts so that potential reuse is maximized?

The background chapter provides a study of social applications in section 2.4, including an attempt to list common features of such applications. As seen

in section 3.3.1, the choice of InstantSocial as the prototype application was influenced by this. This was to ensure that the prototype developed contained many elements typical to this class of applications.

The top-level architecture of section 4.4.1, which divides the application in layers and communicates between devices using service negotiation is not very specific to the InstantSocial application, but rather provides a base which is fitting with the common features of the social applications. As we have seen that sharing user content is central to such applications, the layers presented could at a high be considered to be applicable to other social applications, if the content shared is changed at a lower level.

It is however dangerous to say this based only on the observations of the design, to say this absolutely it is necessary that this design is tested in developing other social applications.

6.5 Experiences

This section contains the most important experiences and knowledge gained using the chosen tools, frameworks and methods of the project. This will mostly consist of somewhat subjective thoughts on the part of the developers, but relevant examples will be used to underline the important points.

6.5.1 The MUSIC Framework

The choice to use the MUSIC framework was made partially because of the thesis assignment, but also because of the background studies in 2.6. Although it was still somewhat in the middle of development the framework was believed to be able to fulfill the requirements of the prototype, and that the requirements and functionality that was not yet supported could be omitted or simplified in the prototype with little impact on the results.

As is discussed in section 5.2, the changes to the prototype was quite large but not, as anticipated, really critical to the testability or relevance of the prototype. Although the lacking functionality sometimes proved somewhat cumbersome to work around, the end result was felt to simulate the fully-working prototype satisfactory. As an example of this, it took quite a lot of development time to make the fake services and to make sure they were properly published, as this was not yet supported by the framework.

The MUSIC framework uses it's own development method, using model-driven approach extensively. This approach allows a simplified and much faster development, where the developer does not have to worry about the majority of the framework-related code. This code, which would require some quite extensive technical knowledge of the framework, is generated directly from the model using a number of transformations. This approach was found to be a great help in the development when it was working well.

However, not only was the framework, including modeling, tools and the middleware in development and not completed. The documentation as well was in development, and not always completely concurrent with the framework. This presented a number of challenges.

As an example, the correct way of modeling in order for the transformations and code generation to be correct was not yet finalized at the time of the development of the prototype. This meant that there was some difference in the way some examples and such was modeled, and the way they needed to be modeled for the transformation to work. This resulted in some frustrating days of development changing much of the model to another form, so the transformations and code generations would work well.

However, when a consistent set of components and their correct use was found, they were found to be very helpful. As these issues are continuously being worked on by the MUSIC team, they will hopefully be no problems by the time the framework is more mature.

Chapter 7

Conclusion

> "I think and think for months and years. Ninety-nine times, the conclusion is false. The hundredth time I am right." .
>
> Albert Einstein

The goal of this project has been to explore the field of social networking in ad-hoc environments. Through the proposed application and design we have seen how an approach to this can be realized using the MUSIC framework. The design presented is a possible solution to the problems in this field, and tries to leverage the support of the available framework in such a way that not only the proposed application, but possibly also other, similar applications can benefit from it.

The developed prototype has, although very early and simplified, proven through the tests that the context adaptation which has been at the core of this project is solvable using this design approach and the MUSIC framework. And although some problems emerged underway, it has been shown that the proposed application using the presented design is fully capable of adapting to context changes. However, because of the limitations presented, a fully working distributed system could not be developed. Without proper testing in the field, with various devices all reacting to the ever-changing context and each other, total certainty of this cannot be achieved. This is an important continuation of this work.

Much work remains in this field before a fully working application of this type is realized, but the work presented in this report can be an important foundation on which further work in this exiting new field can be made, and might just present a small glimpse of the future of mobile computing.

Chapter 8

Future work

> When it comes to the future, there are three kinds of people: those who let it happen, those who make it happen, and those who wonder what happened.
>
> John M. Richardson, Jr

As this project is still early in development and the prototype developed is a very early one, there is a number of points and challenges ahead before the proposed application is fully completed. With a look back at the tests and design this chapter will outline what is believed to be the most important future work in this field.

8.1 InstantSocial Features

As indicated in section 5.2, the functionality of the implemented prototype was somewhat simplified compared to a fully functional application as envisioned in the scenario. In this section the missing features will be discussed.

8.1.1 Prototype Device and Browser Proxy

As seen in 5.2.1, the prototype was developed for a simple laptop computer, while the application is intended for mobile devices. As the prototype is based on the design, which is aimed at mobile devices, and uses the same frameworks as a mobile application, this should not prove too much of a job. This also means the browser proxy must be changed from the current implementation into an implementation more suitable for mobile devices.

8.1.2 Thumbnail Generator

Because of limited time available, the thumbnail generator was not included in the prototype. It is not a complex module, and a simple version should not prove difficult to implement. However, as is mentioned in 4.4.2, the possibility to have different variations providing different levels of quality and resource demands should be further investigated.

8.1.3 Service Hosting

As indicated in section 5.2.4 the hosting of services was entirely absent from the developed prototype, because this was not yet supported by the framework. This is a very important step in the further development of the application. The current prototype relies on the specially constructed TestEnvironment (see section 5.3) to operate. Realizing the hosting of services for other devices in the application is critical for the proper realization. As the operational field of the described full application is based heavily on ad-hoc approaches, a final application without service hosting or other ways of proper communication with similar devices is of little value.

8.1.4 Fully Distributed Content Repository

Section 5.2.5 showed how the lack of support for service hosting prevented the development of a proper distributed content repository. As this is a central part of the proposed application, it warrants a close look in the further work of this project. The proper completion of this module will likely requires extensive insight and knowledge in the field of distributed computing and storage.

8.1.5 Resource Context Sensor

As discussed in 5.2.6, the implemented resource utilization sensor was designed to allow the tester to set the resource usage directly and does not monitor the actual resource usage of the device. A fully working application will need a resource sensor that is in line with the designed and proposed sensor, reading and mapping the resource usage in a meaningful manner. The work on this includes writing proper resource readers for the important resources of the device, and performing further research and testing of the proper mapping between the various resources and the resource utilization context element.

8.2 The design

As indicated in chapter 4, the design proposed and presented in this report is intended to be a possible solution to applications and problems similar to the InstantSocial case. While the tests and implementation showed that it was appropriate for the developed prototype and problem case, it would require more such problems and prototype implementations to truly prove that it is an applicable approach to a larger set of problems.

8.3 Further Testing

Because of the limited nature of the developed prototype, there is a number of interesting test cases that could not be tested. The most important points for further testing is indicated in section 8.1. With the full features implemented important testing could be done on resource utilization, how a group of devices will operate together and better weighing and insight into the utility functions and property predictors.

Appendix A

Acronym

CA Content Access Service

CDC Connected Device Configuration

GPS Global Positioning System

MS Membership Service

MUSIC Self-adapting applications for Mobile Users In ubiquitous Computing environments

NTNU Norwegian University of Science and Technology

OLED organic light-emitting diode

PN Presentation Service

SLP Service Location Protocol

QoS Quality of Service

References

[All09] Alliance, OSGi: *The osgi architecture*, 2009. http://www.osgi.org/About/WhatIsOSGi, [Online; accessed 13-July-2009].

[BNSW94] Bill N. Schilit, Norman Adams and Roy Want: *Context-aware computing applications*, 1994.

[Boc09] Bock, Laszlo: *Changes to recruiting*, 2009. http://googleblog.blogspot.com/2009/01/changes-to-recruiting.html, [Online; accessed 25-March-2009].

[Coa03] Coates, Tom: *My working definition of social software...*, 2003. http://www.plasticbag.org/archives/2003/05/my_working_definition_of_social_software/, [Online; accessed 25-February-2009].

[CtHS06] Counts, Scott, Henri ter Hofte, and Ian Smith: *Mobile social software: realizing potential, managing risks*. In *CHI '06: CHI '06 extended abstracts on Human factors in computing systems*, pages 1703–1706, New York, NY, USA, 2006. ACM, ISBN 1-59593-298-4.

[DA00] Dey, Anind K. and Gregory D. Abowd: *Towards a better understanding of context and context-awareness*, 2000.

[EGD99] E. Guttman, C. Perkins, J. Veizades and M. Day: *Service location protocol, version 2*, 1999. http://tools.ietf.org/html/rfc2608, [Online; accessed 13-July-2009].

[Fon09a] Fonseca, Rhys Lewis & José Maunel Cantera: *Delivery context ontology*, 2009. http://www.w3.org/TR/2008/WD-dcontology-20080415/, [Online; accessed 28-April-2009].

[Fon09b] Fonseca, Rhys Lewis & José Maunel Cantera: *Delivery context ontology, section: Device hardware*, 2009. http://www.w3.org/TR/dcontology/#DeliveryContext_HardwareEntity, [Online; accessed 27-May-2009].

[FZ94] Forman, George H. and John Zahorjan: *The challenges of mobile computing*. Computer, 27(4):38–47, 1994, ISSN 0018-9162.

[HMPR04] Hevner, Alan R., Salvatore T. March, Jinsoo Park, and Sudha Ram: *Design science in information systems research*. MIS Quarterly, 28(1), 2004. http://dblp.uni-trier.de/rec/bibtex/journals/misq/HevnerMPR04.

[Ita09] Italiana, Hewlet Packard: *Deliverable 4.3 - system design of the music architecture*, 2009. Internal Music Document, accessed 16-June-2009.

[JFG06] Jacqueline Floch, Svein Hallsteinsen, Erlend Stav Eliassen Frank Eliassen Ketil Lund and Eli Gjorven: *Using architecture models for runtime adaptability*. IEEE Softw., 23(2):62–70, 2006, ISSN 0740-7459.

[LFS08] Luis Fraga, Svein Hallsteinsen and Ulrich Scholz: *"instantsocial" - implementing a distributed mobile multi-user application with adaptation middleware*. Electronic Communications of the EASST, 11(12):1–7, 2008, ISSN 1863-2122.

[LY02] Lyytinen, Kalle and Youngjin Yoo: *Introduction*. Commun. ACM, 45(12):62–65, 2002, ISSN 0001-0782.

[Mic03] Microsystems, Sun: *Cdc overview*, 2003. http://java.sun.com/javame/technology/cdc/overview.jsp, [Online; accessed 13-July-2009].

[MW09] Merriam-Webster, Incorporated: *Merriam-webster online*, 2009. http://www.merriam-webster.com/, [Online; accessed 16-June-2009].

[Tea09] Team, Mobile Web Server: *Mobile web server*, 2007 - 2009. https://secure.mymobilesite.net/, [Online; accessed 27-May-2009].

[Tec09] Technologies, Information Society: *Music*, 2009. http://www.ist-music.eu/, [Online; accessed 22-February-2009].

[Tho08] Thomassen, Jan: *Partners in the music project*, 2008. http://www.ist-music.eu/MUSIC/about-music/partners-in-the-music-project, [Online; accessed 22-February-2009].

[Tho09] Thomassen, Jan: *The music project*, 2009. http://www.ist-music.eu/MUSIC/about-music, [Online; accessed 13-July-2009].

[VKV08] Vijay K. Vishnavi, William Kuechler Jr.: *Design Science Research Methods and Patterns*. Auerbach Publications, first edition, 2008, ISBN 978-1420059328.

[Web04] Webb, Matt: *On social software consultancy*, 2004. http://interconnected.org/home/2004/04/28/on_social_software, [Online; accessed 25-February-2009].