NTNU
Innovation and Creativity

Bluetooth in Context acquisition

Henrik Davidsen

Master of Science in Communication Technology
Submission date: July 2006
Supervisor: Per-Oddvar Osland, ITEM
Co-supervisor: Per-Oddvar Osland, Telenor R&D Tyholt

Norwegian University of Science and Technology
Department of Telematics

Problem Description

Context is information that can be used to characterize the situation of an entity.
It may be aquired in many ways, e.g. from user input (status in an IM client), sensed (temperature, location, heart rate), or derived (e.g. combination of location and time of day).
The objective of this assignment is to investigate how a Personal Area Network (PAN) may be used to enrich context information for a person.

The following should be addressed:
- Background studies, literature review
- Describe how a PAN can be used for context harvesting
- Design and, if possible, implementation of the above solution
  For this purpose, Bluetooth is a suitable PAN candidate.

Assignment given: 16. January 2006
Supervisor: Per-Oddvar Osland, ITEM

Bluetooth in context acquisition

Henrik Davidsen

Department of Telematics
Norwegian University of Science and Technology
Trondheim
Norway

July 2006

Preface

This Master's thesis was carried out in the period January to July 2006 at Department of Telematics, at the Norwegian University of Science and Technology.

I would like to thank my advisor Per-Oddvar Osland for his valuable advices and comments during the work on both report and implementation. I would also like to thank Kari Barca Davidsen, Lene Stavang Olsen and Knut Magnus Backer for help reading through the report.

Trondheim, July 2006

Henrik Davidsen, Trondheim July 2006

Contents

List of Figures

List of Tables

List of Abbrevations

3G	-	Third generartion of mobile communication technology
API	-	Application Programming Interface
Bluetooth SDP	-	Bluetooth Service Discovery Protocol
BT	-	Bluetooth
CH	-	Context Harvester
CM	-	Context Manager
CPU	-	Central Processing Unit
CORBA	-	Common Object Request Broker Architecture
DAML+OIL	-	DARPA Agent Markup Language + Ontology Integration Language
DB	-	Data Base
DCOM	-	Distributed Component Object Model
DHCP	-	Dynamic Host Configuration Protocol
EC	-	European Commission
ECG	-	Electrocardiogram
ESDP	-	Extended Service Discovery Profile(Bluetooth)
GPS	-	Global Positioning System
GPRS	-	General Packet Radio Service
GSM	-	Global System for Mobile Communications
GUI	-	Graphical User Interface
HCI	-	Host Controller Interface
IETF	-	Internet Engineering Task Force
IM	-	Instant Messaging
IP	-	Internet Protocol
JCP	-	Java Community Process
JSR	-	Java Specification Request

Abbrevations

J2EE	-	Java Platform, Enterprise Edition
J2ME	-	Java 2 Platform, Micro Edition
J2SE	-	Java Platform, Standard Edition
KVM	-	Java virtual machine technology for embedded devices.
LCD	-	Liquid Crystal Display
MAC	-	Media Access Control address of a networking device
MIDP	-	Mobile Information Device Profile
MSC	-	Message Sequence Chart
MSD	-	Multi-Protocol Service Discovery
.NET	-	Development platform by Microsoft
PAN	-	Personal Area Network
PDA	-	Personal Digital Assistant
PIN	-	Personal identification number
PPP	-	Point-to-Point Protocol
RFID	-	Radio Frequency Identification
RMI	-	Remote Method Invocation
RX power	-	Power level of receive signal, radio
RPC	-	Remote Procedure Call
RSSI	-	Received Signal Strength Indication
RTT	-	Round-trip delay time
RUP	-	Rational Unified Process
SCP	-	Service Control Protocol
SDK	-	Software development kit
SDL	-	Specification and Description Language
SDP	-	Service Discovery Protocol
SIP	-	Session Initiation Protocol
SLP	-	Service Location Protocol
SOAP	-	Simple Object Access Protocol
SP2	-	Service Pack 2 of Microsoft XP
SSDP	-	Simple Service Discovery Protocol
UDDI	-	Universal Description, Discovery, and Integration
UMTS	-	Universal Mobile Telecommunications System
UPnP	-	Universal Plug and Play
URL	-	Uniform Resource Locator
USB	-	Universal Serial Bus
UUID	-	Universally Unique Identifier
UWB	-	Ultra wideband
VB	-	Visual Basic
WLAN	-	Wireless Local Area Network
WS	-	Web Service
WSDL	-	Web Services Description Language
XML	-	Extensible Markup Language

Abstract

PAN, Personal Area Network, may be described as the connection of personal devices, allowing information exchange over short ranges. This term is used on several of todays short range wireless technologies like Bluetooth, ZigBee, Infrared etc. In this thesis we look in to how Bluetooth, as a PAN technology, may be used to enrich context information to a person.

Context is information that can be used to characterize the situation of an entity. It may be acquired in many ways, e.g. from user input (status in an IM client), sensed (temperature, location, heart rate), or derived (e.g. combination of location and time of day).

Several scenarios showing the benefits of context aware computing will be presented, and proposals on how Bluetooth may fit in as an enabling technology will be evaluated.

In the end, a design and implementation showing Bluetooth in context acquisition will be presented.

Chapter 1

Introduction

Today we see an increasing number of embedded devices added to the mobile and computer market. They often have advanced capabilities enabling communication through multiple technologies like WLAN, GSM, 3G, Infrared and Bluetooth. These features may be exploited in what is referred to as context aware pervasive computing.

Pervasive computing is about integrating technology into our everyday life, in a way that hopefully will make our lives easier. A critical element of this work is automated collection of context information and include this in the computational behavior. [1]

Context is any information about your situation. It could be whether you are busy or not, the temperature of the room you are in, the location you are at or which electronic devices surround you. Context aware computing is about using this information when providing, selecting and accessing services.

By structured and intelligent harvesting, distribution and processing of this context information, the software running on your terminal, e.g your cellular phone, can make choices based on your current status without bothering you with any complex configurations. The selection of services offered to your terminal may be carefully picked, based on your interests. And people, including yourself, have the opportunity to investigate your current status before acting, in any kind of session. The context aware application using this information is thereby referred to as a context consumer.

In this Master's thesis we evaluate Bluetooth, and its service discovery features as an enabling technology of context awareness.

The service discovery technologies are crucial to context management systems. They provide functionality to harvest useful context information like nearby devices and services. Bluetooth, being a short range radio technology, is of special interest

because of its capability of providing direct location-based information of nearby devices and services.

Several scenarios and examples are presented trying to illustrate the benefits of this technology domain. In the end, a system design proposal of this technology is modeled, and an implementation of this presented. The implementation illustrates the way Bluetooth may act in a context harvesting system. The system also provides a way to distribute this information in an adaptable way to a large scaled context managing system.

1.1 Motivation

"The human dialog is based on context. The human way of making decisions is based on context. Therefore, it just seems natural to consider using context when building applications that make decisions involving their users." [3] This quoting is from an article on Nokia's website[1], and describes in three simple lines the essentials of context awareness, in development of mobile applications.

In this report we will look into the possibilities of using Bluetooth to realize this context awareness.

To show some of the benefits of context aware computing, we present the following scenarios:

1. Health scenario Tom suffers from a heart disease and his condition needs constant surveillance. Lucky for Tom however, he has a rather advanced cellular phone, vitaPhone[2], constantly measuring his heart activity, ECG. This phone is also equipped with GPS, providing location of Tom wherever he is. We consider information gathered by VitaPhone to be context information of Tom. The situation is described in Figure 1.1. Let's picture a scenario where the heart activity of Tom suddenly changes. Tom does not even have to respond, his change in context is already updated(1) against the health monitoring system which analyzes Toms health parameters and considers his health context history. The Medical Service Center is alerted(2) and gets to analyze his health context, at the same time as an ambulance is on his way(3). The driver knows the position of Tom through his location context. This information is automatically fetched by his car computer and turned to a map position pointing the location of Tom.

[1] www.nokia.com

[2] More information on Vitaphone can be found at http://www.vitaphone.de/en/

Of course, this is just a conceptual scenario, not evaluating security and reliability issues. But we see the value of context, and how a change in context triggers the system to respond, evaluating the full context profile of Tom (including history logged by the system) and responding by activating processes like alerting the Medical Service Center and provide relevant information to the ambulance crew.

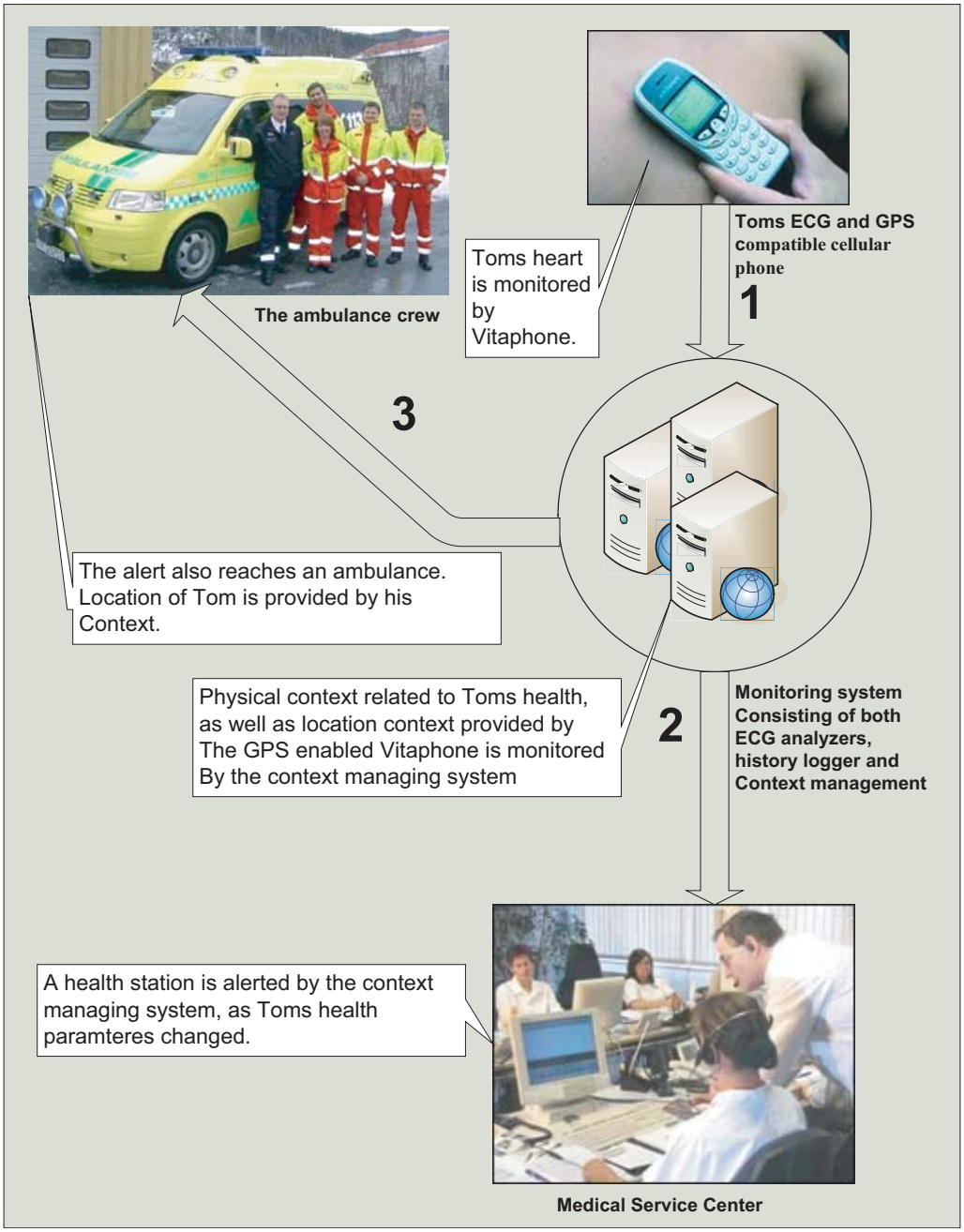

Figure 1.1: Health context aware scenario

Scenario 2 - Airtrain The following scenario is based on Egil C. Østhus' article "Context based session mobility" [2].

Bob is on his way to the airport when a colleague calls him and wants him to participate in a video conference. Because Bob, by the moment he enters the video conference, doesn't have any video equipment available, he joins the conference by voice only, using his 3G cellular phone. When Bob enters the airport, however, his situation has changed. He is still participating in the video conference by voice. At the airport though, there are several video booths[3] available. Due to positioning of Bob's terminal, a centralized system may detect these video booths. This is realized by a mapping of position and devices and services within the zone of this position.

This system also monitors the current session of Bob's, which includes use of video. The application taking care of Bob's video conference session, may alert Bob of this capability and let him switch to video and voice mode.

The centralized system of this scenario is referred to as a context manager and described in section 2.4.2.

We see that based on Bob's context, that is his location and the nearby video booth, he is given new capabilities in his video conference. The scenario is described in Figure 1.2.

[3]A video booth is similar to the well-known phone booths back in the 80ies. But instead of having a phone; it is equipped with a large screen, a loudspeaker system, a microphone, and a video camera. [2]

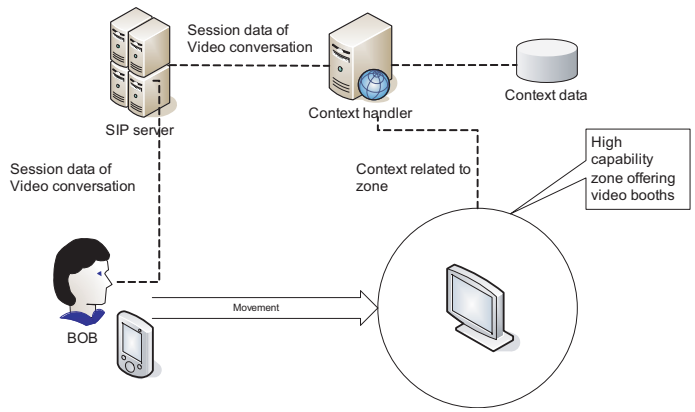

Figure 1.2: Video conference session example. The user moves into video both zone while participating in a video conference. Signaling data of the session is handled by the SIP protocol

1.2 Problem definition

The work of this report is supposed to be a continuation of the project report "Bluetooth and Service discovery" [34] of 2005, and implements the proposed design of a Bluetooth acquiring system. This report describes the details of this process. At the same time, we look further into the theory of Bluetooth, as a PAN technology, in context aware systems.

We look into related work of context aware systems and their context harvesting mechanisms, answering the following:

1. How may a PAN technology, like Bluetooth, be used for context harvesting?

2. How may Bluetooth be implemented for this purpose?

In light of this, we also evaluate how such a solution may fit in to a large scaled context managing system, like the Akogrimo project [13] when it comes to gathering, managing and structuring relevant context information.

1.3 Limitation of scope

In this section we discuss the limitation of scope. We limit the technologies to be discussed as well as the scope of the implementation.

1.3.1 Technologies

In this report we will present several service discovery technologies and discuss their ability to harvest context information. We choose Bluetooth as the technology for our design and implementation, other service discovery are only briefly covered. Bluetooth is chosen because of its appealing features, and great potential in this technology domain - harvesting surrounding device and service context information.

Presenting the enabling technologies we look into local service discovery implementations. Harvesting context information like device and service availability is thereby the main priority in this report, but relating this information to location aspects will also be looked into.

Finally we present a concrete implementation, realizing the concepts Bluetooth in context acquisition. This implementation is a proposal of how a PAN technology may be exploited to harvest context.

1.3.2 Implementation

When it comes to implementation of a service discovery solution, we need to focus on one technology. Again, Bluetooth stands out as the most obvious choice to illustrate the concepts of PAN and context harvesting.

The implementation of this report is to be considered a sub system of a large scaled context management system (described in Fig 2.2). Our design and implementation will cover the context provider of this system.

Our choices of implementation components, will be chosen based on their ability to illustrate how Bluetooth may be exploited to harvest different kinds of context information. Computational context, like information of nearby services will be of main priority, but ways of gathering location data will also be covered.

How the context information is processed, distributed and used by other parts, referring to the center and right circle of Fig 2.2 is to be considered out of scope of this thesis, and will thereby not be covered by this implementation.

1.4 Thesis report overview

This Master's thesis is organized as follows:

Chapter 2, Background: gives the reader some back ground material of this report. This includes explanation of context as an expression, and theory of pervasive computing. In the end, related work to this thesis will be presented.

Chapter 3, PAN and context harvesting: gives the reader an evaluation of how Bluetooth as a PAN technology may be used in context harvesting. Adaption possibility to a large scaled context management system is also described.

Chapter 4, Enabling technologies: introduces some of the front running technologies making service discovery possible. Main focus is on Bluetooth, other technologies are only briefly covered. This chapter also shows how these technologies may fit into pervasive computing in terms of interoperability, semantic discovery and context awareness. We also describe the functionality of web services.

Chapter 5, System Design and Implementation: provides details around the implementation process of a Bluetooth enabled Context Harvester, illustrating some of the concepts evaluated in this thesis.

Chapter 6, Discussion: discusses the evaluation of Bluetooth as a context harvesting technology. We also look in to the choices of implementation made during the work of this Master's thesis. This chapter also discusses how an adaptable transition to a large scale context managing system can be done.

Chapter 7, Conclusion and Future work: discusses the level of achievement relative to the problem statements of the introduction. A conclusion of the report as well as a proposal of future work are also included.

Chapter 2

Background

In this chapter we look in to and evaluate some related work of this Master's thesis. We also discuss and explain some of the common expressions of this technology domain.

2.1 Pervasive computing

"The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it." [5]

This quoting is from Mark Weiser's paper, dated 1991. In this paper he described his visions of ubiquitous computing, now also called pervasive computing. Weiser's visions of an integrated technology, almost invisible to the user, was far ahead of its time. Today, however, this need of making devices automatically work together, exploiting each others resources has become crucial.

The need for adaptable integrating technologies, self configuring devices and far different technologies interacting without bothering the user, naturally follows the rapid increase of mobile computing. All electronic devices capable of doing any kind of communication are intended to be part of this pervasive computing environment. Along with this however, comes new technological as well as ethical challenges.

2.2 Context

Context is a general expression of the kind of data that describes a persons situation and capabilities. It is defined by Dey and Abowd as "any information that can be used to characterize the situation of an entity. An entity is a person, place or object that is considered relevant between the user and the application, including the user and application themselves" [6]. This information may be exploited in what we refer to as context aware applications. "We define applications that use context to provide task-relevant information and/or services to a user to be context-aware". [6]

An example is the well known presence function in Microsoft's Messenger. Anyone in your chatting community is allowed to see your presence status. You may be offline, away or out having lunch,- as long as you have allowed your contacts this information this is considered available context information to them.

Available equipment and services that surround a person are also part of a persons context. This could be a nearby printer, a plasma TV, a high speed network etc. The challenges are many; How can we make the user, or the electronic equipment possessed by the user, aware of the possibilities in his surroundings? In which scenarios may this information be useful? And who should be able to access the context of a user? In this report we concentrate on gathering the context information, evaluating Bluetooth as a context harvester.

2.2.1 Categories

We may categorize context information into different types of information like described in Figure 2.1. This way of categorizing context information is based on Mostefaoui's article on contextualisation of service discovery [7].

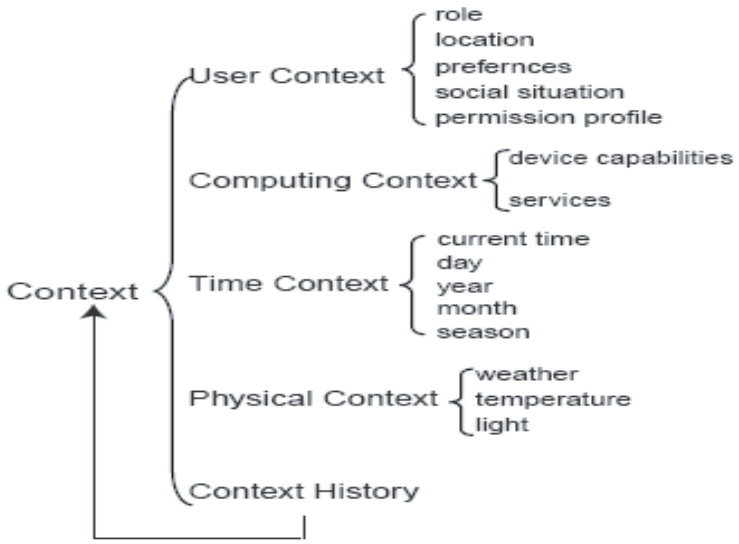

Figure 2.1: Groups of context information [7]

Environment/Physical context: This may be information concerning the physical conditions surrounding the client. These conditions relate to parameters like room temperature, light, noise and air humidity. Gathering this kind of information demands use of different sets of sensors.

User context: This context information represents any kind of dynamic information about a user. This information includes presence and location data, and may be set manually by the user or automatically by trigger mechanisms(e.g, movements of your computer mouse or keyboard activity may influence presence status).

Computing context: This category of context information concerns the availability of devices and services to the user. As a user and his mobile terminal(e.g PDA) moves around, different services will be available, and this availability is intended to be part of the users context. Picture yourself entering the airport lounge. Depending on the network capabilities of your mobile phone, the following devices and services might pop up:

- (Device) A high speed network access point available through WLAN or Bluetooth.
- (Device) Several printers.
- (Device) Several LCD screens.
- (Service) Personal flight information (any information about your flight).
- (Service) Tax free, special offers of the week.
- (Service) Movie streaming.
- (Service) Airport restaurants information.

Time Context Time context relates to any information concerning time aspects, like time of day, week , month and so on.

Context History When all this kind of context, that is time, computing , device etc, is recorded across a time span, we obtain an aspect of context history. In context aware computing this information may be used as an extra parameter.

2.2.2 Motivation

As described in the scenarios of section 1.1, the context information may be used in several ways in what is referred to as context aware computing. One aspect is making things simple, not bothering the user any more than absolute necessary. If you move around, being part of a context aware system, you don't have to bother if your context changes. If this change is critical and needs your involvement you will be alerted. Like the video conference scenario, illustrating enrichment of computational context. The person involved didn't need to think of anything else but the video conference he was participating in (by voice). Context awareness however helped him improve the quality of his session, by investigating the type of session he was in, up against the location he was at, and the devices and services he was surrounded with. In other words, combining all this information, he was given the chance to include picture without interrupting his ongoing conversation.

2.2.3 Context Managing system

To see how the work of this Master's thesis may contribute to a large scaled real life context managing system we provide an overview of the involved parts. In Figure 2.2 a candidate architecture of a Context Managing system is presented. To illustrate the essential part of this system concerning this report, we may divide Figure 2.2 into three parts; the *context provider* to the left, the *context manager* system in the middle and the *context consumer* to the right.

The Context Provider is the part harvesting context information provided to the Context Manager. This information may be gathered by sensors, service discovery protocols or any other kind of technology capable of sensing changes in context.

The Context Manager is the heart of the system. As the name imply, this is where context is managed. The information is stored, distributed, structured and access control maintained. It provides functionality to notify the subscribers of context information/changes.

The Context Consumer relates to the group of entities making use of the context harvested by the Context Provider. Like with the video conversation scenario (section 1.1), the consumer was a context aware system monitoring the context of the user.

This report focuses on the context provider part of the system, concerning harvesting context information. We make use of the Bluetooth technology to see how a short ranged radio technology may contribute in harvesting context.

2.3 Related work

In this section we present some related work to this Master's thesis. We look in to and evaluate methods of harvesting, distributing, deriving, presenting and exploiting context in similar systems to the reference system listed in section 2.4.2. Finally, we describe the reference system of this thesis, the Akogrimo system.

2.3.1 Contextualisation of service discovery

This section is based on Soraya Kouadri Mostefaoui's article "Towards a contextualisation of service discovery and composition for pervasive environments". [7]

The article provides a proposal on how context awareness may be integrated in

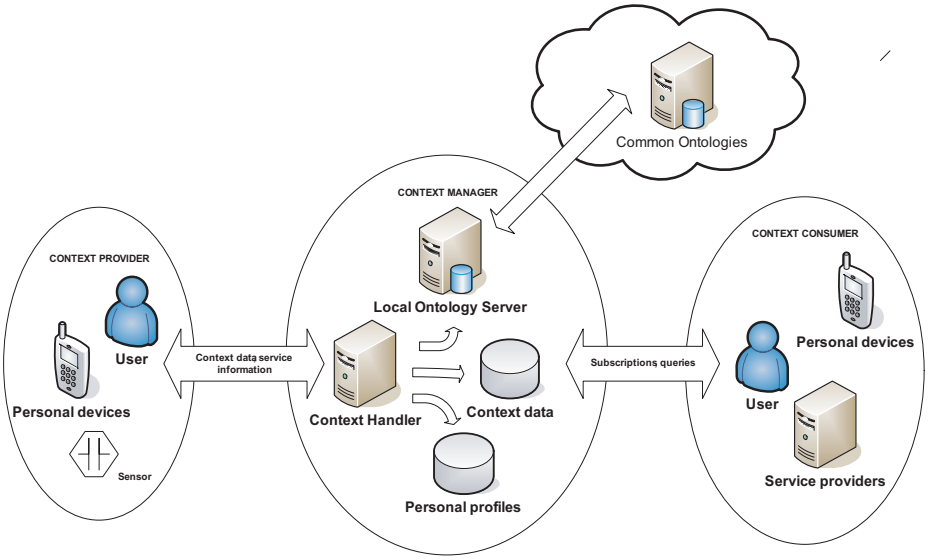

Figure 2.2: Illustrating a context managing system

service discovery technologies. This proposal is based on using agents assisting users to discover and compose services, including context information in this process. By service composition, we refer to the work of combining services intending to provide new more sophisticated services to an end user.

Development of new services based on integrating of existing ones (service composition) is generating considerable interest in recent years in several computer science communities. But, according to Mostefaoui, this work of service composition misses out on the opportunity of exploiting context information. The motivation is therefore to provide a context aware service compositions to an end user.

A proposed layered service architecture is established in order to provide flexibility. A context layer, being one of these layers, consists of a Context Gathering Agent harvesting context information relevant to the services provided. E.g information gathered by sensors, or special messaging between agents of the system.

The compositions and decompositions of services, in order to analyze context relevant to each sub service is done by the work of a Manager Agent and a Service Handler agent.

A scenario illustrating typical context information to be included in a service goes as following:

We define two relevant context primitives of a printing service to be N and L, N being the calculation of the nearest printer relative to the location of the user, L

determining the Load of the printer relative to other printers nearby. By a composition of these parameters we get F as a function of selecting a service based on the context of a user.

In other words, using the location of the user, this service will help him find the nearest and least loaded printer.

2.3.2 SIP as a context delivery enabler

This section is based on Manuel Goert's article "Enhanced SIP Communication Services by Context Sharing" [10].

The article presents an enhancement of the SIP protocol, in order to let it be an enabler of communicating context information. SIP was originally designed for IP telephony as an exchange protocol for session information. The properties of SIP however may very well adapt to be an enabler of efficient context delivery.

The event framework, an extension of the core SIP specification, provides extra functionality in sense of letting applications subscribe on certain events. This property is part of the SIP Instant Messaging and Presence architecture, and obviously well suited to be integrated in a context aware computing domain. When an event occurs, in our case a change in context, subscribers will be notified with a NOTIFY message. In this way the SUBSCRIBE/NOTIFY mechanism may be exploited for transporting context information.

The article describes a proposed Client/Server like architecture making use of this mechanism. A Context Server is defined, managing context information and announcing changes to potential subscribers. Figure 2.3 illustrates this concept.

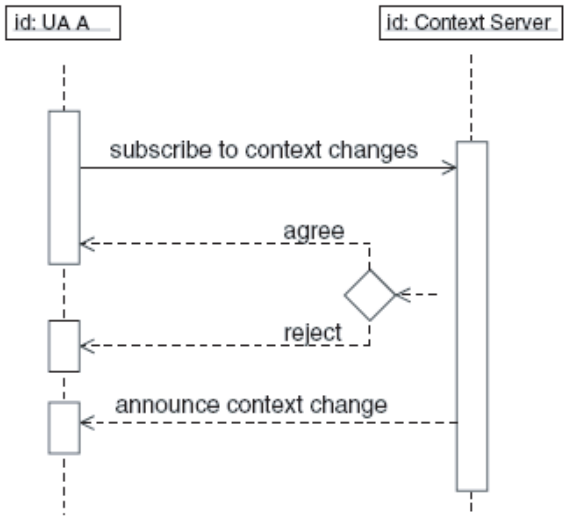

Figure 2.3: Context sharing in Client/Server fashion [10]

2.3.3 A service oriented approach on context awareness

This section is based on Wei Li's article "A service Oriented SIP infrastructure for Adaptive and Context-Aware Wireless Services." [8]

Li looks into the issues on handling context information in a context aware system, making a service oriented approach.

He suggests a layered context stack in order to ease the work of collection, aggregation, interpretation and analyse raw context, generating high level context information. With this, Li proposes a service oriented way of presenting/providing context information, in a way that could ease the adaption of this information to other interested context aware parts.

A separation between direct context acquisition and derived context, i.e a fusion of several context sources presented as one context service, is stressed in the work of defining context information. Li also emphasizes the importance of representing the context acquired in a way that allows interoperability, XML being an obvious candidate on this issue.

In particular he looks into the acquisition of computational context. He proposes a solution on how to harvest this kind of information, concerning ongoing processes of a terminal. He presents a solution on this topic based on a local context agent monitoring changes of running processes. This agent reports changes to the database, while another agent monitoring changes in the database serves as a context source of computational context by monitoring changes in the database.

As a candidate technology of distribution of acquired context Li proposes SIP. By using this protocol a valuable property of session mobility is added, exploiting its capability of "sensing" changes of a session.

2.3.4 SIP and Bluetooth in context awareness

These theories are based on Stefan Berger's article "Ubiquitous Computing Using SIP" [9].

Berger enhances the use of SIP, as an enabler of context delivery, with the use of Bluetooth as a location providing technology. This location, considered location context, is used to derive computational context of the user.

A typical scenario, illustrated in Figure 2.4, goes as follows:

Tommy, as the actor of this scenario, is carrying his PDA when entering a conference

room. Location of Tom is acquired from a local Bluetooth enabled server of the room. This Bluetooth server submits information of the service domain of this zone, also including the room number. A SLP(Service Location Protocol, described in section 4.1) query is sent based on this domain, and information on available services close to Tommy is received. One of the services discovered, a video camera, is registered with a SIP address. In this way Tommy may register this device as a capability of his session and people willing to invite Tommy to conversation sessions has the ability to include video.

We see that Bluetooth, in this case, is used to acquire location context in order to update computational context through SLP service discovery. Once again we see SIP as a crucial part using the acquired context of Tommy to upgrade his capabilities.

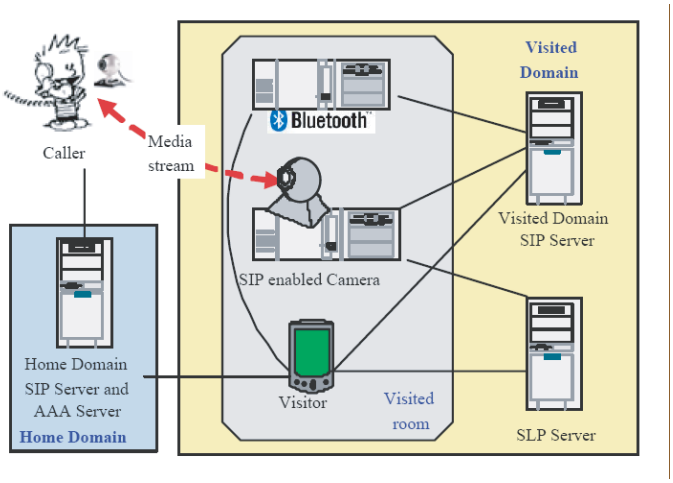

Figure 2.4: Context awareness based on Bluetooth, SLP and SIP [9]

2.3.5 The Akogrimo project

The Akogrimo project is yet another system handling several aspects of context management, including several of the enabling technologies presented in this chapter. Details of this system may be found in section 2.4.2.

2.3.6 Evaluation

Through the related work we see how issues of context harvesting and distribution may be handled. We notice a lot of focus on using a combination of location and

presence when providing computational context. SIP, providing this presence, often act as a context carrier, being a suitable enabler of context distribution.

When handling the actual context we see the value of a layered model, in a service oriented fashion. In this way low level context sources may serve high level context consumers in a structured and well defined way.

The approach of harvesting and distribution of context described in this Master's thesis, however, is somehow different, exploiting the context harvesting capabilities of Bluetooth.

2.4 Akogrimo

Akogrimo[1], Access to Knowledge through the Grid in a mobile World, is a project funded by the EC under the FP6-IST programme. Telenor R&D in Trondheim is one of the 14 partners with responsibility for Network Middelware. In particular, Telenor R&D is developing Context Manager [15] [14], a component enabling realization of context-aware services in the project. The Context Manager is capable of acquiring this information from context harvesting sources, based on technologies like SLP, RFID, and SIP. SLP, Service Location Protocol (see section 4.1) handles service location and discovery of SLP devices on the local network like printers. Context concerning location of the user is based on the RFID technology, while SIP, Session Initiation Protocol, takes care of notifications if any context profile is about to change.

2.4.1 Motivation

We present some of the theories of the Akogrimo project in order to see how a Bluetooth context harvesting entity may fit in to a complete context managing system. Suggestions around this topic may be found in *Adaption possibilities*, section 3.4.

2.4.2 Functionality and Design

The core of the Akogrimo system is the Context Engine (see Fig 2.5) centered in the Context Manager. This is where the context information is processed and organized. As we can see, this engine interfaces three gateways against context sources in order

[1] www.akogrimo.org

to retrieve up to date context information about the customers of the system. The A4C entity handles authentication of the users, referred to as Context consumers. They access the system through the Context Consumer gateway, implemented as a web service. All raw data is stored in the Context database, Context DB.

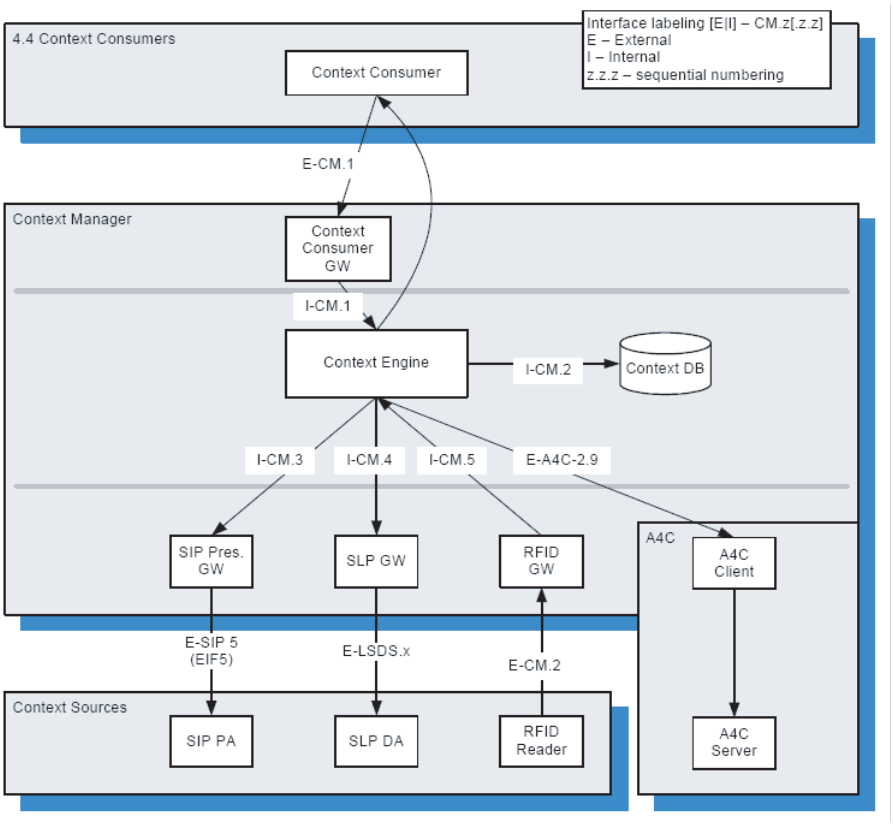

Figure 2.5: The Context Manager design of Akogrimo

Chapter 3

PAN and context harvesting

PAN, Personal Area Network, may be described as the connection of personal devices, allowing information exchange over short ranges. This term is used on several of todays short range wireless technologies like Bluetooth, ZigBee, Infrared etc.

This report concentrates on how to harvest and generate context based on Bluetooth radio technology, and provides a solution on both design and implementation on this topic.

Bluetooth is a short range radio technology, and one of the most promising service discovery technologies in PAN environments. Because of the short range property of Bluetooth, this service discovery technology is well suited to detect and harvest computational context related to nearby devices and also consuming their services.

In this chapter we compare radio based computational context harvesting with centralized computational context generation. We also look into how location data may be provided, and how computing and location context may relate to each other. The Akogrimo system (see section 2.4.2) is used as a reference system, and we evaluate how a Bluetooth solution could contribute as an extra source of context.

3.1 Computational context based on location

In the Akogrimo project (section 2.4.2), computational context is generated by a centralized context manager based on location data from each terminal. We illustrate this with the following scenario, visualized in Fig 3.1

The context management system covers a predefined area. This area consists of several RFID readers. The positions of these readers are mapped to certain SLP

registered services. When Bob, a user of this context managing system, enters one of the zones of these RFID readers, his location is determined by the RFID tag of his personal terminal. Because of the RFID reader mapping to the SLP registered services, the computational context of Bob is enriched.

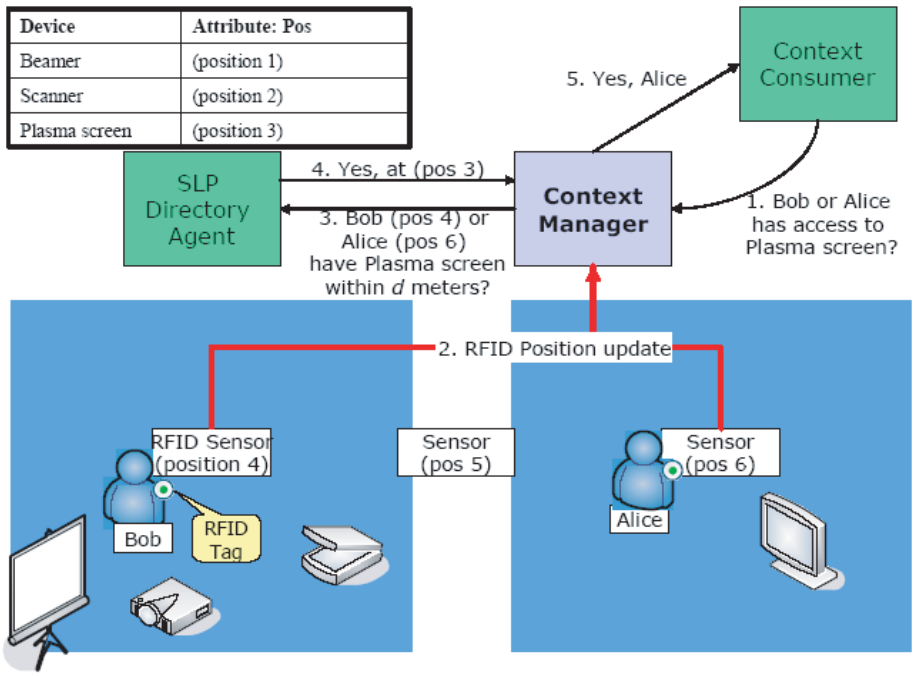

Figure 3.1: Location - device - mapping Example

3.2 Bluetooth and computational context

The system design and implementation of this report show how Bluetooth may be used to enrich computational context, as a parallel to the scenario of the last section.

While the Akogrimo system generates computational context based on location data of each terminal, Bluetooth may be used to harvest this information locally at each terminal, and provide this information to a context manager.

There are several benefits of using Bluetooth in context acquisition, and in particular computational context:

Because of the short range property of Bluetooth, the service discovery process of this radio technology will give a pretty good indication of what is considered nearby to the user.

Second, no location data is needed in order to update context concerning surrounding device and service information. The status of this context information is constantly updated, without any need for extra positioning data. In this way, Bluetooth provides up to date computational context in a mobile environment.

In a large scaled context managing system, this may contribute to an extra source of computational context. With support from other service discovery technologies, like SLP in the Akogrimo system, this solution truly would be an enrichment. As will be described in section 4.7.6, Bluetooth may also provide support to other service discovery technologies, like UPnP and SLP, making this short ranged radio technology an ideal tool in the search for a complete computational picture, regardless of underlying discovery implementation.

The short ranged radio property of Bluetooth, as a mobile context harvester, does have some negative aspects as well though. In some cases, service information beyond the radius of the Bluetooth coverage may be of interests as well. In an ideal world of context aware computing, the radius of what is to be considered relevant information should be decided by the consumer, not by the radio technology doing the harvesting. Another drawback using Bluetooth in context acquisition is actually is mobility(!). As a user walks around in an office environment, his computational context would probably vary a lot. Lets say the user walks with an average speed of 1 m/s, a service discovered by Bluetooth would probably be gone about 20 seconds later. These issues must be handled in a potential implementation of Bluetooth as a context awareness enabler.

Focus of both this report and its implementation have been on acquiring computational context. This information though has little, or none value if the services discovered cannot be consumed by the terminal discovering them. Picture yourself approaching a Bluetooth enabled printer in an office environment; the context managing system responds to this, and refreshes your computational context adding the discovered printing service. If you actually want to use this service, though, the service may be protected by a pin code. This pin code is needed in some cases to pair[1] Bluetooth devices, depending on the implementation of the particular Bluetooth service. In some public environments this authentication procedure might be appreciated in order to limit the service access. In this case, there need to be a way for the service consumer, i.e the terminal, to get these pin codes. A solution proposal on this issue is modeled in Figure 3.2. Whether the user is authorized to see these pin codes or not is determined by the context manager, and hidden in this model. (Details on the context manager implementation can be seen in Figure 2.5)

[1]Pairing of Bluetooth devices involves creation of trust relationship between the devices by generating and storing authentication keys for future device authentication [38]

3.3 Bluetooth and location context

Getting accurate position, both indoor and outdoor, is an important task of a context aware system. Location context is valuable in many ways, as we saw in the service-location mapping of the Akogrimo project. Many technologies provide location service, GPS being one of them. GPS has outstanding features calculating more or less exact position out-door. For in-door positioning we need other ways of locating the user terminals because of lack of sight to GPS satellites. There are many ways to provide indoor positioning, based on different technologies, like WLAN, RFID and Bluetooth.

When choosing positioning technology to a context aware system, it all comes down to the availability of positioning services, and the actual need of position granularity. Because Bluetooth is the chosen technology of this report and design/implementation, we now look into different ways Bluetooth may provide location data as context information.

3.3.1 Bluetooth location by service-location mapping

Similar to the RFID - SLP mapping of location and service in the Akogrimo project, we may adapt this strategy in Bluetooth environments as well. The fact that this equipment has fixed locations may contribute to additional location information for the user carrying the context harvesting terminal. We can describe such a system with the following scenario:

A user enters an office area managed by a context management system, like Akogrimo. This area consists of several preknown devices, like printers, projectors, fax machines etc. The existence of these devices and their locations are frequently, say every month, reported to a central database. This database, being part of the context management system, may then contribute to enhanced information of any device discovered by the users terminal. This scenario is illustrated in Figure 3.2

The detection of a Bluetooth device by itself is not sufficient to determine the location of a user. Several Bluetooth devices may be detected at the same time. These devices may be localized different places, resulting in conflicting area parameters. The solution in this case is making use of radio signal strength and link quality of the Bluetooth connections. In this way, the context managing system has the ability to choose from several potential discovered Bluetooth devices, and select the right one for position computing. That is, link the closest device to an area/position parameter. [32]

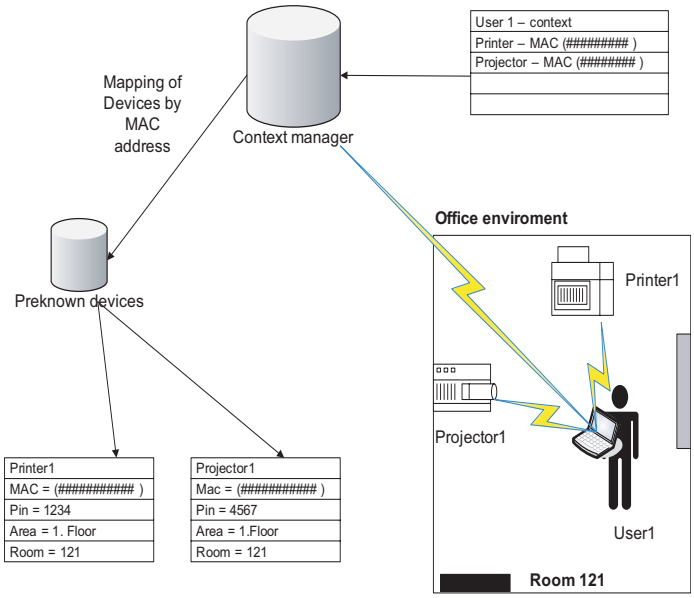

Figure 3.2: Location - device - mapping Example

3.3.2 Bluetooth positioning methods

Most positioning methods involving Bluetooth is based on location knowledge of a number of preknown devices. To get the most accurate position relative to these preknown devices we need to do more than just device discovery. Several devices, holding different positions, may be discovered at the same time. This may indicate conflicting position information. Even if only one device, i.e one position, the discovering terminal may be located anywhere within the radio range of this fixed location device (i.e a radius of 10 meters according to Bluetooth class 3 devices). There are several methods on estimating position with Bluetooth radio. Some of them are presented here, based on Kotanen's article on local positioning with Bluetooth [35]:

Angle of arrival: Angle of arrival, AOA, is based on stationary Bluetooth devices measuring the angle of the arrived signal from a mobile Bluetooth device. If at least two devices perform this measurement, triangulation can be used to estimate the position of the mobile device.

Cell identity: Cell identity, CI, is based on dividing the coverage area into cells, corresponding to the range of a stationary Bluetooth device, e.g 10 m(class 3). When cells overlap, triangulation may be used to improve accuracy.

Time of arrival: Time of arrival, TOA, is based on estimating position using the round-trip time (RTT) of a signal sent from the mobile device to a stationary one. The mobile device measures this time, constructing a circle whose radius corresponds to the half of RTT, the center being the stationary device. This method demands at least three stationary devices, estimating the position of the mobile device to be the intersection of the three circles derived from the RTT to these devices. Obviously, accurate clocks are vital to this approach.

RX power level: RX power level is pretty close to TOA in its way of measuring location. This method also rely on using the intersection of three or more circles, but instead of RTT it uses the power level (RX) of the received signal to estimate the radius of the circles. This measurement is indirectly available to Bluetooth devices through the Received Signal Strength Indication RSSI, available through HCI (Host controller interface, see Fig 4.8).

3.3.3 Bluetooth and GPS

Yet another way to get position information via Bluetooth is consuming GPS services provided by Bluetooth enabled GPS devices. Several of todays GPS equipment includes a Bluetooth interface, making this a perfect location context source to a Bluetooth enabled context harvester.

In context aware computing we may see this as an opportunity as a location context provider. Whether the Bluetooth GPS receivers are placed at fixed positions or following a mobile terminal, other Bluetooth enabled context harvesters may connect and consume this location service.

3.3.4 Hybrid location models

In a large scale context management systems, one might want to be able to estimate location of user both in outside as well as inside areas. In this case, Bluetooth can be used for indoor positioning, while GPS could handle location data outside preknown buildings/areas. To realize this solution the terminal needs both Bluetooth and GPS support. The client side application, concerning location parameters, must have the ability to switch between the two sources of location data retrieval. We might solve this by adding extra information to the indoor areas by letting every boundary room or area get the extra identifier "boundary room". Every time the system detects movements into these zones, the GPS functionality of the terminal is trigged, and

GPS parameters are received. [32] A model illustrating this concept is described in Figure 3.3.

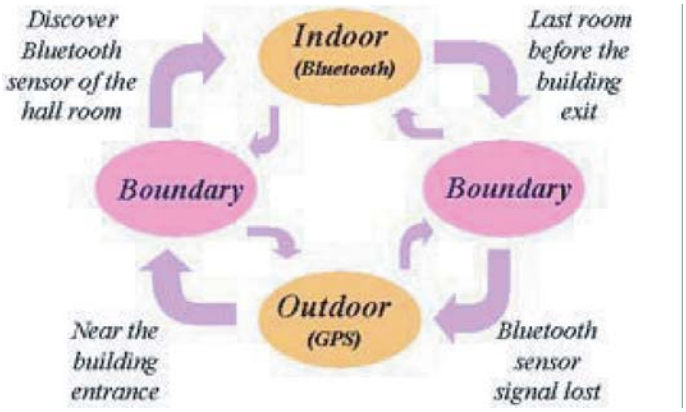

Figure 3.3: Boundary zones for switching of position methods [32]

3.4 Adaption possibilities to Akogrimo

In this Master's thesis we illustrate some of the theories on how Bluetooth may contribute as a context harvester in the Akogrimo (section 2.4.2) project.

The Bluetooth Context Harvester, referred to as CH in the chapter 5 *System design and Implementation*, will fit in to be an extra source of context information concerning surrounding device and service information. In addition, the location information provided by CH, may supplement the current RFID based solution.

CH invokes a web service, in which the host of this service is located at the same domain as the Akogrimo system. That is, the web service acts as a context gateway. The information gathered by the Bluetooth enabled CH refreshes its status via the web service and is structured in XML. An example on this XML file is illustrated in Fig 3.4. This XML file represents device and service context information, as well as GPS position, of one particular customer of the system. The customer is identified by the unique MAC address of his device, tagged as *user name*. We see that user *000f3d38dbf0* of this example is surrounded with 3 different devices offering different sets of services. In addition, it consumes a Bluetooth GPS service, enabling location context as well in the context update.

The information gathered from CH may contribute to an extra entry of context source to the Context Manager CM. (See down to the left of Fig 2.5)

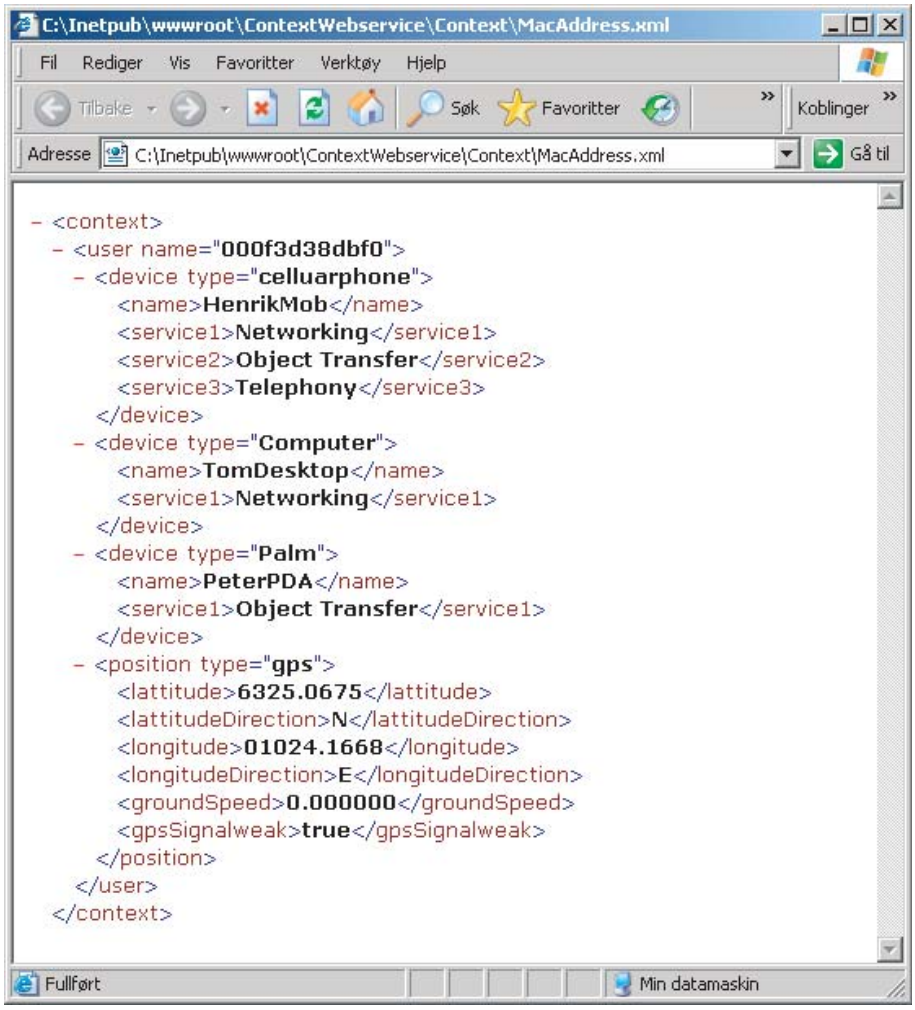

Figure 3.4: XML context structure of Bluetooth devices

3.5 Bluetooth context enriching scenario

To illustrate the concepts of Bluetooth as a context enricher evaluated in this chapter, we present a scenario similar to the one described in section 3.1 where computational context were generated based on RFID positioning mapped to SLP services, all on request from a context consumer.

The scenario of Figure 3.5 sums up the capabilities of Bluetooth as a context harvester. We see a request from a context consumer on the computational as well as location context of user 1,2 and 3. The motivation of this request may be based on the type of session these three actors are participating in, as with the Akogrimo

scenario where availability of LCD screens improved a video conference.

We see that user 1 and 2 are surrounded with Bluetooth enabled devices. Information of these are harvested by the Bluetooth equipped terminal they carry. This information is used by the Context Manager to generate extra computational context of user 1 and 2 based on mapping of preknown devices of room A and B concerning non-Bluetooth enabled devices in these rooms (similar to the RFID - SLP service mapping of Akogrimo). At the same time, Bluetooth positioning techniques are used to compute exact indoor position of 1 and 2. Location information of user 3 is gathered, based on his Bluetooth enabled GPS device. The computational context of user 3, however, is rather poor, but still updated.

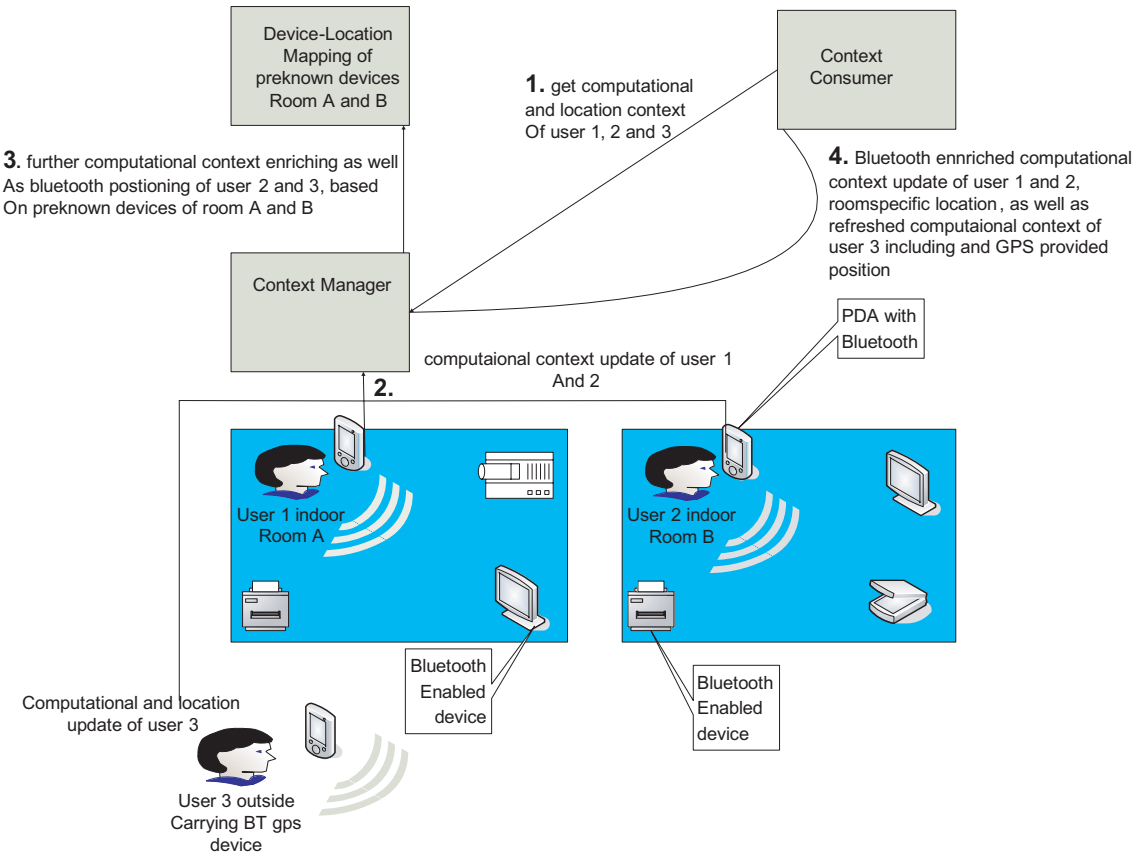

Figure 3.5: Bluetooth harvesting computational as well as location context

Chapter 4

Enabling technologies

This chapter was collected from the project report "Context and Service discovery" [34], though with some modifications related to *The Bluetooth technology*(section 4.7) section, and the added *web services*(section 4.10) section. We include this theory because of its relevance to this Master's thesis.

We give a brief architectural overview, illustrating the essential concepts of service discovery. These are advertisement and discovery of devices and services. Finally we look into how this gathered information can be used in context aware applications. The theory of these technologies is, if nothing else stated, based on Sumi Helal's service discovery paper "Standards for service discovery and delivery" [12] and the "Akogrimo" project report [13].

4.1 SLP

SLP, Service Location Protocol, is a service discovery protocol standardized by IETF,Internet Engineering Task Force. The protocol lets clients running network based applications discover the location of available services in their current network. Service discovery inquiry involves getting information like service address, domain and configuration information. Based on this information the client may connect to the service and use it without any further manual configurations. SLP enables looking for services of a determined type, queries based on service attributes, and requesting service attributes.

4.1.1 Architecture and example

We define three different agents in a SLP discovery domain; User UA, Service SA and Directory DA. A service discovery scenario in this kind of system can be described like this:

1. A SA advertises itself by registering with a DA.

2. The UA sends a service request (on behalf of the user) to the DA.

3. DA browses its registered services, and returns URLs of matching services.

4. UA connects directly to SA, making use of the received URLs.

An illustration of this scenario follows in Figure 4.1.

The UA's discovery of DA and its registered services may operate in two different modes, active and passive. When operating in active mode the UAs and SAs multicast their SLP request to the network, while DA acts as the passive part. In passive mode however, DA is the active one advertising its service periodically to every UA or SA on the network. [11]

In smaller networks use of a Directory Agent might be unnecessary, and the service request will be sent directly to Service Agents in the User Agents domain. Either way the UA will have the opportunity to browse retrieved services and select the preferred one.

4.2 Jini

Jini is a service discovery solution developed by Sun Microsystems, and consists of three protocols; DISCOVERY, JOIN and LOOKUP. The technology is based on the RMI(Remote Method Invocation) concept of the programming language, Java. That is, making devices connect by moving code around the network. Jini makes use of multicast messages, which makes scaling to larger networks difficult. A Jini system is illustrated in Figure 4.2.

4.2.1 Architecture and example

We now describe the architecture of a Jini implemented system, looking into the roles of each of the entities relevant to a service discovery session. The system consists of three actors; A service requesting client, a service provider and a lookup

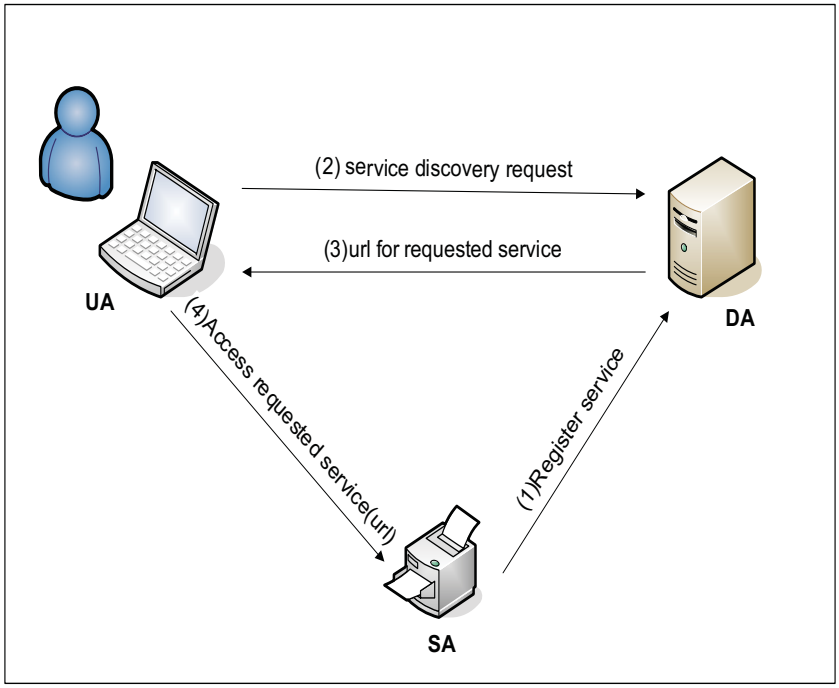

Figure 4.1: Illustrating a typical printing scenario using an SLP Directory Agent

server.

1. A Jini enabled device offering a service, plugs into the network. The device automatically initiates DISCOVERY, trying to locate the lookup service of the current network. When this is found, the device registers with the lookup service making use of JOIN.

2. A client requests a service by specifying the java type or service attributes of the desired service. This request is sent to the lookup service, which returns a service object. The client uses this object to interact with the service directly. The object thereby works as a proxy of the service. This process of locating and invoking the service is provided by LOOKUP.

4.3 UPnP

UPnP, Universal Plug and Play, is a standard developed by Microsoft. It addresses problems like advertisement, discovery and control of devices and services on a network. UPnP lets devices join a network, advertise their capabilities and gain presence and device/service descriptions of other UPnP enabled devices. UPnP

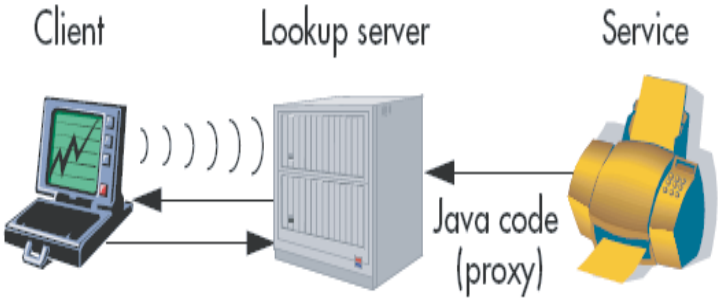

Figure 4.2: Illustrating a Jini system [16]

makes use of SSDP, the simple service discovery protocol, for service discovery and describes device features in XML. UPnP makes use of multicast in this communication consisting XML messages. This traffic is considered heavy, and not suitable to low bandwidth networks. In terms of mobility, however, UPnP provides good support by not letting any unwanted states behind.

4.3.1 Architecture and Example

The use of SSDP in UPnP refers to the trio of protocols in Jini; discover, join and lookup. Service advertisement and discovery, making use of the UPnP standard, is described as follows:

1. A device joins the network by multicasting the SSDP message ssdp:alive.

2. This message is recorded by a control point in the network, or any other present device. The presence of a control point is optional.

3. The control point, if present, advertises itself by multicasting ssdp:discover, and gets unicast response messages from every other device hearing this.

4. The advertise message mentioned in 1, contains an URL pointing to an XML document. This document is the description of the device combined with all information needed to control and make use of the provided service(s). When devices receive advertise messages (URLs), they decide whether the devices/services are relevant to them or not.

Figure 4.3 describes the architecture of an UPnP system. In this figure we see interaction between a client(control point) and an UPnp enabled device providing a service. The Rehydrator converts command information based on the retrieved

XML description, into specific control messages of the service providing device.

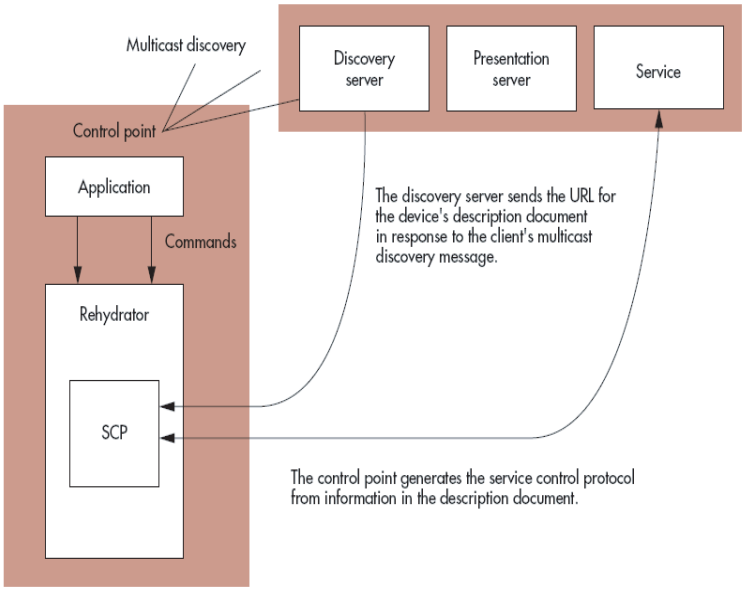

Figure 4.3: Illustrating an UPnP system [16].

4.4 Salutation

Salutation, developed by Salutation Consortium, is another service discovery enabled technology, trying to solve problems with devices and services of dissimilar capabilities. This approach to service discovery makes use of a standardized method for advertising capabilities of applications, services and devices available on a network. Based on this standard set of capabilities, service discovery queries can be made. The use of RPC(Remote Procedure Call) for communication is, however, a problem to small mobile devices of pervasive computing domains due to the heavy need for network resources. A mobile version, Salutation Lite, is intended to solve this problem.

4.4.1 Architecture and example

The Salutation architecture consists of two main entities, the Salutation Manager and Transport Manager. Each client registers with a Salutation Manager, which keeps track of every available service on the current network.

These entities provide service functionality to their clients in the following way:

1. The service provider, e.g a printing device, registers its capabilities with a Salutation manager.

2. A client requests a service, by asking the Salutation Manager. The Salutation Manager of the client could be either local or remote. The client may ask its Salutation Manager to do periodical search for a specific service.

3. The Salutation Managers of the current network synchronize their registered services. They make use of Transport Managers which provide reliable communication independent of the underlying network transport.

4. When the service search is completed by the Salutation Managers, the local Salutation Manager returns the requested service to its client.

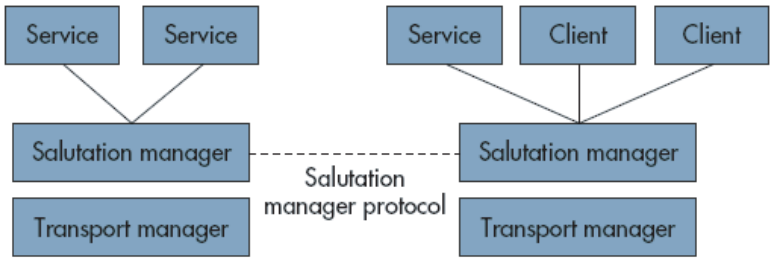

Figure 4.4: Describes a network architecture implementing the Salutation system. We see the Salutation Managers being able to communicate independent of the underlying network. [16]

4.5 Comparison

The table of Fig 4.5 lists the different service discovery technologies discussed so far, and compares their capabilities. This overview is collected from a paper on "Protocols for service discovery in dynamic and mobile networks" [17]. Some of the criteria included here are considered irrelevant to this report, and are therefore not further discussed.

4.6 Multi-protocol Service Discovery

All the service technologies described so far, create a unique basis for context awareness in pervasive computing. The challenge, however, is how to gather all this

	Jini	UPnP	Salutation	SLP
Home Page	www.sun.com/jini	www.upnp.org	www.salutation.org	www.svrloc.org
Main Entities	Lookup Service, Client , Service	Control Point, Devices (Services)	Salutation Manager, Transport Manager, Client, Server	Directory Agent, Service Agent, User Agent
Service Repository	Lookup Service	No	A set of SLMs (Salutation Managers)	DA (Directory Agent)
Service Announcement	Discovery/Join protocol	Advertisement (ssdp:alive)	Registering with local Salutation Manager	Service registration
Service Discovery	Query to lookup service	Contact control point, or listen to advertisement	Query to local SLM and cooperation among SLMs	Contact DA, or multicast to SAs
Access to Service	Service proxy object based on RMI	Invoking action to the service (SOAP), Query for variable state	Service Session management	Service type (service protocol) for the discovered service
Service Description and Scoping	Interface type and attribute matching	Description in XML	FU (Functional Unit) and attributes within it	Service type And attribute matching (fairly powerful matching)
Service Registration Lifetime	Leasing	CACHE-CONTROL header in alive message	No	Lifetime in service registration
Service Group	Group	No	No	Scope
Event Notification	Remote Events	Service publishes event(GENA) when state variable changes	Availability Checking (periodic & automatic)	No
Others	Java-centric architecture	Automatic con-figuration (AutoIP)	Transport independence	Authentication security feature

Figure 4.5: Comparison of service discovery protocols [17]

harvested information and make it available independent of the technology implemented on the clients. Bridging the different technologies by making each and one of them compatible with each other is one approach. In Section 4.7.6 we look into an approach of Bluetooth coexistence with other service discovery technologies.

This section, however, describes another approach of making use of different service technologies located in a heterogeneous network. It is called MSD, Multi-protocol Service Discovery, and provides interoperability of existing service discovery protocols, and propagation of the gathered context information across networks.

The MSD platform consists of the following components: MSD Manager, SD Plugins and MSD Bridges. The MSD Manager is the core of this system, providing context aware service discovery to the clients of the current network as well as remote access information relating to setup information for communicating with the different services.

Each administrative domain, e.g an IP subnet consisting of a set of different devices, elects one MSD manager to control service discovery and make context information available. To make this possible, the MSD relies on its underlying service discovery protocols via the SD plugins. The different kinds of information gathered by a MSD Manager service discovery is structured in a specific format, the MSD service description. The MSD Managers of the different networks cooperate in disseminating harvested information making use of MSD Bridges.

A typical service discovery scenario in a MSD system is described like this;

1. The client specifies the requested service by creating a service description of all relevant service attributes.

2. The MSD manager initiates a service discovery request, exploiting any relevant service discovery plugin. These plugins respond with service descriptions containing the following:

- creation information (used by MSD Manager to learn about its plugins)
- service information (used by clients to learn basic features of service)
- context information (context, e.g environmental information used by MSD Manager)
- propagation information (Forwarding information used by MSD Bridges)
- remote access information (any information related to communicating with the service)

Based on service information collected by these underlying discovery protocols, the MSD Manager responds to its client. Due to the rich service descriptions gathered by the plugins, the manager may aggregate useful context information for semantic service discovery. In Figure 4.6 the concepts of a MSD system is illustrated. [18]

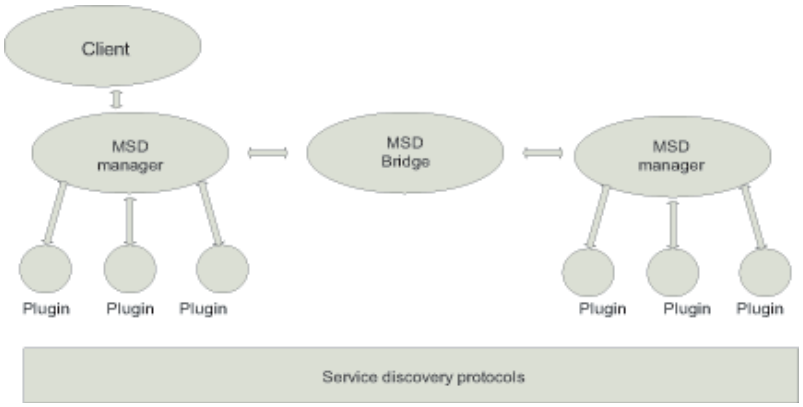

Figure 4.6: MSD system [18]

4.7 The Bluetooth technology

4.7.1 What is Bluetooth

The fact that Bluetooth already is an available technology in most of todays mobile equipment, makes it excellent for service discovery and context aware technologies. We also see an increasing use of Bluetooth as a communication alternative on data peripherals like mice, keyboards, cameras, printers, internet access points etc. This wide range of Bluetooth implementations creates a great potential of generating context information. Unlike the other service discovery technologies presented in this report, Bluetooth provides a fully implemented solution specially designed for mobile clients operating in an ad-hoc environment.

Bluetooth is a low cost wireless technology based on radio. The technology is targeted on low power mobile devices operating in an ad-hoc environment. The communication range is typically ten meters for portable battery-operated devices. The Bluetooth technology operates in the 2.4 GHz band, which is globally available and license free. The intention of the Bluetooth technology was to connect mobile phones to accessories, eliminating the need for cables. This short range wireless technology however, soon enabled several other ranges of use. [23]

4.7.2 Network properties

Analogous to the well known client-server model of a standard computer network, a Bluetooth network is based on a master-slave relationship. One master has the ability to simultaneously connect to seven slaves, forming what is referred to as a *piconet*. Each peer in this piconet however has the ability of connecting to other piconets, creating a *scatternet*. One entity acting as a slave in one piconet may also act as a master in a different piconet. When a master in one piconet becomes a slave in another, it creates a bridge between them. The terminals of a Bluetooth network are able to perform only single hop service discovery, limiting the scaling aspects. The relations of Bluetooth devices in both pico- and scatternets are illustrated in Figure 4.7. [24].

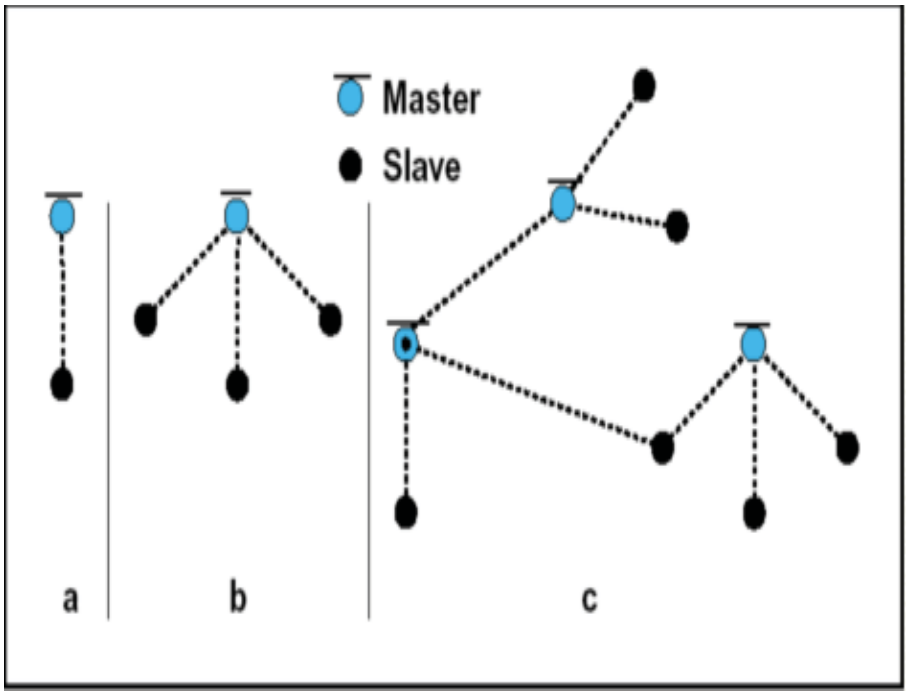

Figure 4.7: Bluetooth network topology: *a* single master slave relation, *b* and *c* master - slave multipoint relation. [25]

4.7.3 Services

Today there are several devices on the electronic market implemented with the Bluetooth technology. That makes this technology well suited for being part of a context aware computing environment. Not only is Bluetooth an excellent way of discovering nearby Bluetooth enabled devices, but it's also capable of carrying multimedia

content like voice. This makes Bluetooth a suitable technology for operating in a context aware computing domain, making use of the service discovery protocol, SDP, to perform discovery and data link establishment for supporting use of the discovered service(s).

Todays Bluetooth enabled devices offer a wide range of services. The most popular of these might be the ability to transmit speech to a Bluetooth handsfree device. Less known is the opportunity to make use of other Bluetooth devices located nearby, like printers , internet accesspoints, loudspeakers etc. Bluetooth also offers a way to emulate communication ports like the well known serial communication ports available on most of todays personal computers. [24]

In this report however, we only look into the possibilities of using Bluetooth as a context harvesting protocol. That is gathering information about nearby devices and services. Later this raw information can be interpreted to context information either by the client itself, or a remote server acting as a context manager.

Each service on a Bluetooth device is represented by a service record. This record holds service attributes, among these the UDDI value of the Bluetooth service. UDDI stands for Universial Unique Identifier, and as the name indicate, uniquely identifies the service in the Bluetooth protocol. Some common UDDI Service Class values of well known Bluetooth services are listed with hex values in Figure 4.1 below [29]: An extended version, including more details and other Device Class UDDIs are listed in Appendix .2.

Service	Value
SerialPort	0x1101
LANAccessUsingPPP	0x1102
A/VRemoteControl	0x110E
Fax	0x1111
Handsfree	0x111E
BasicPrinting	0x1122
HumanInterfaceDevservice	0x1124

Table 4.1: Common Bluetooth UDDIs

4.7.4 Bluetooth protocol stack

The protocol stack of Bluetooth provides several different protocols and APIs for different functionalities. Of special interest in this report is the SDP, Service Discovery Protocol, enabling device and service discovery. Besides service discovery,

Bluetooth also provides functionality as a communication protocol. Links may be set up using L2CAP connections (see Figure 4.8), which may be exploited by high level protocols, e.g other service discovery protocols. This is described in Section 4.7.6.

On top of this stack we see a Java application accessing the functionalities of a Bluetooth radio device through the Java Bluetooth API, JSR-82.The Application and the Java API are present in this figure due to their relevance for the system design in Chapter 5. More details on JSR-82 concerning service discovery may be found in Section 5.4.1. Figure 4.8 shows the different layers of a Bluetooth protocol stack. [26]

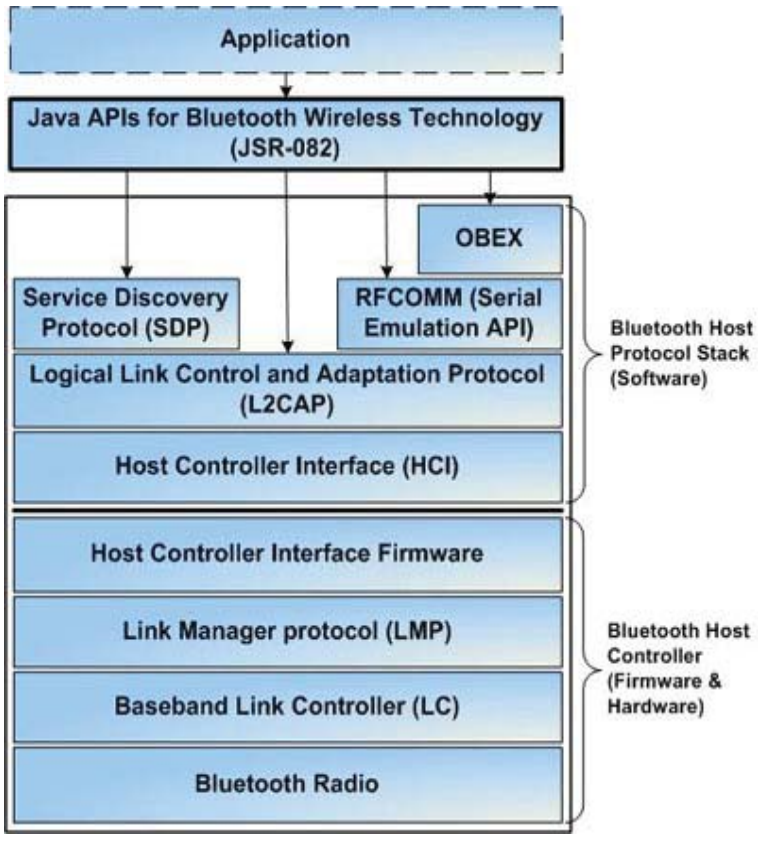

Figure 4.8: Bluetooth application and Java API on top of the Bluetooth protocol stack [26]

4.7.5 Bluetooth service discovery

The motivation for making use of Bluetooth in this setting, is letting a context aware application exploit this wireless technology to harvest valuable context information. The application typically initializes service discovery through its local Bluetooth radio device. Details of this process is found in Section 5.4.1. Performing this discovery, nearby devices and their services will be identified. Special features of the Bluetooth implementation also provide the ability to gather more detailed information about these surrounding services. This information, referred to as service attributes enriches the context profile of the client by adding information like specifications of the service. By requesting extra information, we may also derive other context parameters like location. To illustrate this, picture yourself walking around with a Bluetooth enabled mobile phone in an office environment. All of the sudden, your context aware application senses a printing service and starts gathering information about it. Some of this information may directly or indirectly tell something about the physical location of this service providing device. Due to the short range radio limitation of your Bluetooth device, the location of your service gives a pretty good indication of your own location profile.

Making your context information available to others, and letting people be aware of your capabilities obviously aren't the only motivations for gathering data of your nearby devices. The context information is also supposed to help you or your application making the right decisions. Again think of the printer example of the previous section. Say your client gets to choose from ten different printers. Each of these printers possess different quality of service parameters that need further evaluation. These quality of service parameters include information like potential printing queue, possible paper jam, ability to color print, physical location of the printer etc. Your context available application makes service selection based on all these parameters, making the service selected the most appropriate for you. This was one of several scenarios illustrating the benefits of making service selection based on context information. With the growth of electronic wireless devices surrounding us in addition to the several network solutions available, the need for context aware service selection is likely to grow. [22]

The Norwegian University of Science and Technology, NTNU, currently releases a pilot making a wireless network cover the whole city of Trondheim[1]. This project in co-operation with among others the norwegian telecom company, Telenor, creates a unique platform for wireless services. Expanding the printer selection problem to a selection from hundreds and possible thousands of available services in the same area, certainly puts context aware computing and service selection into perspective.

[1]More information on this project may be found at www.item.ntnu.no/tradlosetrondheim

Even though Bluetooth seems like the perfect service discovery technology, there are some limitations when it comes to being a context harvester in an ad-hoc network. A major drawback is the relatively long(up to ten seconds) inquiry time, that is performing the initial device and service search. This may slow down the performance of a context aware application.

Another disadvantage is that Bluetooth only provides device and service discovery on Bluetooth enabled devices, not enabling service registration. In addition there exists no kind of notification if a service suddenly becomes unavailable.

When it comes to the services provided in a Bluetooth network, every service and service attribute is universally identified with a 128-bit id called UUID. Based on this UUID Bluetooth, applications make service discovery. Being in an area with several services available, this obviously is a major drawback limiting the client only do exact UUID based queries.

To illustrate this problem think of yourself making a service search of luggage stores in a shopping mall. You define the search to match stores that carry 18, 22 and 24 -inch suitcases and you don't want to pay more than 500 dollars. If the service discovery session detects a store carrying 18 and 22 -inches suitcases but not 24 this hit will not be returned. Most of the available services however would not be available at all, due to the inability to register services. An approach to solving the strict UUID based service discovery is presented in Section 4.8. [24] [12] [19]

4.7.6 Compatibility

To succeed in developing a context aware system, exploiting every device and service available, technologies must be able to coexist and cooperate in the same domain. The coexistence relates to both physical communication technologies as well as high layered service discovery protocols. In this section we look into how Bluetooth copes with these issues, and present some research done in the approach to solve them.

Due to the fact that both Wi-Fi[2] and Bluetooth technologies are based on radio communication, making use of the same frequency band, interference may occur and affect the performance. This results in reduced throughput because of packet loss affecting both Bluetooth and Wi-Fi. Locating your Wi-Fi access point nearby any Bluetooth enabled devices is therefore not recommended, if high data rates are desired. The performance of Bluetooth will also vary a lot when several piconets coexist in the same area. Research shows that the probability of no packet collisions

[2]Wi-Fi, or wifi, is a trademark for sets of product compatibility standards for wireless local area networks(WLANs).

in a piconet decreases dramatically as number of coexisting piconets grows. [24]

In a perfect world of pervasive computing we would have one standard service discovery protocol letting every device capable of discovering and exploiting any other device. Unfortunately this is not the fact, and the solution may be to bridge the existing technologies, extracting information specific to each and one of them. Bridging technologies provides interoperability. Without this interoperability, the available services to a client will be limited to its local implementation. Bluetooth is of special interest in these approaches of making different service discovery technologies coexist. This is both due to its wireless capabilities and PPP[3](and thus IP), compatibility. Because Jini, UPnP and SLP all target IP-networks, implementing these technologies on top of Bluetooth is relatively painless. [27]

The Bluetooth SIG[4] defines the Bluetooth Extended Service Discovery Profile, ESDP, targeting the interoperability issue. This profile initially supported extended service discovery for Bluetooth enabled devices in a UPnP service network. By making use of Bluetooth SDP protocol to discover other Bluetooth devices providing UPnP services and retrieve information about these services, service and device context can be harvested beyond the piconet. These services may thereafter be connected to by using either a L2CAP or IP layered connection, both enabled by Bluetooth profiles. [28]

Mapping Salutation on Bluetooth is another proposal in this direction. This may be done in two ways: Either by implementing Salutation on top of Bluetooth, or by replacing Bluetooth SDP entirely with a Salutation solution only making use of Bluetooth as a transport protocol. The latter of these also works with other service discovery implementations like Jini.

And along with interoperability among service discovery technologies comes new and enriched ways of harvesting context information, covering both local as well as remote devices and services reachable to the context aware client. [16]

4.8 Semantic service discovery

When a client is requesting a service in a typical service discovery domain, this request is most often based on preferred service attributes. These attributes form the service description of the provided service. A printer would typically have attributes like *color printing* ability, *serial number* of the printer, *manufacturer* etc. This way of performing service discovery, however, is often inadequate. The different services

[3]PPP: Point-to-Point Protocol
[4]Special Interest Group of Bluetooth

surrounding the clients may describe themselves in different ways, but due to a strict service description request, only a few of the relevant services may be returned. This problem is particularly interesting considering a heterogeneous service discovery network. An example of this problem is given in Section 4.7.5.

To solve this problem, we need to make use of what is known as *semantic service discovery*. The following approach on solving this problem addresses the Bluetooth service discovery limitations due to the UUID based service discovery. This way of coping with inadequate service description based service discovery however, may be used in any other service discovery network, both heterogeneous and homogeneous. It is only a question of implementation.

Semantic service discovery provides a way to add extra information to the service description, increasing the probability of a desired service hit. The extra information includes meta data of the service, like priority data, expected values of attributes and indexes of match closeness- e.g matching two out of three attributes is considered a success. Organizing semantic data relies on a well defined ontology[5]. By making use of an ontology when describing the provided services, the problems with inexact matching of service request and service description decrease. The ontology may be implemented making use of semantic description languages like DAML+OIL - Darpa Agent Markup Language. [19]

4.9 Context aware service discovery

All these service discovery technologies represent methods for gathering information about devices and services in range of a client. This information however is static. By static we mean that the data collected, i.e service descriptions and attributes refer to static variables. Obviously it would be preferable to consider dynamic context information as well in a pervasive computing environment. Services may be down, the load of your surrounding devices may be huge, the location of your preferred service may not be static, the ink level of the printer you want to use may be low. We see that many of these service and device attributes change as time goes by, and we need to make the service discovery technologies support harvesting this dynamic information. We will now look into an approach made to cope with dynamic service discovery in a context aware system making use of an extra dynamic service attribute.

[5]In Information Science, an ontology is the product of an attempt to formulate an exhaustive and rigorous conceptual schema about a domain. This domain does not have to be the complete knowledge of that topic, but purely a domain of interest decided upon by the creator of the ontology [21].

This approach is motivated by the lack of exploiting dynamic context information in todays service discovery technologies. This may be solved by introducing a special context attribute to each service provided in a service discovery domain. This attribute, also referred to as *dynamic attribute*, is a supplement to an ordinary service description only covering static information. The dynamic attribute is determined at the time of lookups, which makes the service information up to date. The idea of making use of this extra set of attributes is tried out on a Jini service discovery system.

The changes made to Jini concerns only the lookup service implementation, leaving the clients unaware of the the extra parameter considered while doing service lookup. When it comes to processing this attribute however, the lookup service might have to involve the service provider. This depends on whether the lookupservice is able to evaluate the context attribute locally, or if a remote service provider needs to fill in on more up-to-date information.

An example of local attribute processing would typically be measuring of network hop counts to a service provider. This process does not involve the remote service object. Service attributes concerning load parameters like size of a printing queue, or current load on a network server however, need involvement of the remote service. In this case the service attribute, located in the service record in the lookupservice, consists of a stub object making a RMI (Remote Method Invocation) call to the remote service object getting the last details of its condition. Figure 4.9 describes the relations of the dynamic attribute in a Jini implementation. The DAttributeL refers to locally managed dynamic attribute, and DAttributeR(Stub) refers to an attribute object that needs remote assistance while being computed. [22]

4.10 Web services

Working with distributed applications, operating in heterogeneous networks, interoperability is a keyword. The web service technology offers this kind of interoperability, among a set of other features. It is built upon a set of standard internet protocols, making the interoperability between different kinds of applications and platforms fairly easy. In this Master's thesis we exploit this flexibility in communicating Bluetooth harvested context, described in chapter 5.

We already have standard communication mechanisms like DCOM(distributed component object model), CORBA(common object request broke architecture) and RMI(RMI Remote method Invoke). These are all tight coupled communication protocols aimed at local networks. The loose coupled web service technology adds scalability.

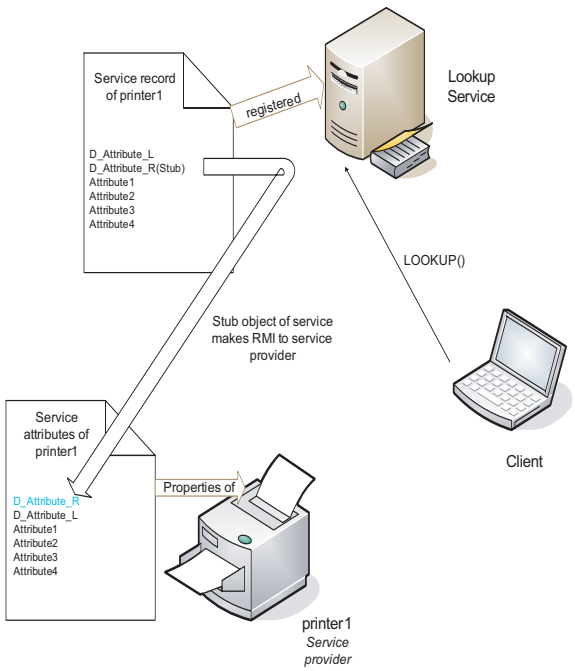

Figure 4.9: Implementation of dynamic attributes in a Jini system

4.10.1 Architecture

A web service is a software component made available by standard network protocols like SOAP and HTTP. The web service itself is described through the WSDL file. Based on this WSDL clients may invoke the methods provided by the web service by generating proxyobjects. Service providers may register their services through UDDI, a XML-based protocol that provides a distributed directory. In this way clients have the ability to discover available services and make bindings to them. A classic web service discovery and service binding scenario is described in Fig 4.10.

The Provider, in Fig 4.10 builds a web service and makes it available through an UDDI registry. A client makes a service search in this registry, makes contact with the service provider and retrieves the WSDL document. The WSDL is the foundation of the proxyobject generating process. The client makes method invocation directly on this proxyobject, which in turn translates this request into SOAP messages. These SOAP messages are based on XML data structures, again contributing to interoperability. The SOAP messages are typically sent over HTTP. A SOAP server, localized at the service provider, translates the XML structure of the SOAP message into the language of the actual service. E.g a Java implemented client may invoke methods of a VB .NET web service without problems. [33]

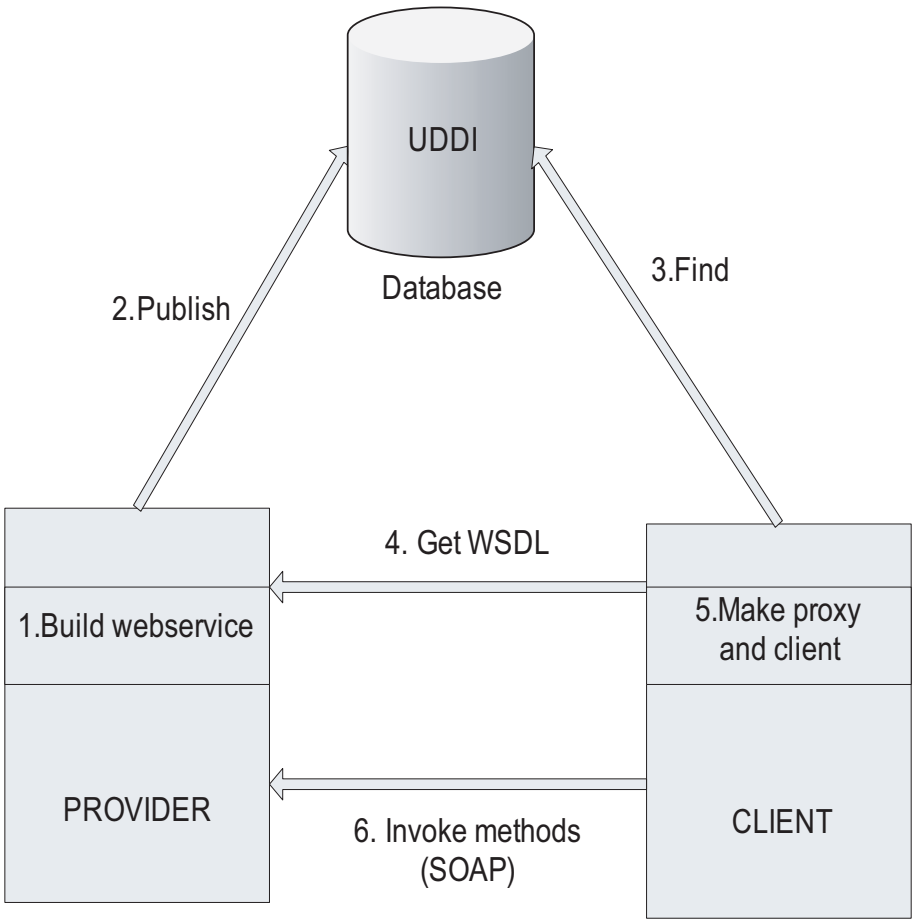

Figure 4.10: A typical web service scenario

Chapter 5

System Design and Implementation

This chapter describes the details of the system development process based on the system design of this report. The implementation is considered a subsystem of Figure 2.2 focusing on context harvesting. This process relates to the *Context Provider* circle and its context sources. In this case the source of context will be a Bluetooth enabled application harvesting information of as well as consuming nearby Bluetooth services, in order to provide this information to a context managing system(relating to the middle circle of Figure 2.2)

Developing a complex communication system like this, requires a structured process. This process involves different stages like identifying the functional requirements of the system, designing system architecture, computational models describing behaviour and states of the system, as well as implementation and testing.

5.1 Method

The two main approaches to this process of system development are the Waterfall and the Iteration method[1]. In this report we choose Iteration, based on the characteristics of the project; This project is a continuation of the system design project of 2005, *Context and Service Discovery* [34]. In this project we completed the first iteration of a System development process, referred to as Inception phase(Figure 5.1). We then identified the basic behaviour of the system. This behavior included message interaction between system entities and activity diagrams of the basic processes.

[1]The Waterfall and Iteration methods are described in [30]

5.1. METHOD

We proceed this work by running new iterations of system development. Working by iterations, we have the opportunity to monitor and modify the system during the process, by making each iteration an improvement of the previous, both on design and implementation details.

RUP, Rational Unified Process [31], is a well known software engineering process framework. It represents an iterative way of system development, dividing the workflow into predefined stages. This process consists of four phases; Inception, Elaboration, Construction and Transition.

Even though the size of the implementation presented in this chapter might be small to fit into this large scale project framework, we try to follow the steps of this process to some extent.

This Master's thesis runs new iterations of the Elaboration and Construction phase. Not all parts of RUP are equally relevant nor manageable in a project like this thesis. We have chosen only the artefacts needed to describe the implementation process of this project, including diagrams illustrating requirements, behaviour and implementation architecture. Based on the stated requirements of the system (section 5.2), we produce the most relevant use cases. SDL description language [39], is preferred in modeling behaviour, and basic processes of the system are present in section 5.3.1. Package and class overview are presented, in order to illustrate the implementation architecture. Finally, we provide several test cases, motivated by and verifying the requirements first stated.

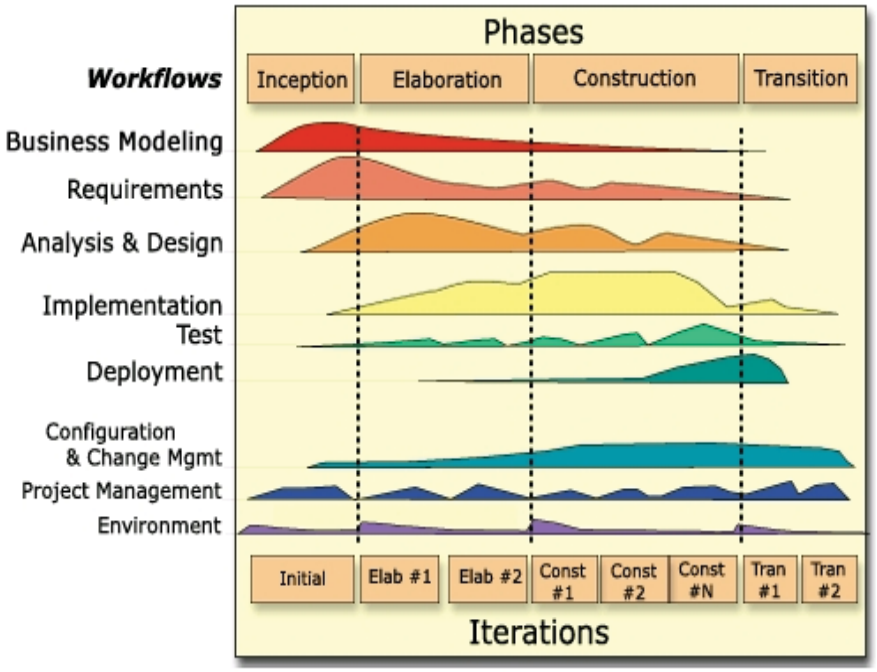

Figure 5.1: The work flow of a RUP process

5.2 Functional requirements

This Master's thesis evaluate how Bluetooth as a PAN technology may be used to harvest context information. It provides the practical approach on this, presenting design and implementation of a Bluetooth context harvester interfacing a context managing system via web service method invocation.

Based on the scenario presented in section 3.5 we now state the functional requirements of the implementation of this thesis. From this scenario we chose to implement a solution that could harvest computational context based on Bluetooth enabled surrounding devices, as well as consuming a Bluetooth GPS service in order to include location context, and provide this to a Context Manager system. In this way, this implementation provides a subsystem to a context managing system similar to the one provided in the Akogrimo project. The main goals of this implementation is thereby to harvest, structure and provide context information via Bluetooth to a context managing system. To realize this we define the following entities (implementation of CM is not part of this master thesis, but included in this list to relate the implementation to a large scaled context managing system):

- CH , Context Harvester, the Bluetooth enabled application harvesting context.

- GPSservice, Bluetooth GPS service consumed by CH.
- CM, Context Manager receiving context updates from CH through a web service gateway.
- WS, The web service acting as a gateway of context from CH to CM.
- WSproxy, web service proxy representing the web service interface of CM at CH.

Number	*Functional requirements CH*	*Priority*
R1	perform device discovery.Frequency configurable .	High
R2	do service discovery for every device found.	High
R3	generate context readable (XML)to a managing system, CM.	High
R4	update CM within 10sec(configurable),if changes appears.	Medium
R5	include GPS based position data in context updates.	Medium
R6	provide a way to let the user validate the position submitted to CM.	Medium

Number	*Functional requirements WS*	*Priority*
W1	provide upload context methods to CH.	High
W2	acknowledge to CH on any request.	High
W3	store context from CH structured as XML .	High
W4	be able to distinguish context updates coming from different terminals.	High

5.2.1 Use cases

Trying to visualize these functional requirements, we now describe a selection of relevant use cases. These are listed and described in Figure 5.2, Figure 5.3 and Figure 5.4.

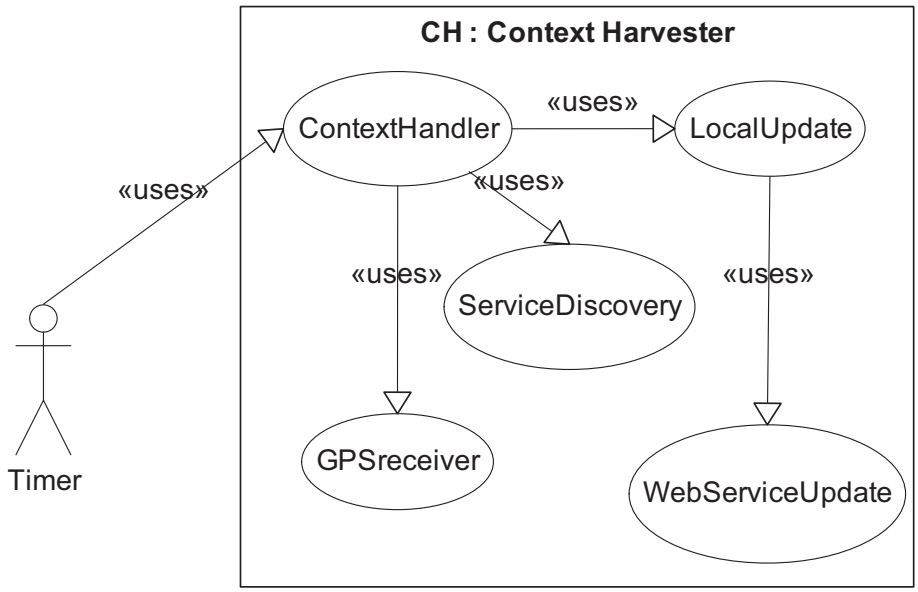

Figure 5.2: Use case Context Harvester CH

Use case	
Use case name	CH Context Harvester
Actors	Timer
Preconditions	Application idle
Postconditions	Service discovery running, GPS signal received, context frequently updated, Web service gets updates
Normal flow	Device and services discovered, GPS signal constantly received, context formatting, update via WebServiceUpdate
Includes	ContextHandler, LocalUpdate, ServiceDiscovery, WebServiceUpdate, GPSreceiver

Table 5.1: Use case Context Harvester CH

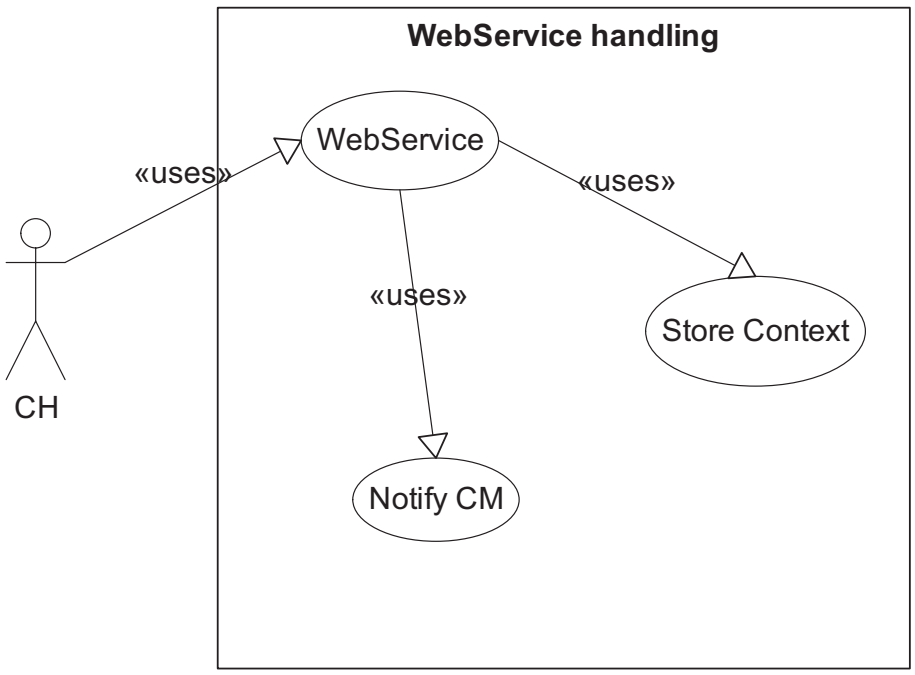

Figure 5.3: Web service handling context from CH

Use case	
Use case name	WebService handling
Actors	CH
Preconditions	Change in context at ContextHandler
Postconditions	Context updated at Webservice, XML generated
Normal flow	Receive Context information by XML-string, store context
Includes	WebService, NotifyRemote, Store Context

Table 5.2: Use case WebService handling

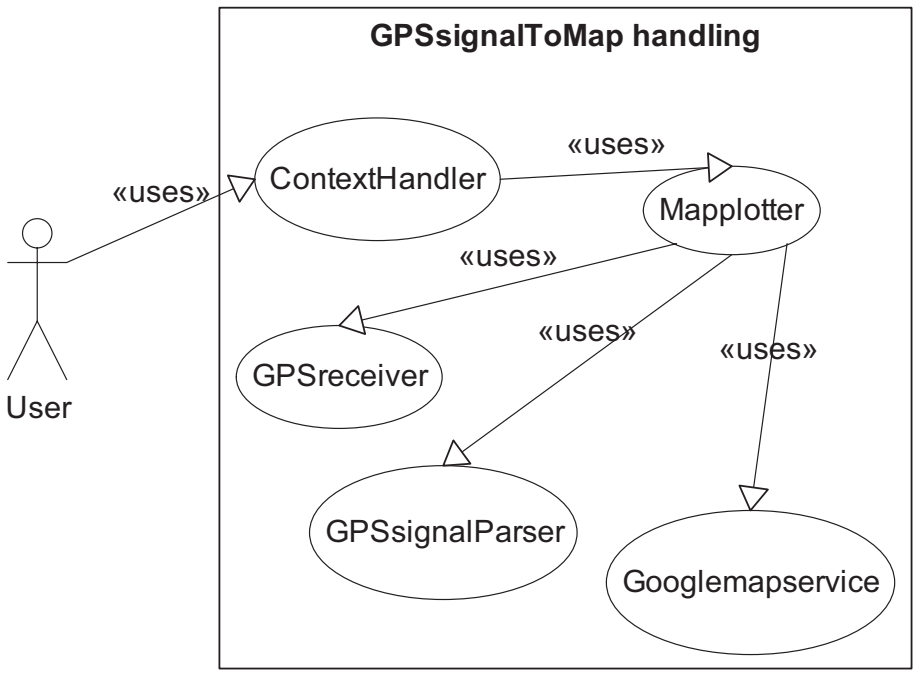

Figure 5.4: Use case GPS googlemap plotting

Use case	
Use case name	GPSsignalToMap handling
Actors	User
Preconditions	valid GPS signals received
Postconditions	Googlemapservice popped, position plotted in map
Normal flow	User pushes button, CH receives GPS coordinates, MS i.explorer launched with the coordinates
Includes	ContextHandler, GPSreceiver, GPSsignalParser, Googlemapservice, Mapplotter

Table 5.3: Usecase Local Context handling

5.3 Design

The system implemented consists of a mobile terminal (a laptop), equipped with a Bluetooth usb-dongle. The application running on this terminal acts as a context harvester, CH. Its function is to do periodically device as well as service discovery, and thereby gather up to date information about nearby Bluetooth enabled devices and their services. The terminal distributes this information through its network interface, connecting by a web service gateway to a predefined context manager CM. Location information gathered by a Bluetooth GPS receiver is included in the context update, marked as either valid or not valid information depending on the user being inside or outdoor. This is indicated with a *gpsSignalWeak* tag of the XML structured context set to true (see Figure 5.19 in section 5.5). If the user moves indoor and this *gpsSignalWeak* warning is set however, the GPSservice still polls the last signals received, indicating that the user is inside a building at that position. In other words, it is still considered relevant position information concerning the context of the user. The context output of CH formatted as a XML structured String on request from the developers of the reference system Akogrimo. An example of this XML structure, including explanation of its content, are presented in section 5.5.3.

A design overview, showing all interacting parts, is present in Figure 5.5.

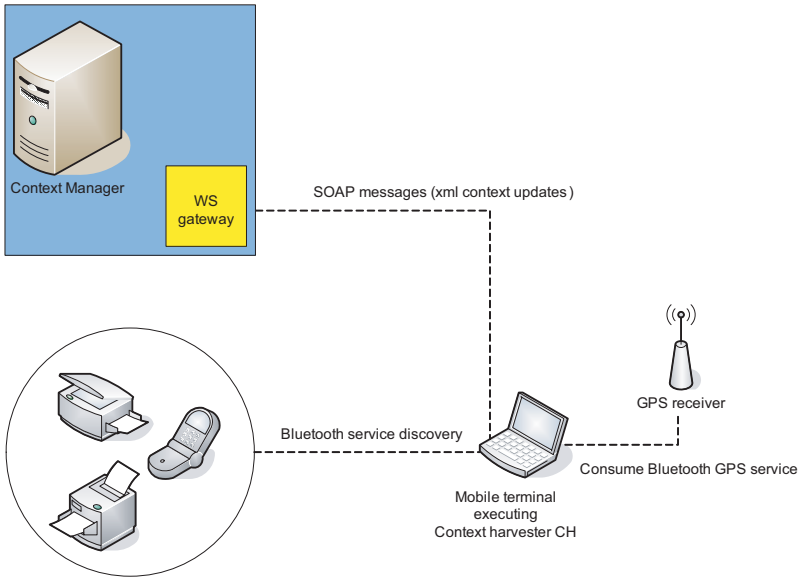

Figure 5.5: Design overview

5.3.1 Behaviour

When describing the behaviour of the implementation, we use SDL description language [39]. We choose SDL because it provides a good way to describe behaviour. Only the most relevant processes of CH and WS are illustrated with SDL process diagrams. The reference procedures, within the processes, considered necessary to understand the processes are present in separate figures marked as procedures. A list of symbols used in SDL may be found in Appendix .3.

The behaviour of the system described in Figure 5.5 consists of three interacting processes. Two of them, CH(Context Harvester) and GPSservice are located at the terminal, while WS(Web service) relates to the web service gateway receiving context updates at the CM(Context Manager) server. A fourth process is also described, though not implemented. This process, illustrated in Figure 5.10, is a proposal on how the received context at the web service gateway may be handled by CM.

5.3.2 Context Harvester CH

CH, Context Harvester, runs cycles of device discovery. This process returns information of nearby Bluetooth devices of the terminal. A web service client is trigged by these discovery processes. If new devices are discovered, these devices will be further investigated by performing service discovery. Any relevant device/service information is stored in an array, acting as an out buffer for potential web service method invocations. For every third cycle a check is done to see whether the device context profile of the terminal has changed since last WS(Web service)-update or not. If there has been a change, due to new or no longer available Bluetooth devices, this check will trigger a WS call, again causing a CM update. Before the context is updated against the WS at CM, updated location information, provided by a GPS Bluetooth service, is included in the update message. If no changes appear, no update will be made, and CH continues its device and service discovery.

We choose to update location whenever computational context is changed due to the fact that this change is a good indication of significant movement. One could argue that location should be updated independent of computational context, but this was not implemented due to the main priority of CH being device and service information gathering.

The process of CH is covered in Figure 5.6, and one of the procedures, is described in Figure 5.7.

The CH GUI provides information of nearby devices as they appear or disappear. This is considered local, relative to the terminal, surrounding device status. The CH also emulates context harvesting functionality in the sense of receiving its own device and service status at CM, the Context manager. In this way we are able to compare terminal and remote device status, as well as exemplifying a context consumer. This confirmation of updates however, is not present in the GUI, only as system outprints.

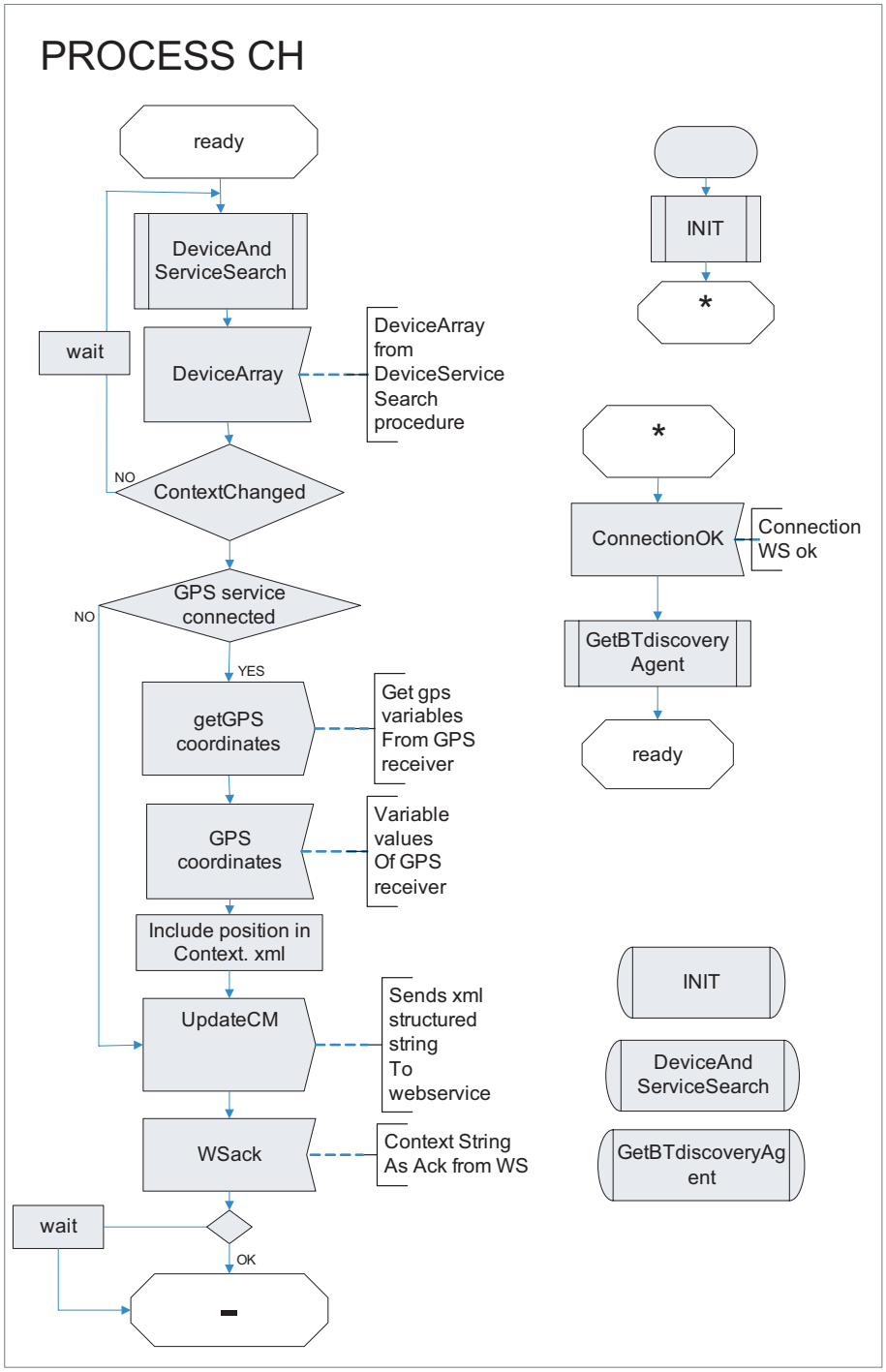

Figure 5.6: Main processes of CH

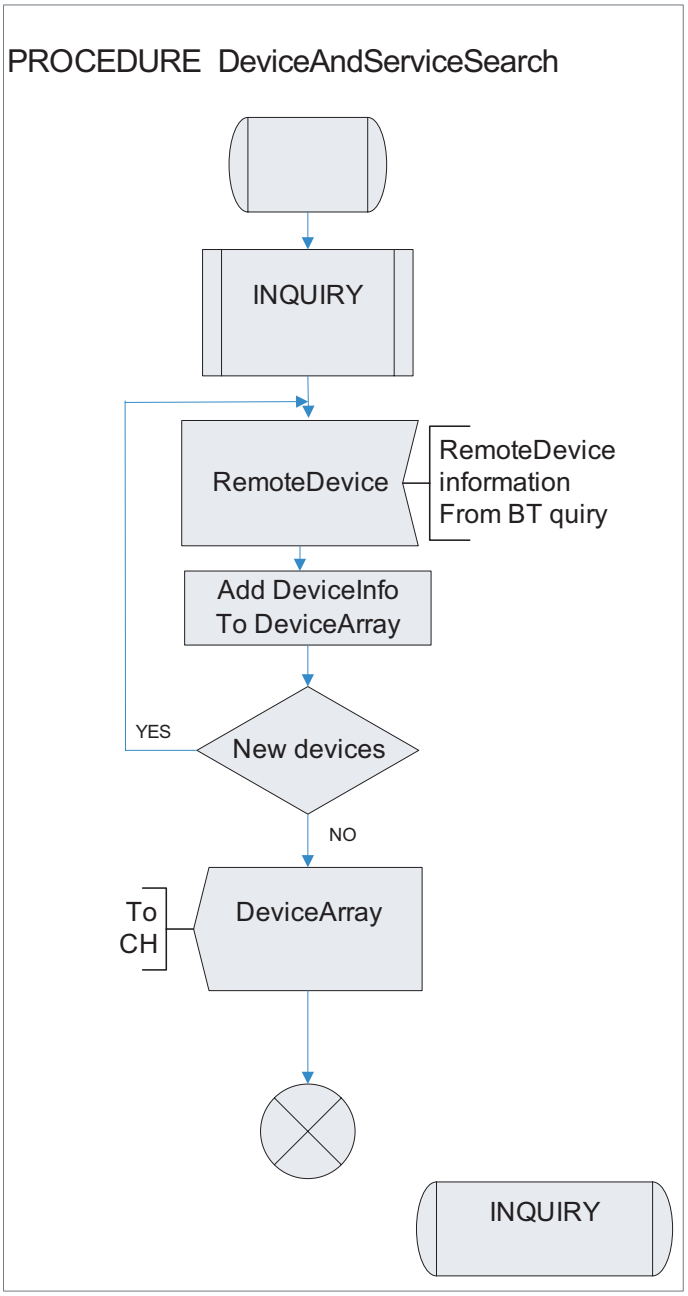

Figure 5.7: Procedure reference from process of CH

5.3.3 GPSserviceprocess

The GPSserviceprocess represents the personal GPS receiver of the user. CH connects to this service, if available, and receives coordinates whenever an CM update is made. The GPS service will be available only when the user is located outside. When he enters a building, the GPSservice continues to poll the last registered location, indicating that his current location is inside a building at that position. This kind of situation is indicated by an extra "weak GPS signal" tag included in the location context submitted by CH at context updates.

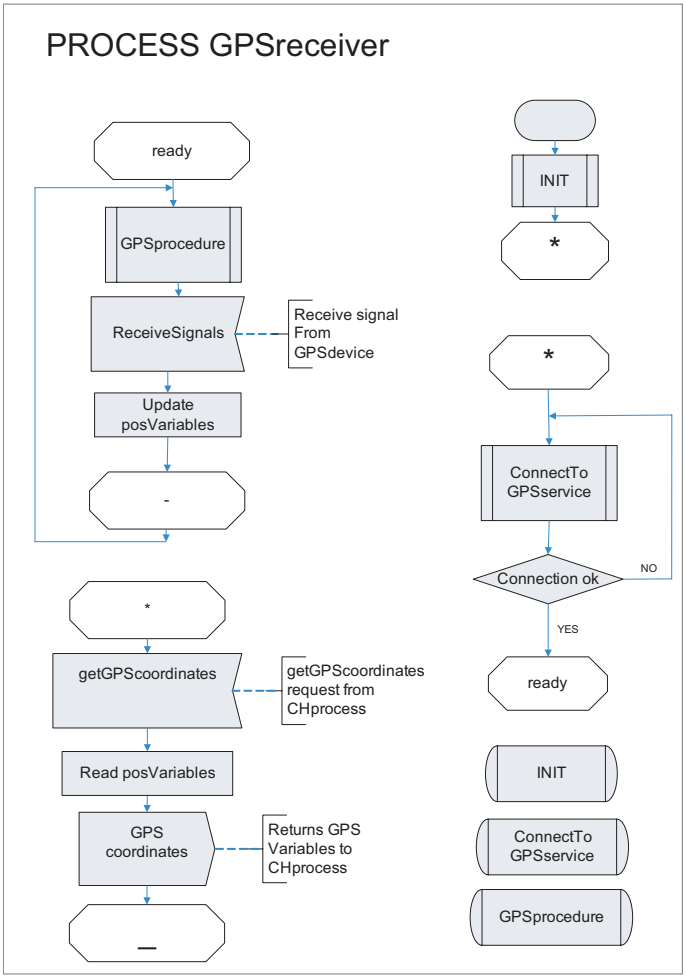

Figure 5.8: Main process of GPSservice

5.3.4 Web service WS

WS, the web service, responds to the actions of CH by updating a server side context profile of the terminal. The web service client, referred to as WSproxy, transmits updates about the surroundings of the terminal, and represents the WS at client side, i.e CH. If no change is present, no update will be sent. The web service implemented in this project has three simple functions, receive context updates, store the update and reply to CH with the updated context profile. Its process is described in Figure 5.9.

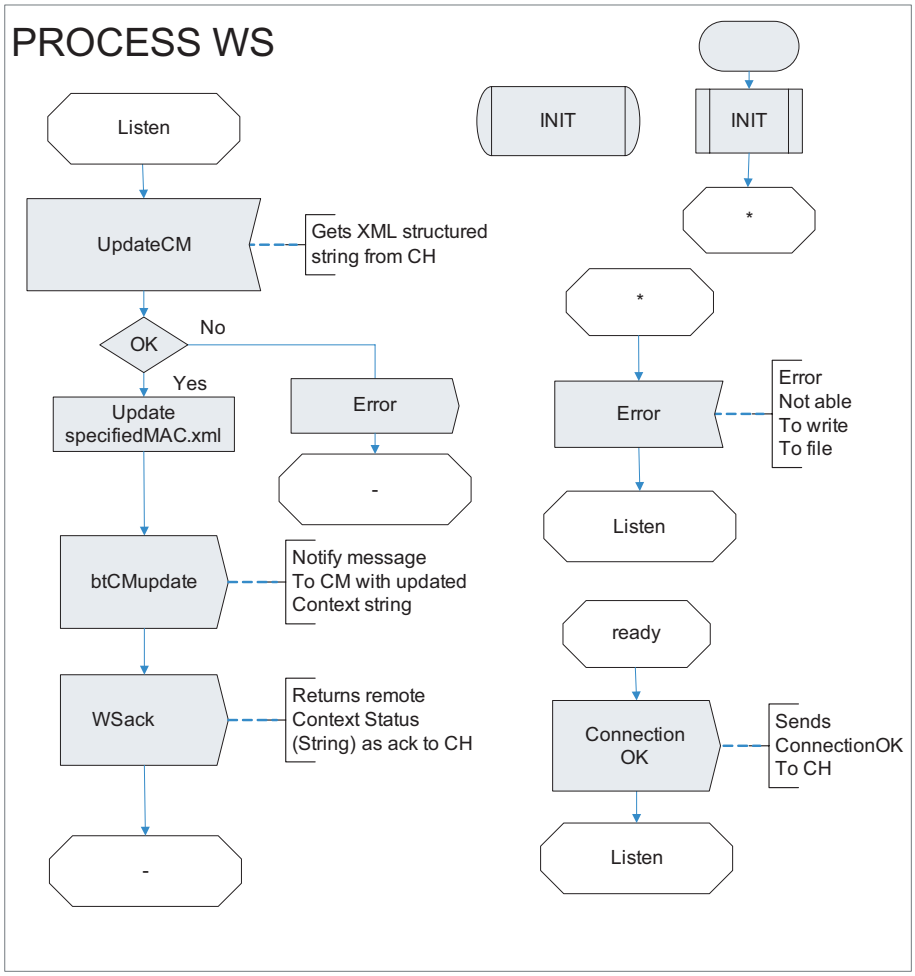

Figure 5.9: Main processes of WS

5.3.5 Context Manager CM

CM, Context manager, processes the information gathered by the interaction with WS, structuring the information and making it available to potential subscribers. These might be other context managing systems, or applications subscribing the computational as well as location context information of CH. Input notifications from WS, in case of change in context caused by CH, may notify context consumers like illustrated in the proposed process of CM in Figure 5.10. We stress that this is just a proposal of how this can be done at CM, and not included in the implementation. The process includes two procedures (not including the mandatory init). These include other context sources of the Akogrimo Context Manager, and provides processing of the total context picture, (e.g handles conflicting context information), before notifying the consumers. For more information on how the Context Manager deals with context sources, processing and distributing data, see section 2.4.2.

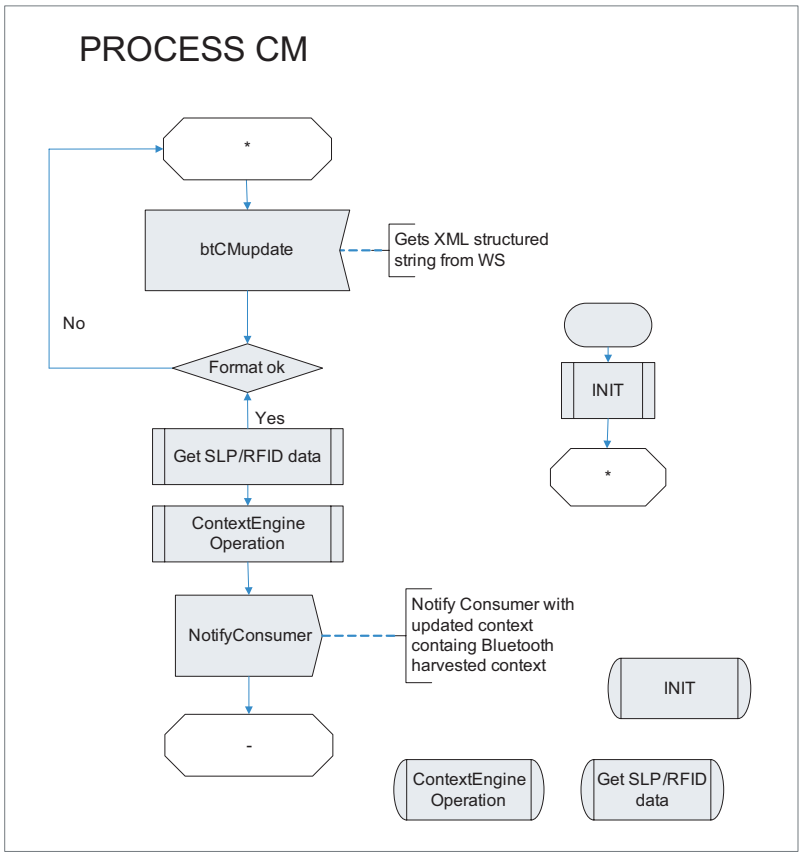

Figure 5.10: Proposed process of CM, Context Manager

5.3.6 Message sequence chart

The overall system consists of an interaction between CH, as the main actor of Bluetooth as a context harvester, a GPS Bluetooth service as well as a web service gateway to a remote context manager. The message flow between these entities is described in Figure 5.11. This message sequence diagram includes the main activities of CH, and illustrates how CH interacts with the BT API (i.e Bluecove, as described in section 5.4.1), the GPS service and the web service gateway.

This is a simplified MSC illustrating the initiation of CH, connecting to its communicating parts, as well as the normal flow of device/service discovery, Context Manager updates, and current position retrieval. The details of creating a WSproxy within CH, and its communication with WS as well as CM are described further in section 5.4.1. In this current MSC we assume that the parameters of connection between WSproxy and WS already are established. In this way, CH only needs to create a WSproxy object, and test connection in order to establish contact. We see that CH receives an XML structured context string as acknowledgment of its sent update (The web service method call returns the string it got as invocation method parameter). In this way we are able to confirm the update at CH, and at the same time simulate a context consumer.

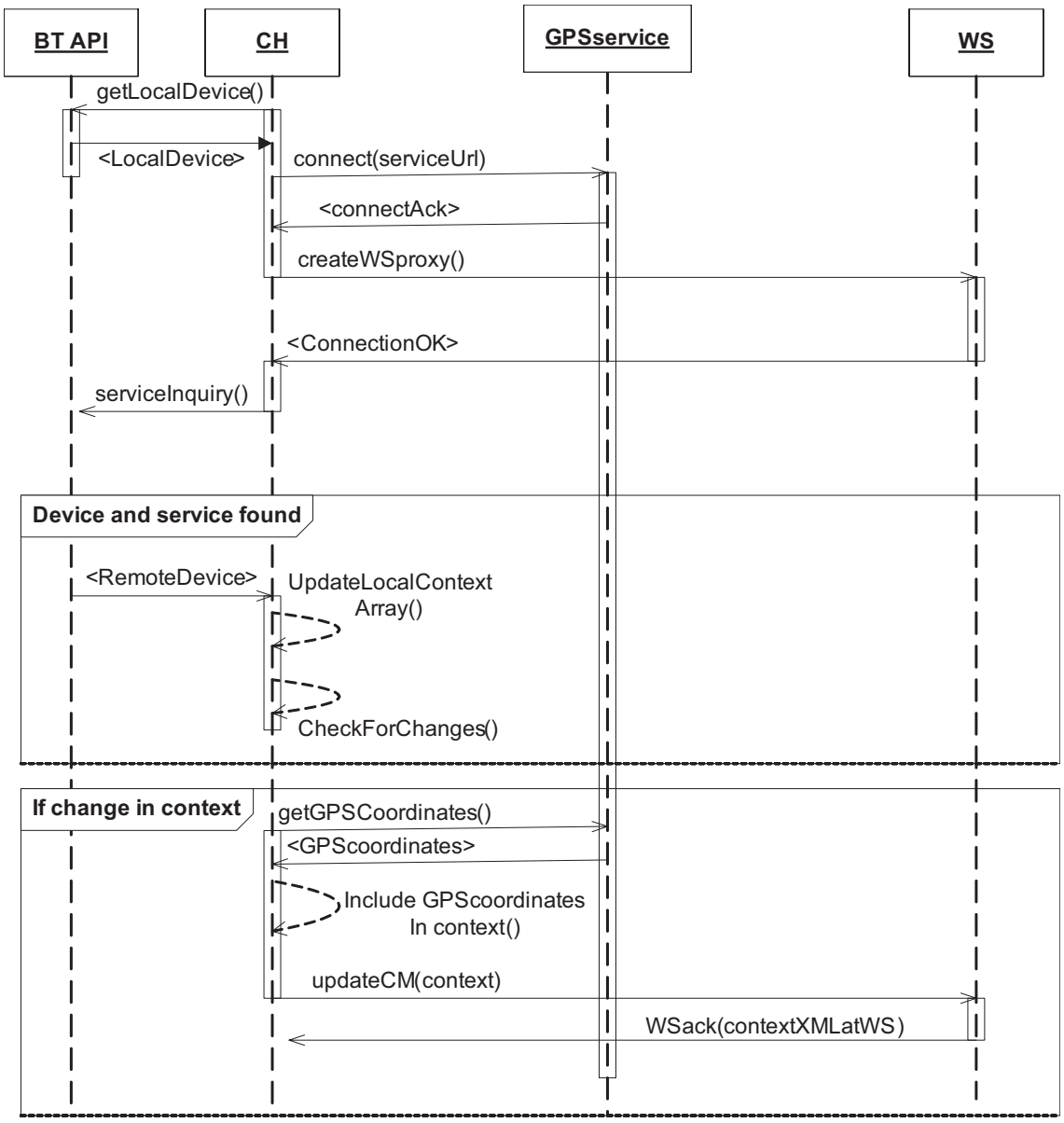

Figure 5.11: Message sequence between CH and its interacting parts

5.4 Implementation

This section covers the implementation details of the system development process. Figure 5.12 provides an overview of the different devices involved in the context aware system, as well as the platform implementation at each one of them. We see

that the core of the implementation of this report is implemented as a J2SE application interfacing the Bluecove open source JSR-82 API against the Bluetooth radio. At server side, i.e the context managing system, a simple web service context gateway was developed using the .NET framework. This project was converted to J2EE in order to fit into the apache server solution. The web service gateway represents a proposed context source gateway at CM. How this information is handled within the context engine of CM (Figure 2.5), is considered out of scope of this project.

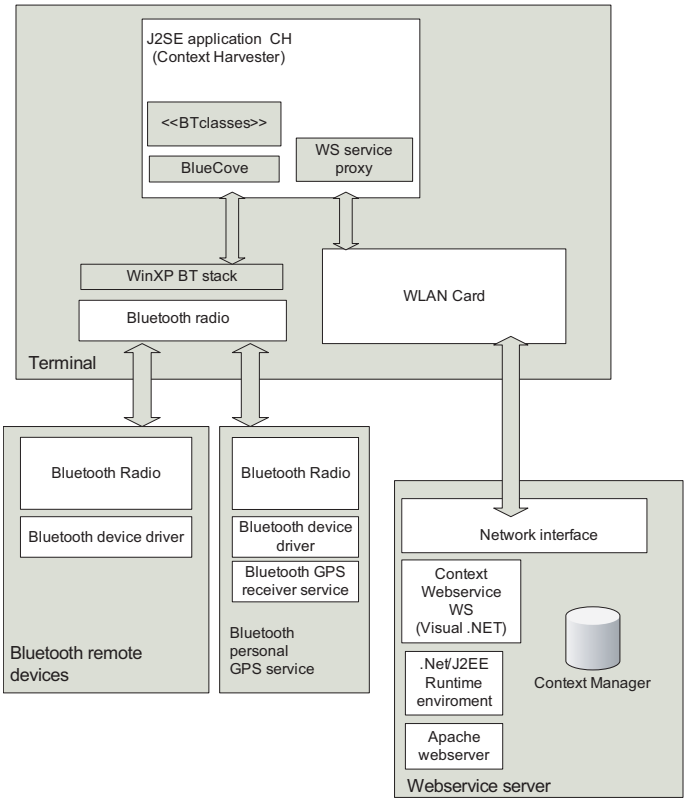

Figure 5.12: System device and implementation overview

5.4.1 Client side implementation

The context harvesting application CH, executing on the laptop computer, is written in Java, J2SE. To access the Bluetooth radio from a Java application, an extra API is needed interfacing the Bluetooth Stack.

A package relation overview at the client side implementation is presented in Figure 5.13.

henrik.com.bt represents the heart of this implementation. The inside classes and their relations are described in Figure 5.14. The main class, holding the main method, is CH. The Service* classes represent the proxy binding to the web service, while the BluetoothInquirer represents the heart of discovery processing, instantiating objects of the javax.bluetooth and the com.intel.bluetooth package.

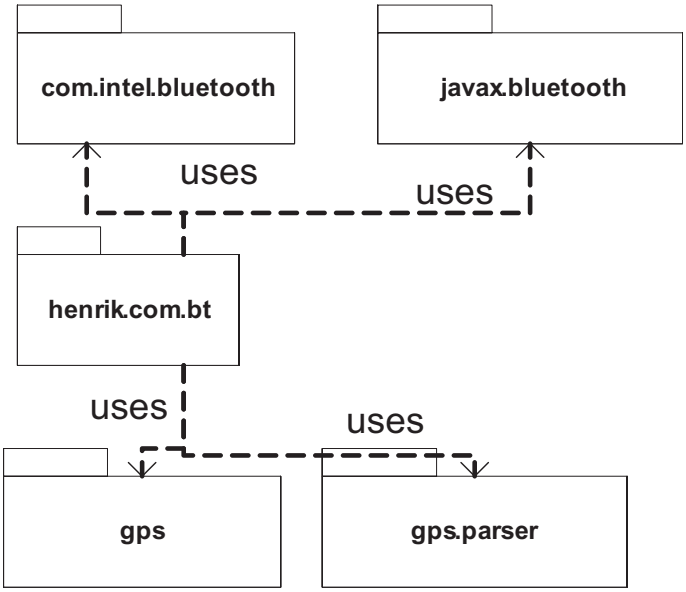

Figure 5.13: Package overview

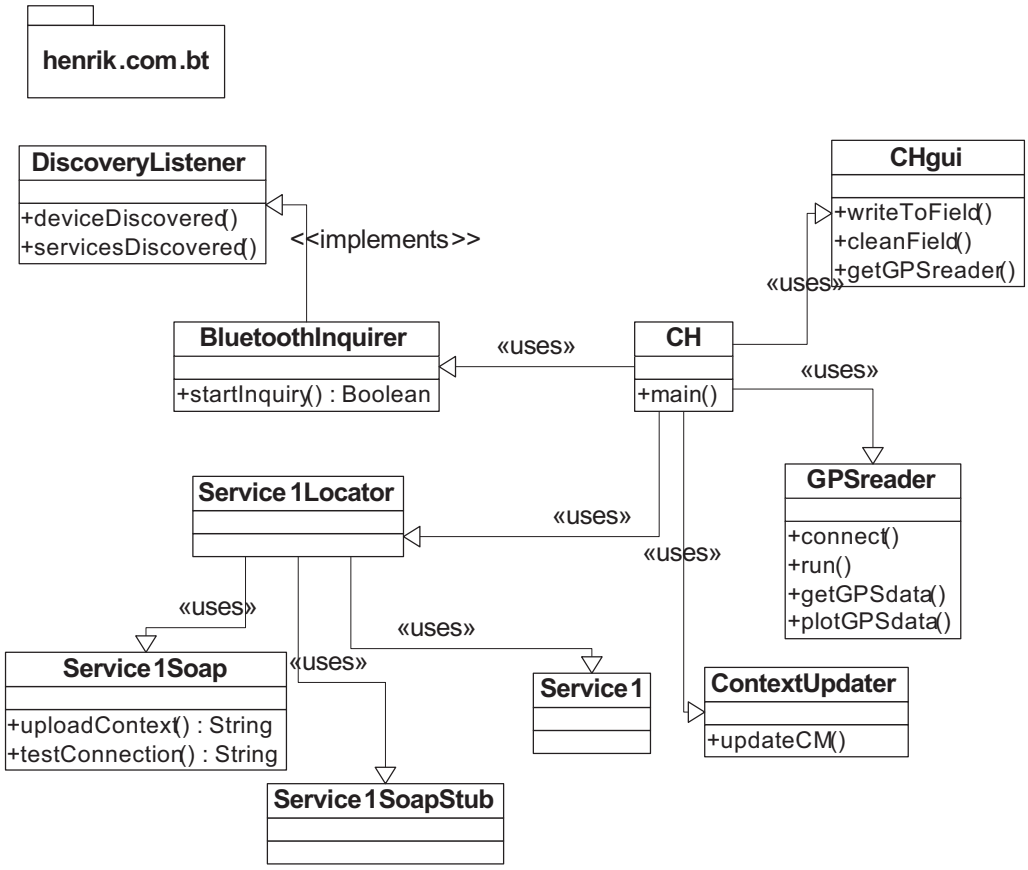

Figure 5.14: Class diagram of henrik.com.bt

Bluetooth interface The choice of Java Bluetooth API depends on the current Bluetooth stack of the terminal. Bluecove is an open source implementation of the JSR-82 Bluetooth API for Java. This solution is based on the standard Microsoft Windows XP Bluetooth stack. JSR-82 is a JCP(Java Community Process) defined standard to support development of Bluetooth enabled applications on Java platforms.

To enable device as well as service discovery on a Java application, the DiscoveryAgent class is needed. The DiscvoreyAgent provides methods for device and service search, and notifying the Client application. The client, in our case a J2SE application, must implement and register the DiscoveryListener interface in order to receive these notifications from the DiscoveryAgent. The discovery procedure for a Bluetooth enabled device is described below:

- Initialize Bluetooth Stack
- Get local Bluetooth device
- Get DiscoveryAgent
- Perform Device discovery
- Perform Service discovery

Initializing the Bluetooth stack is most often done by the Bluetooth Control Center of the particular device. This is typically a preinstalled function, which let you turn the Bluetooth radio on, set *friendly name* and *make discoverable*.

Getting the localDevice is done by calling the getLocalDevice() method, which returns a localDevice object to the client. This localDevice is used to reach the DiscoveryAgent, which is the central part of this implementation.

Getting DiscoveryAgent is done by calling getDiscoveryAgent on a localDevice object. The discoveryAgent object provides several methods for device and service discovery necessary to harvest context information. These methods are retrieveDevices(), startInquiry() and cancelInquiry(). These methods interact with the DiscoveryListener which invokes callback methods for the different stages in the inquiry phase. The message sequence and activity diagrams in the next section describe this further. [26] [29]

Web service proxy In order to establish contact and invoke methods at WS we needed to make a web service proxy object for CH. We automatically generated

the java classes needed for this communication by using the *Apache Ant 1.6*[2] tool. Apache Ant is a Java-based build tool, that uses XML-based configuration. In our case we edited the build XML with the URL of our deployed web service WS, and added the package name of our application. The output was presented in Figure 5.14 at the lower left corner(the Service* classes). More information on Ant and generating of the WSproxy can be found in Appendix .1

5.4.2 Server side implementation

We chose to implement the connection between CH and CM as a web service. Web services are platform independent and provide flexible as well as scalable services to terminals possessing network support. In this case our terminal is a standard laptop, accessing the web service through a standard wireless network interface card (see Fig 5.12).

Our web service acts as an agent between the CH and the CM. It provides a set of standard methods to CH, in order to let the terminal publish its knowledge of discovered devices as well as the current location of the terminal. In this way our web service, WS, acts as a context gateway in order to let CH publish its updates at CM. The connection of WS and CM however was not implemented in this Master thesis. This is left for the developers of the Akogrimo system to establish, and to be considered out of scope of this current project. Because of this, the development of WS was intended to be kept as simple as possible. We chose a Visual .NET solution, and used the *MainWin*Visual .NET plugin(see Appendix .1 for details) to convert the web service to a J2EE project in order to deploy it at the Apache server.

[2]Information on Apache Ant and download can be found at http://ant.apache.org/

5.5 Testing

In this section we describe the tests performed in the implementation process of this Master's thesis. We formulate test cases, and use these to validate the requirements stated. We also describe the test environment.

5.5.1 Testing components and environment

Testing of the Context Harvester CH, and its web service oriented context gateway, were carried out using:

- Standard HP laptop computer running Microsoft XP Pro, SP2.
- USB Bluetooth dongle from D-Link:
  DPT-120 FCC ID PSL-WBT-3022(Figure 5.15)
- Bluetooth GPS receiver from Emtac:
  CRUXII/BTGPS FCC ID NY8D1598(Figure 5.16)

Figure 5.15: D-Link 120, compatible with the Microsoft Bluetooth stack

Figure 5.16: Bluetooth enabled GPS receiver from Emtac

5.5. TESTING

The web service providing the context upload gateway was deployed on a Apache web server hosted by Telenor R&D Tyholt. More information of this solution may be found in Appendix A.

5.5.2 Test cases

To verify the system requirements listed in section 5.2, some test cases were launched. Details of these are listed in table 5.4.

Test cases	
T1	Application is started, check if Bluetooth inquiry activated
T2	GPS receiver turned on, check signal response
T3	Bluetooth remote devices turned on and off to register change in context
T4	remote desktop set up to verify context changes at CM computing domain
T5	checking validity of context XML structure at CM
T6	Pushing "plot position" button of CHgui to validate position outdoor
T7	Moving terminal inside building to verify signal warning of GPS

Table 5.4: Simple test scenarios to verify requirements of CH and WS

Based on these test cases we verify the system requirements of table 5.2. Testing the context harvesting system we realized the following scenario:

A laptop representing the mobile terminal, equipped with a Bluetooth USB dongle hosts the context harvesting application CH. To avoid conflicts of access with the GPS terminal, the USB Bluetooth dongle was paired[3] with the GPS receiver prior to the test phase .

During execution we moved the terminal around, discovering different devices, and of course changing the position. At the same time, the data received by the web service interface at the server hosting CM was monitored through a remote desktop solution.

Realizing this scenario we had the opportunity to monitor the response time of updates to CM as the environment of Bluetooth devices changed. We could also see how the GPS receiver, carried around together with the laptop (as it was to be considered a personal device of the user), changed the position included in each context update. We also got to approve the fact that the warning indicator of the GPS location context turned true as we went indoor.

Print screens of these tests are found in Figure 5.17, Figure 5.18 and Figure 5.19. Fig 5.17 shows a simple GUI of CH visualizing the discovery of any nearby devices to be included in the computational context of the user. Figure 5.18 displays the result of pushing the *plot GPS to googlemap* of the CH GUI. This functionality was added to provide a simple way of validating the position coordinates received by

[3]Pairing of Bluetooth devices involves creation of trust relationship between the devices by generating and storing authentication keys for future device authentication [38]

requirement	verified by	Comments
R1	T1,T3	Test passed. Device update at GUI. The other verified by sys.outprints
R3	T1,T3,T4	Test passed
R4	T3,T4	Test passed
R5	T2,T4,T6,T7	Test passed
R6	T6	Test passed
W1	T4	Test passed
W2	T3,T4	Test passed
W3	T5	Test passed
W4	T4	Test passed. Context saved to XML file on form MACusbdongle.xml

Table 5.5: Test result table

the GPS service, as well as investigating the precision of the location. Figure 5.19 illustrates how the context updates are received and structured as XML at the web service gateway at CM. Documentation of this test, verifying the requirements of the system is presented in table 5.5.

5.5.3 Testings print screens

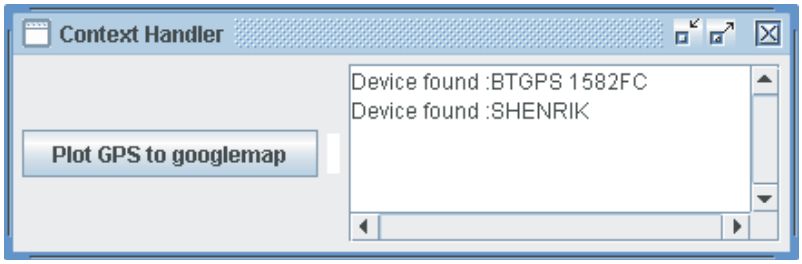

Figure 5.17: The CH GUI displays the current surrounding Bluetooth devices. The GPS device on top in the text field represents the GPS service providing location of the terminal

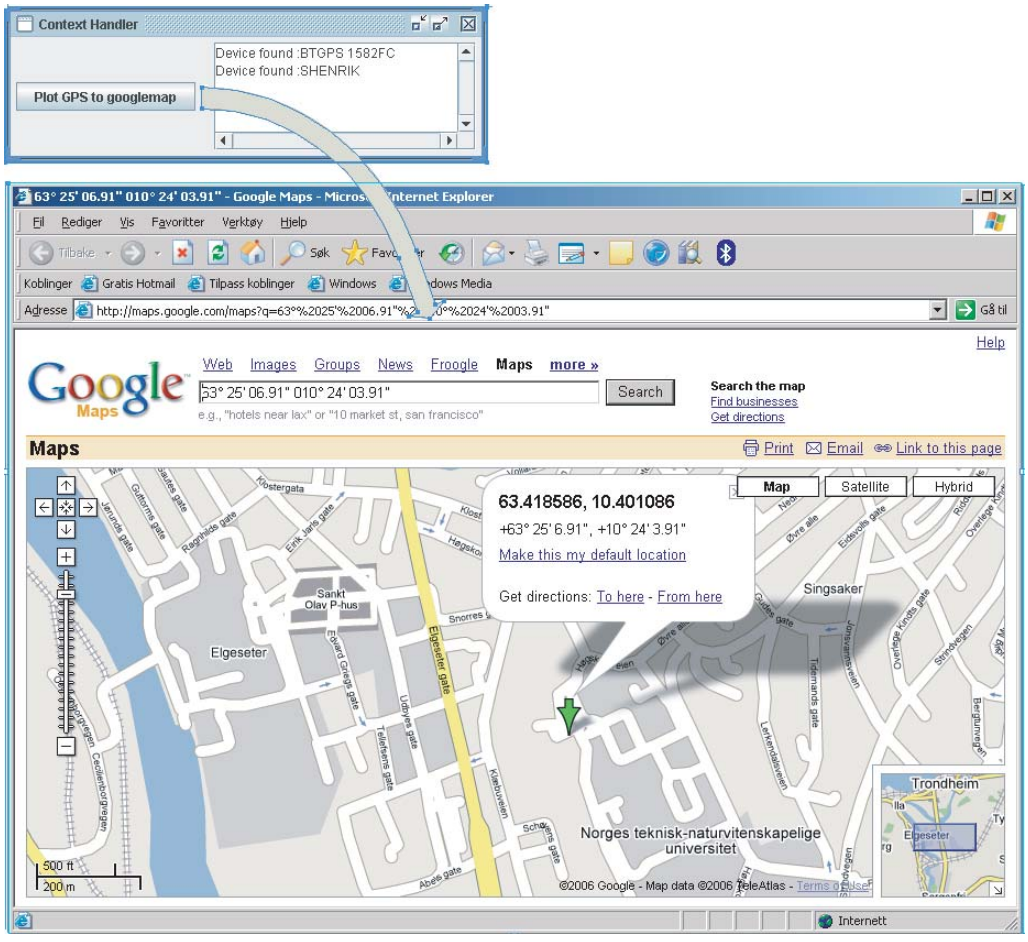

Figure 5.18: Pushing the plot button automatically results in executing i.explorer, with the googlemap as well as the current gps coordinates as parameters

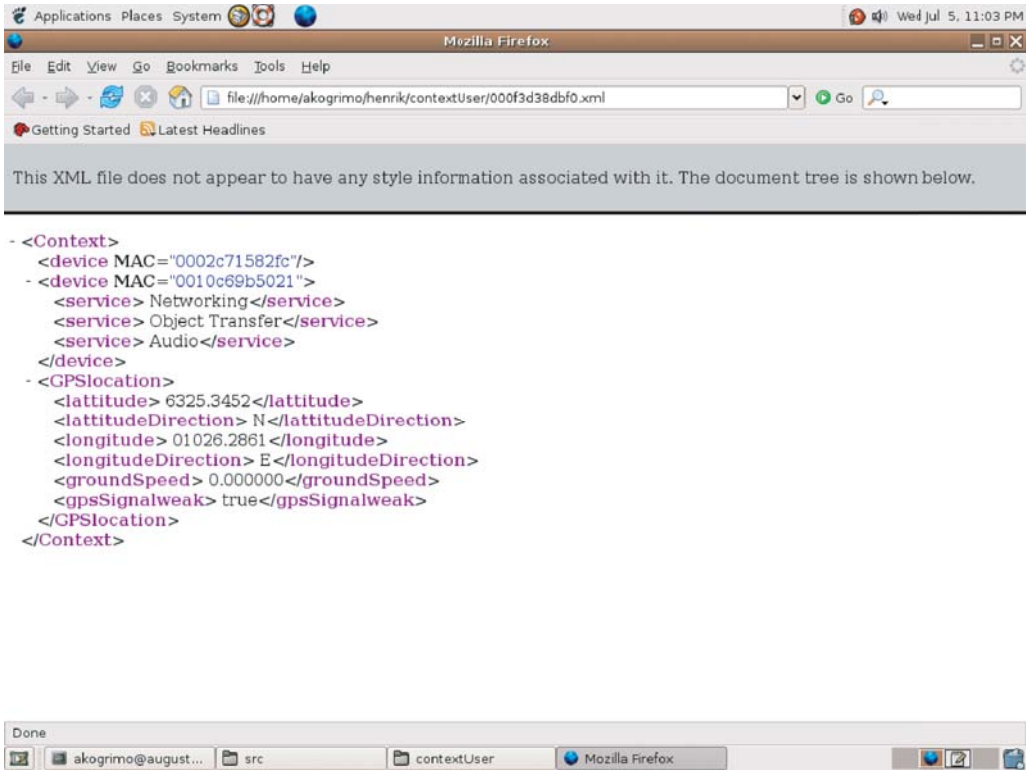

Figure 5.19: The computational as well as location context are updated with WS(at the domain of CM), structured as XML

In Figure 5.19 wee see the result of a context update by CH through the WS context gateway. The XML file is viewed with Mozilla Firefox at the server hosting both WS and CM. In the address field we see the location path of the context-XML file, 000f3d38dbf0.xml. The numbers representing the filename correspond to the Bluetooth MAC address of the Bluetooth USB dongle used by CH. This way of storing the updates helped separate context uploads from potential different terminals (i.e different users).

The content of the file is described as follows: First we list the different MAC addresses of the devices discovered. By including this information way we have the chance of mapping each of these to both location and other devices as described in chapter 3.

Within each device we list the different services provided. The first device includes no such information. This is actually the GPS receiver. The device was defined as an unknown device by the Bluetooth API used in this implementation, in other words no services could be discovered by Service Discovery. But because this was considered a private and preknown device of CH, we connected directly using with the service using a Serial Profile connection of Bluetooth (see the Bluetooth stack

in Figure 4.8, section 4.7.4), and the MAC address of the device. The next device is again identified by its MAC, and we see three different services provided.

The lower section of the file relates to location information consumed by the GPS reader. The signals received are based on the NMEA (National Marine Electronics Association) standard. Notice that the last tag of this section, *gpsSignalweak* is set to true, indicating that the terminal is inside a building.

Chapter 6

Discussion

During this chapter we evaluate the theory of Bluetooth as a source of context information presented in this report, as well as discussing the choices made during the implementation. We look into some of the possibilities of adapting this system to a large scale context managing system, and discuss some proposals on this topic in the light of the Akogrimo project(chapter 2.4.2).

6.1 Bluetooth as a source of context

Throughout this report we have analyzed the different capabilities of Bluetooth as a PAN service discovery technology to contribute as a context harvester. There is no doubt about the potential of Bluetooth as a service discovery technology concerning computational context gathered from Bluetooth enabled devices. But as we have seen, Bluetooth may also contribute to estimate location information in context aware systems. In complex service discovery systems, Bluetooth also has the capability to act as a bridge between other service discovery technologies like SLP and UPnP.

The Akogrimo system was used as reference system, in order to evaluate how a radio PAN technology like Bluetooth could be exploited to enrich computational context compared to a solution based on RFID location and SLP services. Bluetooth has the advantage of being a mobile service discovery solution covering the natural radius of what can be considered "computational context area" of a user, that is what is relative close to him. Also, by using a Bluetooth solution, computational context may be enriched without bothering about location detection or limitations of only discovering registered services.

However, there are certainly challenges to face when involving Bluetooth in a context aware system. Obviously, the limitation of just discovering Bluetooth devices is one of these. This can of course be solved by bridging technologies, but anyhow you would need at least one Bluetooth device to be discovered by the context harvesting terminal in order to reach potential bridged SLP or UPnP services.

Dealing with the discovery range of Bluetooth, its also worth mentioning the limitation of using a Bluetooth radio coverage as a limitation of "relevant" services to a user. What is relevant information of surroundings of a user or not, should be decided by the context consumer and not the physical limitation of a radio.

In chapter 3.2, we briefly looked into the drawback of Bluetooth in harvesting in a mobile environment. Services may pop up any time, both mobile and stationary ones, at the same time as the user may be moving around. In this way, service capabilities may be included in a context update, and in the next second be unavailable. This issue was not a high priority of our implementation and only dealt with to a small extent. By letting the Bluetooth do a fixed number of service inquires before including the discovered services in a context update, we reduced this instability aspect.

Another challenge is access to the devices and services provided by Bluetooth. Performing device and service discovery is rather easy. Consuming the services however is not necessarily straight forward. Making a connection to a remote Bluetooth device usually demands use of a pin code in order to pair the devices before communication. This relative cumbersome way of accessing Bluetooth services definitely represents a weakness when it comes to involving Bluetooth SDP in a mobile dynamic context aware environment.

Yet another aspect of consuming Bluetooth services is the link quality. This Master's thesis has mainly been focusing on the discovery process of Bluetooth services, gathering information. To most of todays Bluetooth services link quality and bandwidth capacity aren't the biggest concern. But in the sense of making Bluetooth part of a context aware system it would be preferable to also admit hight data rate services with minimal delay. We have briefly covered scenarios with context aware systems discovering LCD screens in order to upgrade ongoing sessions of a user. The discovery of this screen could be done with Bluetooth technology. But high variations in bit rate, unpredictable transmission rates and possible interference with other wireless technologies would probably make it difficult to let Bluetooth do anything more than just that. [36]

Some work has been done letting Bluetooth be an enabler for multimedia content transmissions, but the new and high bandwidth providing PAN technology of UWB (Ultra-wide band Communications) is probably more promising at these task are-

nas. [37]

6.2 The implementation

In order to illustrate the strength of Bluetooth in a context harvesting subsystem of projects like Akogrimo, we implemented a Bluetooth enabled context harvester referred to in this report as CH(Context Harvester). The main object of this implementation was to exploit the capabilities of Bluetooth to harvest computational as well as locational context, and notify a remote context managing system through a web service gateway. A simple web service was implemented in order to make this connection possible.

Several choices needed to be done concerning selection of both mobile terminal and implementation languages. The main actor of the system, CH, was implemented as a J2SE application, to be run on a laptop simulating the mobile terminal. We could have chosen a Bluetooth enabled PDA or smart phone, using J2ME (Java micro edition), but due to practical reasons we did not. WLAN and Bluetooth were vital requirements to the mobile terminal. Bluetooth obviously because of the main object of the implementation, WLAN because of the need of a mobile network gateway to the Context Manager. The choice of terminal and J2SE as platform eased the implementation process in ways of finding suitable and free open source solutions (e.g the Bluetooth API, Bluecove), and of course in the work of testing and validating the functional requirements. Still we kept the important property of mobility, choosing a wireless network enabled laptop.

At server side, developing the web service as a context gateway, we chose a Visual .NET solution. In chapter 2.3 we looked into context aware system making use of the SIP protocol for context exchange. Letting the Context Harvester, CH, communicate its context updates by making use of a web service, we avoided letting the source of context be dependent on a SIP implementation. Thereby, we added flexibility to our solution.

Based on this implementation foundation, we were able to make a solution enabling context harvesting via Bluetooth, dealing with computational as well as location information.

6.3 Adaptation to Context Manager

Integrating context, harvested by Bluetooth, as an extra source of context in the Akogrimo will create a need for modifications of the Context Manager system. What kind of adaption needed depends heavily on the tasks a Bluetooth context source is delegated. If the only mission is to gather computational context concerning nearby Bluetooth services the change should be rather straight forward, like adding an extra entry of computational context database of the Akogrimo system.

If however, we want Bluetooth to gather indoor location context, the situation is different. This solution would be based on mapping all fixed Bluetooth devices to areacodes, similar to the mapping of SLP services and RFID. In addition, if this information should make any sense, Bluetooth positioning algorithms should be implemented and a reference map relating areacodes to physical rooms should be established. If coexistence with the RFID location technology, some kind of logic on how to choose between the two location sources should also be implemented.

We have the same situation with computational context; if enhanced Bluetooth service discovery is used to discover services beyond the Bluetooth domain, e.g SLP, UPnP and Jini. The computational context gathered from the original system, i.e SLP services, may conflict with the information gathered by Bluetooth.

Chapter 7

Conclusion and Future work

The object of this Master's thesis was evaluating Bluetooth as a PAN technology in context harvesting. By evaluating related work of context aware computing, we have looked in to how Bluetooth may contribute in this computing domain, answering the following;

- How may a PAN technology like Bluetooth be used for context harvesting?
- How may Bluetooth be implemented for this purpose?

In addition, we have looked into how Bluetooth may fit in to a large scale context managing system, like the Akogrimo project [13] when it comes to gathering, managing and structuring relevant context information.

This chapter provides conclusion as well as what is considered future work of this Master's thesis.

7.1 Conclusion

Throughout this report we have analyzed the capabilities of Bluetooth in context acquisition. A lot of related work on context awareness concerning computational context has been centered around deriving this kind of information based on location and presence status of the user involved.

This report has evaluated the capability of Bluetooth as a replacement or supplement of this method. We have implemented a solution illustrating the behaviour

of Bluetooth as an enabler of harvesting both computational and location context. Through several scenarios, like listed in section 1.1, we have seen the benefits of context awareness and how this technology is dependent on subsystems providing information like computational and location context.

Cons and pros of Bluetooth, as an enabler of context acquisition, have been evaluated, strengths and possible weaknesses pointed out. We have experienced, through implementation and by looking into related work, the advantage of using Bluetooth when collecting computational context. Its adhoc capability and powerful service discovery mechanism provide valuable information of the device and service status of a mobile user, i.e his computational context.

In addition we have looked into several ways of how Bluetooth may be used in providing location context. Proposals on how these methods may be exploited in context aware computing have been provided, and one of them included in the implementation of the Bluetooth enabled Context Harvester, CH.

Compared to related work on how computational context may be provided, we see obvious benefits on how Bluetooth may be used to improve this acquisition. We have seen how the Akogrimo project provides this kind of information to their users by tracing the location of the terminal and map to registered services at this area. This way of discovering and of course, in time consume services, may be replaced or enhanced with Bluetooth technology. This would contribute to improved mobility in a context aware computing domain, in the sense of allowing dynamic changes in location of both registered and unregistered devices and services. At the same time we reduce the dependability of location data by RFID (mapped to SLP services), in order to enrich computational context of a user.

We made a context gateway proposal to the context harvesting application in order to provide the information gathered to a context manager. We see that some modification must be made concerning functionality of the Context Manager of Akogrimo in order to fully exploit a Bluetooth harvesting mechanism. A conceptual design of a process of CM handling this issue was presented in section 5.3.5, Figure 5.10.

Based on this Master's thesis we see great value in exploiting the capabilities of PAN technologies in context aware systems. If Bluetooth is the one technology breaking through at this computing domain is left to see. Due to its already wide acceptance and gained popularity it certainly has got the potential.

7.2 Future work

- The Context Harvester, CH, has yet not been tested with the Context Manager, CM, of Akogrimo due to a lack of connection between the web service, WS, of this Master's thesis and CM. The context format, requested to be an XML structured string has been provided, though some implementation adaptations will probably need further work.

- We also suggest further analysis on how services, not only may be discovered by Bluetooth, but also consumed in a way that suites the ideas of pervasive computing. The pin code distribution architecture proposed in section 3.3.1 could be a candidate enabler of this.

- Last, due to the high density of Bluetooth enabled devices in our environment today, it could be valuable to investigate the methods of indoor Bluetooth positioning further in the light of context aware computing. Concerning the Akogrimo project, this way of measuring indoor location could contribute as a supplement to using RFID, and performance comparison could be carried out.

Bibliography

Bibliography

[1] G.Abowd,M.R.Ebling,H.Gellersen,G.Hunt,H.Lei;
"Context-aware pervasive computing";
IEEE Wireless Communications;
2002;

[2] E.C.Oesthus, P.O.Osland, L.Kristiansen;
"Context based session mobility";
ICIN'06;
2006;

[3] P.Huuskonen,D.Pavel,U.Tuomela;
"Context-awareness: A new dimension for communication";
http://www.nokia.com/nokia/0,,53721,00.html;
2003;
(online 11.12.05)

[4] Olsen.L.R, Murakami.H, Schwefel.H.P, Prasad.R;
"User centric Service Discovery in Personal Netorks";
http://www.ist-magnet.org /private/files/Dissemination/WP2/wpmc2004.pdf
2004;
(online 11.12.05)

[5] M.Weiser;
"The Computer for the 21st Century";
Pervasive Computing, IEEE, Vol 1, Issue 1;
2002;

[6] A.K.Dey,G.D.Abowd,D.Salber;
"A Context-Based Infrastructure for Smart Enviroments";
www.cc.gatech.edu/fce/contexttoolkit/pubs/MANSE99.pdf;
(online 11.12.05)

[7] S.K.Mostefaoui,G.Kouadri.Mostefaoui;
"Towards a contextualisation of service discovery and composition for pervasive enviroments";
AAMAS conferance workshop on webservices, Australia;
2003;

[8] L.Wei;
"A service oriented SIP infrastructure for Adaptive and Context-Aware Wireless Services";
Proceeding of 2'nd International Conferance on Mobile and Ubiquitious Multimedia;
2003;

[9] S.Berger,H.Schulzrinne,S.Sidiroglou,X.Wu;
"Ubiquitious Computing Using SIP";
ACM NOSSDAV'03;
2003;

[10] M.Goertz,R.Ackermann,R.Steinmetz;
"Enhanced SIP Communication Services by Context Sharing";
Proceeding of the 30th EUROMICRO Conferance;
2004;

[11] E.Guttman,
"Service location protocol: automatic discovery of IP network services";
Internet Computing, IEEE Volume3,Issue4;
1999;

[12] S.Helal;
"Standars for service discovery and delivery";
Pervasive computing IEEE, Volume 1, Issue 3;
2002;

[13] Telenor;
"Akogrimo D.4.2.1",Overall network middleware requirements report, Ver 1.0;
2005;
(see reference 15 for more details);

[14] P.Osland;
"Enabling context-aware applications";
Proceedings of ICIN 2006: Convergence in Services, Media and Networks

Bordeaux, France 29 May - 1 June;
2006;

[15] P.Osland;
Final Integrated Services Design and Implementation Report
Akogrimo Deliverable D4.2.2. November 2005. URL: www.akogrimo.org
(path - Downloads - Deliverables - Final)
Integrated Services Design and Implementation Report;
URL last visited 31.1.06;
2005;

[16] G.G.RICHARD III;
"Service Advertisement and Discovery: Enabling Universal Device Cooperation";
Internet Computing, IEEE;
2000;

[17] C.Lee,S.Helal;
"Protocols for service discovery in dynamic and mobile networks";
Internastonal Journal of Computer Research;
Nova Science Publishers;
www.harris.cise.ufl.edu/projects/publications/servicediscovery
(online 11.12.05)

[18] P.G.Raverdy,V.Issarny;
"Context-Aware Service Discovery in Heterogeneous Networks";
World of Wireless Mobile and Multimedia Networks, IEEE;
2005;

[19] S.Avancha, A.Joshi,T.Finin;
"Enhanced Service Discovery in Bluetooth";
Computer, Volume 35, Issue 6;IEEE;
2002;

[20] P.Golding;
"Next generation wireless applications", chapt.11;
Wiley;
2004;
ISBN: 0-470-86986-0

[21] Service Discovery - Wikipedia article;
http://en.wikipedia.org/wiki/Service_discovery;
Last visited 5.7.2006;

[22] C.Lee,S.Helal;
"Context Attributes: An Approach to enable Context-awareness for Service Discovery";
Applications and the Internet;IEEE;
2003;

[23] C.Bala Kumar, P.Kline, T.Thompson;
"Bluetooth application programming with the Java APIs";
Morgan Kaufmann Publishers;
2004;

[24] D.Kammer,G.McNutt,B.Senese,J.Bray;
"Bluetooth application developer's guide", chapter 1;
Syngress Publishin Inc;2002;
ISBN: 1-928994-42-3

[25] S.Liu
"Bluetooth Technology"
progtutorials.tripod.com/Bluetooth(us)Technology
(online 11.12.05)

[26] C.E.Ortiz;
"Using the Java APIs for Bluetooth Wireless Technology
Part1- API Overview";
2004;
developers.sun.com/techtopics/mobility/apis/articles/bluetoothintro/
(online 11.12.05)

[27] J.Allard, V.Chinta, S.Gundala, G.G.Richard;
"Jini Meets UPnP: An architecture for Jini/UPnP Interoperability";
Applications on the internet;IEEE conferance proceeding;
2003;

[28] Bluetooth SIG;
"Bluetooth ESDP for UPnP";
2001;
www.comms.scitech.susx.ac.uk/fft/bluetooth/ESDP(u.s)UPnP(u.s)0(u.s)95a
(online 11.12.05)

[29] B.Hopkins,R.Antony;
"Bluetooth for Java";
Apress;
2003;
ISBN:1-59059-078-3

[30] I.Sommerville; "Software Engineering" 6th Edition, chap.3;
Addison-Wesley;
2001;
ISBN:0-201-3981-5-X

[31] P.Kruchten;
"Tutorial:Introduction to the Rational Unified Process";
Software Engineering;
IEEE;
2002;

[32] C.Flora, M.Ficco, S.Russo, V.Vecchio;
"Indoor and outdoor location based services for portable wireless devices"
Proceedings 25th IEEE conferance, Distributed computing systems;
IEEE;
2005;

[33] J.Li, R.Tong, M.Tang, J.Dong;
"WebService-Based Distributed Feature Library";
Proceedings of 8th int conferance on Computer Supported Cooperative work;
IEEE;
2003;

[34] H.Davidsen;
"Context and Service Discovery";
Project carried out at Institute of Telematics;
2005;

[35] A.Kotanen,M.Hannikainen;
"Experiments on Local Posistioning with Bluetooth"
IEEE computers and communications;
2003;

[36] H.B.Kazemian;
"An Adaptive Control for Video Transmission
Over Bluetooth"
IEEE wireless communication;
2006;

[37] G.Aiello, Gerald.D.Rogerson;
"ultra-wideband wireless systems";
IEEE microwave magazine;
2003;

[38] B.A.Miller;
"Bluetooth revealed";
Prentice Hall PTR;
ISBN 0-13-067237-8;
2002;

[39] R.Braek, Haugen;
"Engineering real time systems. An object oriented methodology using SDL";
Prentice Hall;
1993;

.1 APPENDIX A

Installation notes:

Terminal (client CH)

Requirements:

- In order to get this application to work Microsoft Bluetooth stack must be installed, i.e the dongle (like D-link 120)supporting this stack must be installed
- J2SE SDK 1.4 or above must be installed
- Windows XP SP2 installed
- MS Internet explorer installed at default location (see attached readme file)

Running the application:

All the source code of both Context Harvester and the web service gateway can be collected from the attached file of this Master's thesis. A readme file is also included.

1. Place intelbth.dll in your library path, system32 catalog of windows, and BlueCove.jar in your class path, all from the Bluecove package
2. Unzip the folders of CH.zip and import into your workspace of Eclipse
3. Plug the USB Bluetooth dongle, and turn on both wireless network and GPS receiver
4. Run CH (from the henrik.com.bt package at src of the bluecove folder) as Java application

More information of Bluecove, as the chosen BT API of this Master thesis, may be found at http://sourceforge.net/search/?type_of_search=soft&words=bluecove

Notice, that if another Bluetooth dongle, or another GPS device is used then the ones in this Master thesis, Bluetooth pairing of these devices must probably be done before running CH. This includes a simple procedure of exchanging self generated pin codes. Information of this process may be found at: http://kb.palmone.com/SRVS/CGI-BIN/WEBCGI.EXE?New,Kb=PalmSupportKB,ts=Palm_External2001,case=obj(14817)

Serverside (Web service WS)

As long as the web service server at Telenor R&D is up and the apache webserver started, contact should be established as the application starts

As explained in section 5.4.1, the proxy classes were generated with an Apache Ant tool using the URL of the web service WSDL(Web Services Description Language) file as parameter in a build XML file. The build XML file is attached to this Master's thesis along with the other parts of the implementation in the EXTRAS folder. This XML file could easily be investigated and edited, in case any change to the Webservice should appear. (E.g new functionality, new host server etc) The version of apache ant used is apache-ant-1.6.5

The web service was implemented in Visual Stuido .NET, and converted to fit the apache server solution by the *MainWin* Visual Studio plugin. This plugin helps converting the Visual project to a J2EE project, creating a *.WAR* file. This WAR file represents the web service when deployed at the Apache server. Information on the MainWin plugin may be found at
http://www.codeproject.com/showcase/Mainsoft_grasshopper.asp.

The web service was deployed at the following url:
http://129.241.219.164:8080/chTOcmVer2/Service1.asmx?wsdl

The source code of this Visual .Net service can be found in a separate folder of the attached file.

.2 APPENDIX B

Bluetooth Service UUIDs

.2.1

SerialPort	uuid16 0x1101	See Generic Access Profile, Bluetooth SIG.
LANAccessUsingPPP	uuid16 0x1102	
DialupNetworking	uuid16 0x1103	See Dial-up Networking Profile, Bluetooth SIG.
IrMCSync	uuid16 0x1104	See Synchronization Profile, Bluetooth SIG.
OBEXObjectPush	uuid16 0x1105	See Object Push Profile, Bluetooth SIG.
OBEXFileTransfer	uuid16 0x1106	See File Transfer Profile, Bluetooth SIG.
IrMCSyncCommand	uuid16 0x1107	See Synchronization Profile, Bluetooth SIG.
Headset	uuid16 0x1108	See Generic Access Profile, Bluetooth SIG.
CordlessTelephony	uuid16 0x1109	See Cordless Telephony Profile, Bluetooth SIG.
AudioSource	uuid16 0x110A	
AudioSink	uuid16 0x110B	
A/V_RemoteControlTarget	uuid16 0x110C	Audio/Video Control Profile, Bluetooth SIG
AdvancedAudioDistribution	uuid16 0x110D	Advanced Audio Distribution Profile, Bluetooth SIG
A/V_RemoteControl	uuid16 0x110E	Audio/Video Control Profile, Bluetooth SIG
VideoConferencing	uuid16 0x110F	Video Conferencing Profile, Bluetooth SIG
Intercom	uuid16 0x1110	See Intercom Profile, Bluetooth SIG.
Fax	uuid16 0x1111	See Fax Profile, Bluetooth SIG.
HeadsetAudioGateway	uuid16 0x1112	See Generic Access Profile, Bluetooth SIG.
WAP	uuid16 0x1113	See Interoperability Requirements for Bluetooth as a WAP,

.2.2

		Bluetooth SIG.
WAP_CLIENT	uuid16 0x1114	See Interoperability Requirements for Bluetooth as a WAP, Bluetooth SIG.
PANU	uuid16 0x1115	The Personal Area Networking profile for Bluetooth, Bluetooth SIG
NAP	uuid16 0x1116	The Personal Area Networking profile for Bluetooth, Bluetooth SIG
GN	uuid16 0x1117	The Personal Area Networking profile for Bluetooth, Bluetooth SIG
DirectPrinting	uuid16 0x1118	See Basic Printing Profile, Bluetooth SIG
ReferencePrinting	uuid16 0x1119	See Basic Printing Profile, Bluetooth SIG
Imaging	uuid16 0x111A	[IMAGING]
ImagingResponder	uuid16 0x111B	[IMAGING]
ImagingAutomaticArchive	uuid16 0x111C	[IMAGING]
ImagingReferencedObjects	uuid16 0x111D	[IMAGING]
Handsfree	uuid16 0x111E	Handsfree Profile, Bluetooth SIG
HandsfreeAudioGateway	uuid16 0x111F	Handsfree Profile, Bluetooth SIG
DirectPrintingReferenceObjectsService	uuid16 0x1120	See Basic Printing Profile, Bluetooth SIG
ReflectedUI	uuid16 0x1121	See Basic Printing Profile, Bluetooth SIG
BasicPrinting	uuid16 0x1122	See Basic Printing Profile, Bluetooth SIG
PrintingStatus	uuid16 0x1123	See Basic Printing Profile, Bluetooth SIG
HumanInterfaceDeviceService	uuid16 0x1124	See Human Interface Device, Bluetooth SIG

.2.3

HardcopyCableReplacement	uuid16 0x1125	See Hardcopy Cable Replacement Protocol, Bluetooth SIG
HCR_Print	uuid16 0x1126	See Hardcopy Cable Replacement Protocol, Bluetooth SIG
HCR_Scan	uuid16 0x1127	See Hardcopy Cable Replacement Protocol, Bluetooth SIG
Common_ISDN_Access	uuid16 0x1128	See CAPI Message Transport Protocol, Bluetooth SIG
VideoConferencingGW	uuid16 0x1129	See Video Conferencing Profile (VCP), Bluetooth SIG
UDI_MT	uuid16 0x112A	[UID]
UDI_TA	uuid16 0x112B	[UID]
Audio/Video	uuid16 0x112C	See Video Conferencing Profile (VCP), Bluetooth SIG
SIM_Access	uuid16 0x112D	[SAP]
PnPInformation	uuid16 0x1200	Bluetooth Device Identification, Bluetooth SIG
GenericNetworking	uuid16 0x1201	n/a
GenericFileTransfer	uuid16 0x1202	n/a
GenericAudio	uuid16 0x1203	n/a
GenericTelephony	uuid16 0x1204	n/a
UPNP_Service	uuid16 0x1205	[ESDP] and possible future profiles.
UPNP_IP_Service	uuid16 0x1206	[ESDP] and possible future profiles.
ESDP_UPNP_IP_PAN	uuid16 0x1300	[ESDP]
ESDP_UPNP_IP_LAP	uuid16 0x1301	[ESDP]
ESDP_UPNP_L2CAP	uuid16 0x1302	[ESDP]

.3 APPENDIX C

SDL symbols

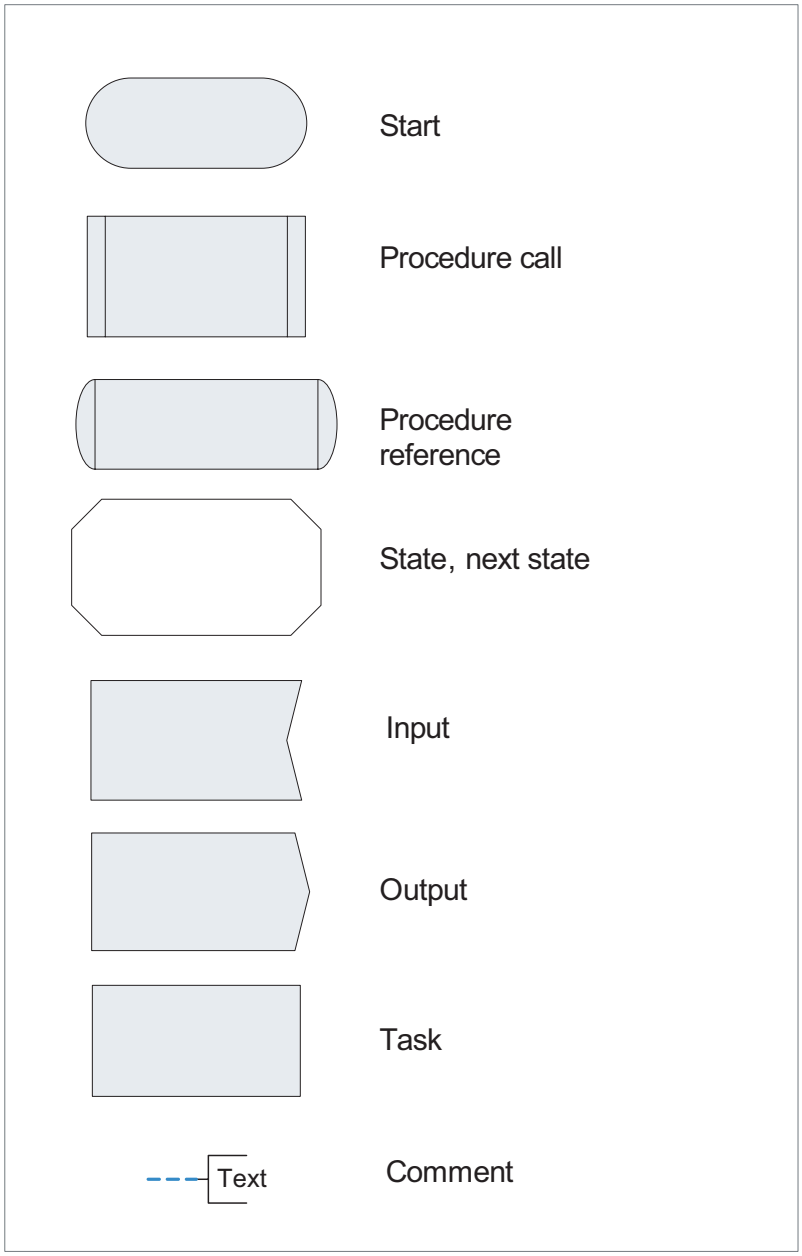

Figure .1: SDL symbols [39]