MOWAHS - Optimised XML Management in Mobile Environments

Tobias Melcher

Course teacher: Alf Inge Wang
Supervisor: Carl-Fredrik Sørensen

Norwegian University of Science and Technology
Dept. of Computer and Information Science
Fall 2004
Abstract

Mobile computing becomes more and more popular. The heterogeneity and diversity in mobile environments highly motivates the need for a framework supporting communication in such environments. We have chosen the eXtensible Markup Language (XML) for data transfer due to its support for heterogeneity and widespread adoption. However, the use of XML is resource intensive and does not combine well with limited resources associated with mobile computing.

This report looks at the properties of XML and identifies several issues which may be addressed to allow optimisation. It looks at the concepts of mobile and wireless computing and identifies several challenges connected with these environments. Further the report looks at different technologies which might be relevant in an implementation. The report also looks at research related to the topic of XML optimisation, including a number of methods for XML compression.

A number of general requirements which need to be satisfied by a successful model enabling optimisation of XML documents are identified and a model addressing these requirements is suggested. The model’s main part is a proxy server responsible for optimising XML documents by deciding how this may be done based on a set of heuristic rules. Further, the some of the functionality required by the model is analysed and a set of functional and non-functional requirements is set up. An architecture for a prototype system is suggested based on these requirements, and the prototype is partly implemented.
Preface

This project report presents the results of the work done by Tobias Melcher in the course TDT4735 Software Engineering, Depth Study at the Department of Computer and Information Science (IDI), Norwegian University of Science and Technology (NTNU), during Autumn 2004.

This report is a result of the Mobile Work Across Heterogeneous Systems (MOWAHS) project, funded by the Norwegian Research Council and conducted as a joint effort by the software engineering group and the database research group at NTNU.

I would like to take this opportunity to thank my supervisor, PhD Fellow Carl-Fredrik Sørensen, for providing valuable guidance and feedback through all phases of this project. I would also like to thank my course lecturer, assistant professor Alf Inge Wang for useful input and support.

Tobias Melcher
Oslo, November 2004
# Contents

## I Introduction

1 Introduction ................................. 5  
  1.1 Motivation ............................... 5  
  1.2 Project context .......................... 6  
  1.3 Problem definition ..................... 7  
  1.4 Project outline ......................... 7  

## II Prestudy

3 The Extensible Markup Language (XML) .......... 15  
  3.1 XML elements and tags ................... 15  
  3.2 XML validation and well-formedness ........ 16  
  3.3 Namespaces ............................... 17  
  3.4 XML parsing ............................... 17  
    3.4.1 The Document Object Model (DOM) ...... 18  
    3.4.2 The Simple API for XML (SAX) .......... 18  
  3.5 Applications of XML ..................... 19  
  3.6 Challenges and benefits using xml ........ 19  

4 Basic concepts ................................ 21  
  4.1 Mobile devices ............................ 21  
    4.1.1 Challenges for Mobile Devices .......... 22  
  4.2 Wireless networking ...................... 22  
    4.2.1 Challenges for Wireless Networks ...... 23  
    4.2.2 Wireless network classification ....... 24
## CONTENTS

4.3 Distributed systems .............................................. 25

5 Relevant technologies ........................................... 27
  5.1 Middleware ...................................................... 27
    5.1.1 Characterising middleware systems ..................... 28
    5.1.2 Middleware for mobile distributed systems ............. 29
  5.2 Web services .................................................. 30
    5.2.1 Communication protocols ................................ 30
    5.2.2 Service description ....................................... 31
    5.2.3 Service discovery ......................................... 31
  5.3 Development platforms for mobile devices ................. 31

6 State-of-the-Art survey .......................................... 33
  6.1 Cost of XML parsing ........................................... 33
  6.2 XML compression ............................................... 34

III Own contribution .............................................. 37

7 A model enabling XML optimisation ................................ 39
  7.1 Motivating scenario .......................................... 39
  7.2 Model requirements .......................................... 40
    7.2.1 XML properties .......................................... 40
    7.2.2 Behaviour and support ................................... 41
    7.2.3 Services provided ....................................... 41
  7.3 Overall description of the model ............................ 42
    7.3.1 Optimising an XML document ............................ 43
    7.3.2 Defining heuristic rules ................................ 44

8 Requirements ...................................................... 47
  8.1 Overall system functionality ................................. 47
  8.2 System requirements .......................................... 59
    8.2.1 Functional requirements for the proxy-server ........ 59
    8.2.2 Functional requirements for the mobile client ....... 60
    8.2.3 Non-functional requirements ............................. 61
  8.3 Summarizing the requirements ................................ 62
    8.3.1 Mapping the use-cases to the requirements .......... 62
    8.3.2 Prioritising the requirements ......................... 62
# CONTENTS

9 The prototype system 65
   9.1 Architecture and design 65
      9.1.1 Architectural goals and considerations 65
      9.1.2 Overall architecture 67
      9.1.3 Logical view 69
      9.1.4 Process view 72
   9.2 Implementation 73
      9.2.1 Communication framework 73
      9.2.2 The mobile client 74
      9.2.3 The proxy server 75
      9.2.4 The content provider 75
      9.2.5 Remaining work 76

IV Evaluation and conclusion 77

10 Evaluation and discussion 79
   10.1 Discussion 79
      10.1.1 The use of a central proxy server 79
      10.1.2 Performance issues 80
      10.1.3 The development of heuristic rules 81
      10.1.4 The mobile client 81
   10.2 Evaluation 81
      10.2.1 Answering the research questions 81
      10.2.2 Meeting the requirements for the prototype 83

11 Conclusion and further work 85
   11.1 Conclusion 85
      11.1.1 Important themes 85
      11.1.2 Contributions of this report 86
   11.2 Future work 86
      11.2.1 Finish the prototype implementation 86
      11.2.2 Extensions of this work 86
      11.2.3 Further research 87

A Glossary 95

B XML technologies 101
   B.1 The Extensible Stylesheet Language (XSL) 101
      B.1.1 XSL Transformations (XSLT) 101
      B.1.2 XPath 102
## List of Figures

3.1 DOM parsing .............................................. 18
4.1 Network with mobile devices ................................. 23
4.2 Nomadic network ........................................... 25
4.3 Mobile Ad-hoc Network ................................... 26
5.1 Middleware characterisation ................................ 28
7.1 Overall system ............................................. 43
8.1 Use-case diagram: System initialisation ...................... 48
8.2 Use-case diagram: XML optimisation ........................ 50
8.3 Use-case diagram: XML client receive ...................... 53
8.4 Use-case diagram: XML client send ........................ 56
9.1 Overall prototype architecture ............................... 68
9.2 XML document filtering ................................... 69
9.3 The top level class diagram ................................ 70
9.4 The parser .................................................. 70
9.5 The transformer ............................................ 71
9.6 The heuristic .............................................. 71
9.7 Process view ............................................... 72
D.1 Sony Ericsson’s T610 ...................................... 109
D.2 HP’s iPAQ h4155 Pocket PC ................................ 110
D.3 Sony Ericsson’s P910i ..................................... 110
E.1 The J2ME architecture .................................... 112
E.2 .NET Compact Framework .................................. 114
E.3 BREW device architecture .................................. 116
LIST OF FIGURES
List of Tables

3.1 XML example ........................................ 16
3.2 XML attributes example .............................. 16
3.3 XML namespace example .............................. 17

5.1 Comparison of platforms for mobile devices .......... 32

7.1 General requirements .................................. 40

8.1 Use-case: System initialisation .......................... 50
8.2 Use-case: XML optimisation ............................ 51
8.3 Use-case: Parsing a transformed XML document ......... 52
8.4 Use-case: Sending an XML to the client ................ 54
8.5 Use-case: Suspend communication ...................... 55
8.6 Use-case: Receiving an XML from the client ............ 57
8.7 Use-case: Suspend communication ...................... 58
8.8 Mapping use-cases to functional requirements ........... 63
8.9 Prioritised functional requirements .................... 63
8.10 Prioritised non-functional requirements ................. 64

9.1 Response message ...................................... 73
9.2 Request message ....................................... 74

10.1 Meeting the functional requirements .................. 84

B.1 XML document example ................................ 102
B.2 XML stylesheet ......................................... 103
B.3 Result of applying a XML stylesheet .................... 103

F.1 Use-case: Explanation of the textual use-cases ........ 117
LIST OF TABLES
Part I

Introduction
Chapter 1

Introduction

The theme of this project report is better support for XML in mobile environments. The focus is on developing a model and a prototype which support optimisation of XML documents by saving processing and transfer time. This chapter outlines the motivation for developing such a model and why the use of XML is practical in a mobile setting. It describes the context of this project and defines the problems addressed. Lastly, it describes the structure of this report, serving as a roadmap for the reader.

1.1 Motivation

Computers play an increasing role in our everyday life as we begin the 21st century. Everywhere around us there is some kind of computational support, in the house, the car, our pockets, and even inside our bodies. This is the trend which is referred to as pervasive or ubiquitous computing. As early as 1991, Mark Weiser introduced his vision on ubiquitous computing. He foresaw that computers would play a bigger role in our life, emerging into every human task becoming invisible to our common awareness [62].

Mobility is an important property of pervasive computing. Sales rates for laptops and other mobile devices are increasing and in 2004 more than 3 billion users have mobile phones. Supporting mobility and hence wireless communication is necessary to further incorporate computational support into our lives. Unfortunately, the infrastructure for this kind of support is immature, and an immense diversity of tools and equipment exists. Support for communication across heterogeneous systems is therefore necessary.

The eXtensible Markup Language (XML, see 3) is a common format for
information exchange which supports heterogeneity very well. One reason for this is that XML is based on ASCII characters, hence avoiding difficulties such as byte ordering and floating-point representation associated with binary information exchange. An example of the usefulness of ASCII-based messaging protocols is the Simple Mail Transfer Protocol (SMTP), developed by Jon Postel during the early 80s. SMTP is still widely deployed.

Unfortunately, XML parsing is a very resource intensive operation, and with the limited capacity - in terms of processing speed, memory, bandwidth, and power - on mobile devices, this limits the XML messages that can be received and interpreted within acceptable time and resource usage. A research challenge is to extend the range of XML messages that can be effectively utilized by mobile devices.

1.2 Project context

The work presented in this report has been conducted in the context of the MOWAHS project (Mobile Work Across Heterogeneous Systems) [41]. It is a research project conducted at NTNU and carried out jointly by the software engineering and database research groups at IDI. The Norwegian Research Council has funded the project with NOK 5 Million over 4 years.

MOWAHS can be divided into two parts. These include “process support for mobile users using heterogeneous devices (PC, PDA, mobile phones)” and “support for cooperating transactions/workspaces holding work documents”. This work is motivated by the fact that the usage of mobile devices with Internet connection keeps growing. Some companies have adopted this technology creating virtual organisations, with people distributed over several locations. Unfortunately, the current infrastructure and tools supporting virtual organisations are immature.

The MOWAHS project has three main goals:

G1 Helping to understand the work processes in virtual organizations and, by assessing them, how they can be improved.

G2 Provide a flexible and common work environment for a variety of electronic devices where work processes can be executed and shared together with their artifacts.
CHAPTER 1. INTRODUCTION

G3 Distribute the research results.

This thesis is in context of the first part of MOWAHS, focusing on communication between heterogeneous devices.

1.3 Problem definition

During the first part of the project the following problem definition was defined in cooperation with the project supervisor:

The purpose of this project is to explore the possibilities of transforming XML to optimise processing and transfer time. The project should develop models for communication between mobile devices based on XML messages and look at the possibility to develop a set of heuristic rules deciding when to transform these messages. A prototype system and testing environment supporting transformation of XML needs to be set up. This system should preferably consist of a supporting server and mobile clients.

We believe a considerable amount of resources can be saved by optimising XML messages for mobile clients, enabling mobile clients to handle larger XML documents and increasing the scalability of wireless networks incorporating such clients. We also believe that optimising XML messages not necessarily gives better results in all cases. If the XML is short enough, a transformation might even yield a negative cost-benefit ratio. Therefore, we wish to see if it is possible to develop a set of heuristic rules which can decide when a transformation is worth while. These theories need to be tested, and we therefore need to set up a platform for testing and implementing the heuristic rules.

Chapter 2 dives deeper into the problem and outlines a methodology for solving it.

1.4 Project outline

This report is organized into eleven chapters which are divided into four parts as follows:

- Part I provides the setting for this report.
  - Chapter 1 (this chapter) provides the background and context, and defines the problem addressed.
CHAPTER 1. INTRODUCTION

– Chapter 2 further outlines the addressed problem and describes the methodology used in the report.

• Part II reviews relevant issues and work in the field this report addresses.
  – Chapter 3 provides a description of XML.
  – Chapter 4 describes the concepts of mobile devices and wireless networking.
  – Chapter 5 outlines technologies relevant to this work.
  – Chapter 6 provides a survey of solutions addressing some of the aspects of the problem definition.

• Part III describes a model and a prototype solving the problem addressed.
  – Chapter 7 proposes a model for XML optimisation in mobile environments based on some general requirements.
  – Chapter 8 summarize the requirements which need to be addressed in a system based on the model described in Chapter 7.
  – Chapter 9 describes a prototype system based on the model proposed in Chapter 7 and the requirements in Chapter 8.

• Part IV evaluates and concludes the work done in this report.
  – Chapter 10 is a discussion and an evaluation of the work done.
  – Chapter 11 concludes the project and provides some suggestions for further work.
Chapter 2

Research agenda

The goal of this chapter is to provide a basis and a methodology for the work presented in this report. First, Section 2.1 defines a main research question and further outlines this question with more specific questions. Section 2.2 presents some research hypothesis connected to the main research question, and finally, Section 2.3 gives a methodology for approaching the main research question.

2.1 Research questions

The main question that this report aims to answer is:

How can we build a system based on mobile clients and supporting servers which optimise processing and transfer time of XML-based information between these?

This question leads to the definition of the following sub questions, determining further work:

RQ1 Current state: Are there any efforts which can answer or help answer the main question?

RQ2 Requirements: What is the nature of XML messages to mobile devices and what kind of hardware restrictions exist? Then, what requirements do these impose on the optimisation in handling XML messages?

RQ3 Solution: What is needed to provide a foundation for meeting the requirements? What kinds of technologies are suitable for an implementation?
CHAPTER 2. RESEARCH AGENDA

RQ4 Evaluation: How well does the solution solve the problem given in the main question?

Other questions might include:

- Can we build a set of heuristic rules determining when to transform XML messages before sending them to mobile devices?
- Is transformation the only means to optimisation? Are there other means?
- What kind of transformation gives better results concerning processing and communication?
- What are the bottlenecks in XML message handling?

2.2 Research hypothesis

A couple of research hypothesis can be deducted from the main research question (see 2.1) and the problem definition (see 1.3);

H1 Optimising XML messages yields a saving in transfer (T) time versus extra processing (P) time needed to handle the XML message.

\[ T_{t,b} + P_{t,b} > T_{t,a} + P_{t,a} \]

\( t_{b} \) - time needed before transformation
\( t_{a} \) - time needed after transformation

H2 It is possible to develop heuristic rules deciding when we can profit on a transformation of an XML message.

H3 A system can be developed yielding a performance gain in XML document passing of more than 10%.

H4 A reduction of XML size can increase the utilisation of a wireless network and thereby increase the scalability.

2.3 Research methodology

The Oxford Dictionary and Thesaurus [55] defines research as:
CHAPTER 2. RESEARCH AGENDA

“the systematic investigation into and study of materials, sources, etc., in order to establish facts and reach new conclusions and an endeavour to discover new or collate old facts, etc., by the scientific study of a subject or by the course of critical investigation”.

This view outlines the importance of a systematic approach. Hence, setting up a methodology and sticking to it is of utmost importance. The lack of methodology has been a problem in the early days of software engineering. Glass [25] summarizes four research methods;

The scientific method A model is built based on observations done on the real world.

The engineering method Current solutions are studied, changes are proposed and then evaluated.

The empirical method Empirical methods are used to evaluate a proposed model.

The analytical method Empirical observations are compared to a formal theory.

The method applied in this report can best be classified as an engineering method. The problem given in Section 1.3 is approached by first surveying and analysing the current situation and related solutions in Part II. Part II will reveal if there exist any suitable solutions to the problem at hand, as well as a number of requirements which need to be addressed. These findings will lead to the design of a model, and the development of a prototype system, described in part III. Part IV evaluates the findings in this report.
Part II

Prestudy
Chapter 3

The Extensible Markup Language (XML)

XML can briefly be described as “a clearly defined way to structure, describe, and interchange data.” [3]. Its popularity has made it the prime standard for exchanging data on the web [22]. It is a formal recommendation from the World Wide Web Consortium (W3C) [61], and it is derived from the Standard Generalized Markup Language (SGML).

This chapter describes the structure and some properties of XML documents. It also explains how information is extracted from XML documents, and it provides some applications of XML. At last some benefits and challenges connected with the use of XML are summarized.

3.1 XML elements and tags

XML is a meta language, and as a meta language XML defines a generic syntax for mark-up symbols (or tags). In XML, tags usually come in pairs\(^1\), comprising of start- and end-tags. These tags are unlimited and self defining, and they are used to describe the content of a file or document. A tag-pair surrounds an element, which might be a single value, or a nesting of elements. Table 3.1 shows how tags are used. XML does not define any tags on its own, but the XSLT and XHTML are examples of mark-up languages implemented using XML.

XML elements can also use attributes to express information. An attribute consists of a name and an appurtenant value separated by an equals

\(^1\)Elements with no content, or empty elements, can be represented by a single tag, starting with \(<\) and ending with \(/\)
CHAPTER 3. THE EXTENSIBLE MARKUP LANGUAGE (XML)

Table 3.1: A well-formed XML document, which describes the content of a wardrobe.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<wardrobe>
  <shoe>
    <brand>Nike</brand>
    <colour>white</colour>
  </shoe>
  <jacket>
    <brand>Blend</brand>
    <colour>brown</colour>
  </jacket>
</wardrobe>
```

Table 3.2: A XML document using attributes to express information about elements.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<wardrobe>
  <shoe brand="Salomon" colour="brown" />
  <jacket brand="Levis" colour="red" />
</wardrobe>
```

3.2 XML validation and well-formedness

An XML document can be encapsulated in a tree structure and it is well-formed if it adheres to certain syntax guidelines concerning the tree structure, termination of elements etc. (see Table 3.1 for an example). The ability of being well-formed is important since XML represents structured data,
CHAPTER 3. THE EXTENSIBLE MARKUP LANGUAGE (XML)

and since corruption of such may have unwanted consequences. An XML document can be further validated against either a Document Type Definition (DTD), an XML Schema or some other schema language, which provide a means for defining legal building blocks of XML documents.

3.3 Namespaces

Namespaces provide a means to distinguish between elements and attributes from different XML vocabularies that have the same name [27]. This is done by attaching a prefix to the elements or attributes name. This prefix is mapped to a Uniform Resource Identifier (URI) \(^2\) by an xmlns:prefix attribute. Each element and attribute connected to the same URI belongs to the same namespace. Table 3.3 shows a simple example of how namespaces may be used.

```
<?xml version="1.0" encoding="UTF-8"?>
<people>
  <phonebook:person xmlns:phonebook="http://www.phonebook.com/xml">
    <phonebook:name>John Smith</phonebook:name>
    <adress>203 Imagineville, California</adress>
  </phonebook:person>
  <medical:person xmlns:medical="http://www.medical.com/xml">
    <medical:name>Jack Unlucky</medical_record:name>
    <injury_description>Left leg fractured</injury_description>
  </medical:person>
</people>
```

Table 3.3: An XML document using namespaces.

3.4 XML parsing

Parsing of XML documents is necessary to extract information from the document. The main responsibilities of the parser include retracting necessary information, check for logical errors within the structure, and to validate the XML document if required. Three types of parsers exist:

\(^2\)See http://www.ietf.org/rfc/rfc2396.txt for elaborating information on URIs.
CHAPTER 3. THE EXTENSIBLE MARKUP LANGUAGE (XML)

The model parser: Reads through the whole XML document and creates a representation of it in memory.

The push parser: Reads through the whole XML document and notifies a listener whenever certain parts of the document are encountered.

The pull parser: Reads through a smaller part of the document. New parts are read as they are requested from the application using the parser.

The two most common models for XML parsing include the Document Object Model and the Simple API for XML, which both are language-independent APIs for accessing the contents of an XML document. A number of DOM- and SAX-parsers have been developed.

3.4.1 The Document Object Model (DOM)

DOM provides navigational access to an XML document and can be classified as a model parser. It reads through the entire XML document and builds a tree-structure of objects for traversal and manipulation (see Figure 3.1). This structure resides in the computer's temporary memory, which implies a large memory consumption when parsing larger documents. The DOM structure is useful for applications requiring random access to different parts of the document, and for applications which need to update the document quickly.

![Figure 3.1: This figure depicts how a DOM parser transforms the XML document into a tree-structure of objects.](image)

3.4.2 The Simple API for XML (SAX)

SAX provides sequential access to an XML document and can be classified as a push parser. It is an event-based API, which means that as the XML
document is read, the parser tells your program whenever it sees a start-
tag, end-tag, character data, or a processing instruction [27]. Unfortunately, the sequential reading disallows random access as provided by DOM, on the other hand the entire XML document does not have to reside in memory. This makes SAX useful when reading large documents and random access is not required.

### 3.5 Applications of XML

XML has been widely adopted by the industry as a format for storing and interchanging structured data, and it has thereby become the prime standard for data exchange on the World Wide Web [22]. XML is used in a wide range of applications, ranging from databases to message exchange in distributed systems.

A number of languages based on XML have appeared. The Extensible HyperText Markup Language (XHTML) [46] is a good example. XHTML can be described as a reformulation of HTML in XML. It is a family of document types and modules which reproduces and extends HTML 4.

XML has also been used as a basis in the Web services framework\(^3\), which provides a mean to allow interaction between incompatible applications on the web.

### 3.6 Challenges and benefits using xml

XML is verbose and redundant, this in turn might introduce documents which are larger than they need to be. Handling large XML documents in a mobile setting with limited resources, such as low bandwidth and CPU power, is a challenge. Developing a means for this is the main goal of this project.

The main advantage of using XML in a mobile and often heterogeneous setting is how well it handles the heterogeneity. XML is based on ASCII characters, and problems associated with binary data transmission do not exist. This is also the reason why XML is interesting in the setting of the MOWAHS project as a format for data exchange.

\(^3\)Section 5.2 gives a closer description of Web services.
CHAPTER 3. THE EXTENSIBLE MARKUP LANGUAGE (XML)
Chapter 4

Basic concepts

Forman et al. [20] defines mobile computing as “the use of a portable computer capable of wireless networking”. Connection to a network greatly enhances the utility of any device, and with mobile computing the opportunities are immense. Continuous access to information and services, work “anywhere, anytime”, and increased awareness on multiple areas, are some of the opportunities the combination of wireless networking and mobile devices brings to us. Unfortunately, they also introduce a number of technical challenges.

Section 4.1 describes the characteristics of mobile devices and outlines some of the challenges involved both in terms of software and hardware. Section 4.2 describes issues regarding wireless communication and networking. It also describes how wireless networks might be characterised. At last, Section 4.3 describes distributed systems and provides a means for their characterisation.

4.1 Mobile devices

Mobile devices come in many different shapes and deliver a broad range of services to end users. Most commonly known are probably the mobile phone, the personal digital assistant (PDA)\textsuperscript{1}, and the laptop. A common factor for all of them is that they can be carried around and hence be moved to different locations with ease. The mobility and the increasing requirements for computational power introduce a number of challenges.

\textsuperscript{1}See Appendix D for a closer description and some examples of mobile phones and PDA’s
CHAPTER 4. BASIC CONCEPTS

4.1.1 Challenges for Mobile Devices

The ability to be portable brings a number of constraints to mobile devices. These constraints include limits to size, weight and power consumption. [20] lists the design pressures put on mobile devices:

**Low power:** Mobile devices depend on batteries, hence battery lifetime constrains the use of such devices. Unfortunately, battery weight is considerable, and reducing their size might lead to frequent recharging, which is undesirable. Another problem is the demand for computational power and additional features such as cameras. These demands increase the need for energy. Design solutions include smaller and more efficient chips - reduced capacitance and voltage - and power management.

**Risks to data:** Mobile devices have an increased risk of physical damage, unauthorized access, loss and theft. Loss of data and privacy breaches therefore become more likely. Actions to overcome these problems include removal of essential data from the device, authentication of users, encryption of sensitive data, and backup of essential data.

**Small user interface:** Size and energy constraints on mobile devices impose small user interfaces. Presenting information on small user interfaces, and selecting reasonable ways for information input are challenging. Solutions include goggles displaying data and input devices such as pens for pointing and writing.

**Small storage capacity:** As with user interfaces, the size and energy constraints limit the storage space on mobile devices. Solutions include compression and remote storage.

**Limited CPU:** The CPU’s computational power is limited by the power requirements put on mobile devices. Remote computation of resource intensive operations might be a solution.

4.2 Wireless networking

A wireless network connection is essential to enable mobile computing. A number of technologies enabling wireless networking exist\(^2\). These technologies are diverse in how they handle communication, and which services they

\(^2\)See C for a description of some wireless technologies
provide. On the physical level, either radio waves, sound, or pulsing infrared light is used for communication [20]. The coverage area for a single transceiver varies greatly, as well as their power consumption.

Figure 4.1 shows an example of a network where mobile devices are connected to stationary devices through both wireless and cable connections.

Figure 4.1: This network is an example of how mobile devices can interconnect both wireless and through cable.

The environment puts a lot of strain on wireless networks; it disturbs and blocks signals, and it introduces noise and echo. These environmental challenges increase the risk for disconnection and reduce the available bandwidth. In other words, wireless communication faces a number of challenges which wired communication avoids.

4.2.1 Challenges for Wireless Networks

Wireless communication can be characterized by low bandwidth, frequent disconnections and higher error rates. These characteristics impose a number of challenges to wireless networks identified by [20]:

23
CHAPTER 4. BASIC CONCEPTS

Disconnection: Wireless networks are exposed to frequent disconnections due to the environment, the users movement, and the scarce resources in mobile devices. To overcome these challenges, the asynchronous communication paradigm can be introduced.

Low bandwidth: Wireless networks deliver lower bandwidth, often by an order of magnitude, than wired networks. Hence solutions exploiting available bandwidth, these might include compression and buffering. Another obvious solution is to improve the network capacity.

High bandwidth variability: Wireless networks tend to vary in their available bandwidth. Applications can face this problem by either assuming a high bandwidth, or a low bandwidth, or adapt to the available resources.

Heterogeneous network: As mobile devices move out of range of one network transceiver and switch to another, transmission speeds and protocols might change. Interfaces might also need to be changed, for instance when moving from indoors to outdoors.

Security risks: Signals sent in a wireless network are not protected by cables, and hence unwanted eavesdropping might occur more frequently. Solutions include encryption by either software or hardware.

4.2.2 Wireless network classification

Wireless networks can basically be categorised into networks with and without any infrastructure. Combinations of these two exist and will probably become more popular in the near future.

Nomadic networks

In nomadic networks, the wireless part of the network is connected to what we can refer to as a wired backbone. This backbone can provide services such as data storage and resource intensive computation. Figure 4.2 depicts an example of a wireless network with an infrastructure.

Mobile Ad-hoc Networks (MANET)

MANETs are characterized by their lack of infrastructure. They allow communication between nodes (mobile devices) without a centralized administration, and therefore depend on each node to act as both end-user and router.
Figure 4.2: Nomadic network

[65, 4]. Figure 4.3 shows an example of MANET.

The handling of data and routing between nodes have gained a lot of re-
search attention, however there are other problems such as lack of scalability
and large power demand.

4.3 Distributed systems

A distributed system is a collection of components, distributed over several
hosts connected through a network. Middleware\textsuperscript{3} is used to facilitate interac-
tion between the distributed components and the network operating system.
Mascolo et al. [40] identify three concepts which can be used in the classifi-
cation of distributed systems; the concept of device, network connection and
execution context.

\textsuperscript{3}See 5.1 for a closer description of middleware.
A device can be either mobile or fixed. While fixed devices generally are powerful, mobile devices have limited capabilities. Network connections can be either intermittent or permanent. Permanent connections provide continuous connection usually with high bandwidth. Wireless connections, such as Bluetooth, GSM and WiFi (see C.1 for a closer description), are characterized by variable performance, such as variable bandwidth and risk of disconnection. Hence the term intermittent is used for wireless connections.

The type of execution context can be either static or dynamic. In a static context, conditions influencing the system, such as internal resources, quality of network connection, location and connected hosts, usually do not vary. On the other hand, these conditions may change rapidly and unannounced in a dynamic context.

[40] distinguish traditional distributed systems, nomadic distributed systems and ad-hoc mobile distributed systems. Where traditional distributed systems consist of fixed devices with permanent network connections and a static execution environment, ad-hoc mobile distributed systems consist of mobile devices with intermittent network connections executing in a dynamic environment. While nomadic distributed systems represent a compromise between these two consisting of a set of mobile devices and a fixed infrastructure.
Chapter 5

Relevant technologies

One of the central parts of this project is to build a prototype of a system enabling optimisation of XML messaging. Hence, a knowledge base of relevant technologies for development is needed. This chapter provides an overview of some available technologies relevant for the development of a prototype enabling optimisation of XML messages.

5.1 Middleware

Vaughan-Nichols [60] describes middleware as “a software layer that resides between programs, OSs, hardware platforms, and communication protocols”. The goal of middleware systems is to ease the developers’ task by hiding complexities. Some of the benefits provided by middleware include:

- Reduces the cost involved with application development.
- Enables portability of applications across platforms.
- Enables communication between different applications.

A number of middleware technologies intended for distributed systems have been developed and successfully used in the industry. CORBA [14], Java RMI [30] and Microsoft’s DCOM [16] are examples of such middleware. Unfortunately, these systems are not necessarily suited for a mobile setting. In particular, the interaction primitives, such as remote procedure calls and remote method invocations, assume a high bandwidth connection and constant availability of components [39]. However, in mobile systems low bandwidth and unreachability tend to be the norm.
This section describes a characterisation scheme for middleware and gives a short summary of some of the available solutions.

### 5.1.1 Characterising middleware systems

The reference model depicted in figure 5.1 is introduced in [40] to enable middleware comparison.

![Diagram of middleware characterisation]

**Figure 5.1: Characterisation of middleware systems (Adopted from [40])**

Middleware systems can be distinguished through how well the non-functional requirements, such as the reliability on providing services, are met. The type of computational load denotes the amount of resources needed to meet these non-functional requirements. The second way to distinguish middleware is through which communication paradigm they support, synchronous or asynchronous. The last characterizing element is on what type of context representation the middleware provides to the applications, transparency or awareness.

Through this framework, the relationship between the physical structure of nomadic and ad-hoc systems and the characteristics of associated middleware can be identified:
CHAPTER 5. RELEVANT TECHNOLOGIES

- Mobile devices impose the need for a light computational load. The limited resources of mobile devices justify this requirement.

- Intermittent network connections impose the need for asynchronous communication. In order to allow interaction between components in an environment where network connections are opportunistic, an asynchronous form of communication is necessary.

- Dynamic context imposes the need for awareness. The high variability in a mobile environment requires context aware techniques to optimise the behaviour of the application and middleware.

5.1.2 Middleware for mobile distributed systems

The mobile environment imposes a number of challenges on middleware. Researchers are therefore actively working to design middleware suited for this environment, and a number of systems have been proposed.

Allowing mobile applications to be aware of their context, can help them to optimise their behaviour and better utilize their available resources. Eliassen et al. [19] introduced the term reflective middleware, where some of the transparency introduced by middleware is given up to allow greater awareness for applications. [40] presents some approaches where reflection is used in mobile environments to offer context-awareness and mechanisms for adaptation. The Universally Interoperable Core (UIC) [56], Gaia [10], and an event notification architecture [63] all share the same main idea, which is to offer the ability to change the middlewares behaviour based on the knowledge of changes in context.

Middleware systems, allowing service discovery, can be useful in a mobile setting as they, amongst other, allow mobile clients to move resource intensive operations away from the device. Yuan and Arbaugh [65] present a scheme for service discovery in small or medium sized MANET’s, which allows clients to locate and evaluate services without the need of a directory agent or a central registry. Chen et al. [11] describe a network service platform for mobile pervasive computing. Their goal is to make “full use of available resources in the surrounding so as to augment capabilities and to save scanty resources of mobile hosts”.

Supporting disconnected operations and data sharing is a major issue. Solutions try to maximize the availability of data by giving users access
CHAPTER 5. RELEVANT TECHNOLOGIES

to replicas. XMIDDLE [39] is an XML based middleware for peer-to-peer computing allowing “on-line collaboration, off-line data manipulation, synchronization, and application dependent reconciliation”. Other data sharing middleware includes Coda\textsuperscript{1} [49], Oddysey [48], and Bayou [17].

Researchers have also adapted traditional middleware to the mobile setting, first of all to allow interaction between mobile devices and existing fixed distributed networks. However, the synchronous connection paradigm in these middleware systems is a problem.

5.2 Web services

A number of similar services provided by multiple companies and based on that incompatible applications exist on the Internet. To allow interaction between these applications, a framework is needed. Web services provide such a framework “built on top of existing Web protocols and based on open XML standards” [15].

The Web services framework provides a modular approach for interaction between online applications. It defines a standardized mechanism for describing, locating and communicating, turning applications into accessible Web service components. The following describes the three main parts of the framework.

5.2.1 Communication protocols

The communication protocol enables communication among Web services. As the Web is a heterogeneous environment, mechanisms enabling communication hence need to be platform independent and international [15]. XML covers these requirements, and using it to build a communication protocol is a natural approach.

The Simple Object Access Protocol (SOAP) [26] is a W3C specification. It is based on XML and provides messaging and remote procedure calls (RPC). SOAP does not define its own transport protocol, instead it uses existing ones such as the HyperText Transfer Protocol (HTTP).

\textsuperscript{1}The Coda Filesystem relies on a fixed core infrastructure and is not especially developed for mobile environments.
CHAPTER 5. RELEVANT TECHNOLOGIES

XML-RPC is a communication protocol which might be used instead of SOAP. XML-RPC was initially developed by Dave Winer and it uses HTTP to relay XML messages through the network [31].

5.2.2 Service description

Achieving communication between Web service components is not enough to enable interaction. A method to tell what messages that must be exchanged is needed.

The Web Services Description Language (WSDL) [12] provides a formal description of Web services. It is an XML format which describes a Web service’s interface and provides a point of contact for the user [15].

5.2.3 Service discovery

A centralized and searchable directory is needed to enable promotion and discovery of Web services.

The Universal Description, Discovery and Integration (UDDI) [57] is a specification for Web-based registers of Web services. It provides a definition of what information about each service to be stored, and how the information is encoded, together with a query and update API [15].

The UDDI Business Registry provides a browser-accessible directory for Web services. The stored information within this registry comprises of general details, a means for categorization, and technical data.

5.3 Development platforms for mobile devices

Mobile devices entail a number of constraints which in turn puts a challenge to application development for these devices. To cope with these challenges, platforms adapted for mobile and resource-constrained devices have been developed to ease the task of the programmer. These platforms usually have a low memory footprint, which is necessary because of the limited resources available. In this section, I try to evaluate the different development plat-
CHAPTER 5. RELEVANT TECHNOLOGIES

forms for mobile devices. Table 5.1 compares some of the available platforms.

<table>
<thead>
<tr>
<th>Platform</th>
<th>Memory footprint</th>
<th>XML support?</th>
<th>Programming language(s)</th>
<th>Adoption</th>
<th>Cost (USD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>J2ME</td>
<td>Depending on profile</td>
<td>Extension</td>
<td>Java</td>
<td>200+ devices</td>
<td>Free</td>
</tr>
<tr>
<td>.NET CF</td>
<td>1500 KB</td>
<td>Yes</td>
<td>Visual Basic, .NET and C#</td>
<td></td>
<td>Free</td>
</tr>
<tr>
<td>BREW</td>
<td>150 KB</td>
<td>Extension</td>
<td>C and C++ (Java)</td>
<td>150+ devices</td>
<td>1000+</td>
</tr>
</tbody>
</table>

Table 5.1: Comparison of platforms for mobile devices

Java’s portability, support for heterogeneity, and popularity makes it a reasonable choice as the programming platform for a system aiming to optimise the processing and transfer time of XML based information.

\(^2\)Appendix E presents some of the available development platforms suited for mobile and limited devices
Chapter 6

State-of-the-Art survey

This chapter gives a brief description of some of the work done which introduce relevant ideas to this project. Section 6.1 takes a look at the cost of XML parsing, while Section 6.2 looks at different solutions reducing the size of XML documents.

6.1 Cost of XML parsing

During Fall 2003, Knutsen [32] conducted a project at NTNU in a similar context as this project. This project is actually a follow up of his work. Knutsen did some experiments regarding expensiveness of XML parsing. However, the hypothesis he worked on where self-fulfilling, it is clear that the size of an XML document has an impact on both the time and the memory needed to parse it. On top of this, most of his results cannot be classified as new, as similar results have been shown in other work such as in Maruyama et. al. [38]. The interesting and valuable part of Knutsen’s work is his proposed scheme for reducing the size of XML documents by shortening tag lengths. He also revealed that a reduction of XML documents with the proposed scheme yielded a reduction in SAX parsing cost but an increase in DOM parsing cost.

Nicola and John [43] look at real-world XML database situations where parsing is a key obstacle. They identify obstacles which must be overcome to improve XML parsing, and they analyse the performance of SAX XML parsing. Their findings include a list of the three most expensive parser components; (1) memory management, (2) attribute handling and (3) transcoding the input document encoding into UTF-16. They also show that the average parsing cost for documents less than 100k is approximately 175,000
CHAPTER 6. STATE-OF-THE-ART SURVEY

CPU instructions.

Vaaland [58] showed in his work how needed time and memory to parse large documents can be saved using a special-purpose parser. He developed a parser intended for allowing an editor to view and edit large XML documents (+40Mb).

6.2 XML compression

Reducing the size of XML data is a central part in a model for better support of XML in mobile environments. Looking at compression techniques and other solutions for size reduction is therefore important.

XML can, of course, be compressed by general purpose compressors such as gzip. However, Liefke and Suciu [36] showed in their work that their XMill tool can achieve almost twice the compression rate, at about the same speed, compared to a general-purpose compressor. XMill is probably the most renown tool for XML compression. It is a lossless compressor/decompressor, which applies three important principles; separate structure from data, group related data items and apply semantic compressors.

WAP Binary XML Format (WBXML) [37] is a binary encoding scheme designed to be used in the Wireless Application Protocol (WAP). WBXML replaces common tag and attribute names and values with tokens. It is designed to reduce the transmission size of XML documents without removing the functionality or semantic information.

Girardot and Sundaresan [24] present the Millau system for compression and streaming of XML data. Millau extends the encoding format of WBXML. It introduces a scheme where the XML document is fragmented and reordered, structure is also separated from data. The performance analysis done on the system shows that the compression rate of Millau is outperformed by gzip. However, this is justified because Millau retains the structure of the data which allows faster processing of documents. Further, the analysis shows that Millau reduces the parsing time to about 20% of a regular XML parser.

One last method for XML compression worth looking at is the SqueezeX system presented by Cannataro et. al. [9].

1 a token is a single byte
sizer/compressor for XML data, it allows (1) data aggregation and summarization through restructuring the schema of an XML document (synthesis) and (2) further compression using a set of general- and special purpose compressors. The preliminary results of their work are promising as compression rates in the area of 98% are achieved.

A general assumption is that when compression is introduced, the time needed to handle and parse the XML document would increase. However results presented in [24] indicate that the difference is negligible when using a SAX parser, [32] presents similar results.

The methods mentioned above are however unsuited in the setting of this project. These methods transform XML into binary data which removes the value of using XML in a heterogeneous environment. However, some of the principals introduced, such as separating data from structure and the synthesis of XML documents, are interesting.
Part III

Own contribution
Chapter 7

A model enabling XML optimisation

Part II reveals a number of requirements which should be addressed in a model optimising processing and transferring time of XML documents in a mobile and wireless setting. In this chapter, a generic model giving a possible solution to the main research question is described. Section 7.1 tries to illuminate the need for such a model, while Section 7.2 summarizes the requirements. Section 7.3 describes the model covering the general requirements.

7.1 Motivating scenario

To illuminate the need for adaptation and optimisation of XML documents, the following describes a fabricated case showing how XML document passing might be problematic on mobile devices.

A car manufacturer has installed a number of sensors monitoring the factory. Unfortunately, these sensors have different interfaces and are based on diverse processors. Up to now, the company has used as many monitoring stations as the number of different sensors. They have hired SensorSoft to develop a means to monitor the information provided by the sensors on one single device.

SensorSoft has decided that a practical solution will also be a mobile one, and the development team has decided to implement a monitoring application on a PDA. As this PDA monitors a number of sensors, which all share the same bandwidth, a means to filter and optimise the information exchange is
CHAPTER 7. A MODEL ENABLING XML OPTIMISATION

needed. At the same time the system needs to handle heterogeneity.

7.2 Model requirements

Given the research done in part II, some general requirements can be set up, which should be satisfied by a successful model enabling optimisation of XML documents. Table 7.1 summarizes these requirements.

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GR1 Supporting several parsers</td>
<td>The XML optimisation model should implement several parsers.</td>
</tr>
<tr>
<td>GR2 Information adaptation</td>
<td>The XML optimisation model should be able to extract and adapt information from XML documents.</td>
</tr>
<tr>
<td>GR3 Compression</td>
<td>The XML optimisation model should support a lossless compression of XML documents.</td>
</tr>
<tr>
<td>GR4 Tailorable set of heuristic rules</td>
<td>The XML optimisation model should support tailoring, adaptation, and interchange of heuristic rules.</td>
</tr>
<tr>
<td>GR5 Support for distribution and heterogeneity</td>
<td>The XML optimisation model should meet the distributed and diverse nature of mobile environments.</td>
</tr>
<tr>
<td>GR6 Awareness</td>
<td>The XML optimisation model should provide a mechanism for notification of events to the involved actors.</td>
</tr>
<tr>
<td>GR7 Asynchronous communication</td>
<td>The XML optimisation model should allow disconnections without loss of information or work.</td>
</tr>
<tr>
<td>GR8 Information filtering and aggregation</td>
<td>The XML optimisation model should be able to filter and aggregate information to decrease the bandwidth needed.</td>
</tr>
</tbody>
</table>

Table 7.1: General requirements imposed on a model supporting optimisation of XML documents in a wireless network.

7.2.1 XML properties

The facts that XML is a verbose language, and that XML documents tend to contain redundant information, can introduce large documents with smaller
amounts of valuable information. These properties introduce requirements such as the need to support several parsers, adapting information, and compression.

GR 1. Supporting several parsers: The amount of memory and time needed to parse an XML document depends heavily on the kind of parser selected. Supporting more than one parser and being able to determine which is more appropriate for the task at hand is important in a model optimising the processing time for XML documents.

GR 2. Information adaptation: Particularly large XML documents may be impossible to parse on a mobile device. A means to adapt these documents so that they can be read by a mobile client is important. Such adaptation might include splitting the document into smaller parts or extracting the documents structure.

GR 3. Compression: Lossless compression of XML documents, while still keeping the advantage of XML being based on ASCII characters, is important to better utilize the available network.

7.2.2 Behaviour and support

The nature of mobile environments impose a number of requirements. These include those related to behaviour and support. Behaviour is associated with execution within the model, while support addresses the distribution and heterogeneity of mobile and wireless environments.

GR 4. Tailorable set of heuristic rules: Mobile environments are constantly evolving. They also tend to be rather unpredictable concerning available resources. Considering these two issues, the need for a set of tailorable, adaptable and interchangeable heuristic rules becomes evident.

GR 5. Support for distribution and heterogeneity: Mobile environments are diverse in nature, and these environments are distributed and heterogeneous in nature. Therefore, support for these issues is an important element.

7.2.3 Services provided

A system should provide some general services to extend the range of XML messages which can be utilized in a mobile environment. These include awareness, filtering and adaptation of information, and compression.
CHAPTER 7. A MODEL ENABLING XML OPTIMISATION

GR 6. Awareness: Optimisation of the process of transferring and transforming XML documents calls for a mechanism which allows involved actors to be aware of events which may affect their actions and help them avoid conflicts.

GR 7. Asynchronous communication: Communication within the model will be intermittent. Allowing users to connect and disconnect without facing undue restrictions is desirable. Asynchronous communication introduces flexibility into the model.

GR 8. Information filtering and aggregation: Systems might have a number of sources which produce a large amount of similar XML documents. Filtering and aggregation of information can decrease the bandwidth used.

7.3 Overall description of the model

A model, enabling transformation of XML documents to optimise processing and transfer-time within a mobile setting, can be suggested based on the motivating scenario and the general requirements outlined in the previous sections. My suggestion consists of a system which can be divided into two main pieces; a proxy server and a mobile client. Figure 7.1 depicts an overall view of the suggested system.

The proxy server resides between the communicating parties. Its objective is to allow transformation of XML messages based on a set of heuristic rules. Hence, the server has to accept XML messages, process them as defined by the heuristic rules and be able to forward them within reasonable time.

The second part of the system, residing on the mobile client, consists of a parser able to receive, interpret and send transformed XML messages. The parser is placed between the communication layer and the application layer. As elaborated in Section 4.1, the mobile device has a limited amount of resources, hence the system needs to take this into consideration.

As depicted in Figure 7.1 a third part exists; the content provider or sender of the XML message. The intention is that this part will not be affected by the system. The sender needs only to be aware of the system so that messages are not sent directly to the mobile device but are routed through the proxy-server. This can be achieved by sending information from
CHAPTER 7. A MODEL ENABLING XML OPTIMISATION

7.3.1 Optimising an XML document

The optimisation framework will concentrate on minimizing the size of XML documents. In other words, different schemes for XML size reduction may be implemented on the proxy server. Initially four different methods are suggested;

**Redundancy reduction.** The size caused by redundancy in XML documents can be reduced. A simple scheme for achieving this would be to reduce tag and attribute names to single characters.

**Data and structure separation.** The structure of an XML document may be separated from the data. The structure could be sent to the client to provide an overview, and to allow the client to require only parts of the data.

**Synthesis and aggregation.** In a system where multiple sources feed a single recipient with similar information, a scheme aggregating the information might be useful. One example would be to provide an average of some value collected from several XML messages.
Segmentation. A very large XML document might be impossible to parse on a device with limited resources. Splitting the document into smaller parts would allow the user to parse smaller parts of the whole document.

All these schemes may be used either independently or in conjunction. If more than one scheme may be used, the client needs to be aware on which one is used in each case. It also needs to be able to handle different kinds of transformed documents.

7.3.2 Defining heuristic rules

The proxy server has to decide when an XML document can be transformed to optimise the transfer and processing time. Developing reasonable heuristics is important. However, this work is challenging and requires considerable effort. First, significant factors driving the decision made by the rules need to be identified, second, these factors need to be incorporated into the rules. Defining a full set of heuristic rules is beyond the scope of this project and is left for further research. However, the following provides some examples of significant factors and heuristic rules:

Significant factors

- *Available bandwidth*. The available bandwidth determines how fast an XML document may be transfered.

- *Available memory on receiving client*. The memory on the client sets how large the document to be parsed may be.

- *Available processing power on receiving client*. The available processing power on the client determines how fast an XML document may be parsed.

- *Parser used by client*. SAX parsers use less memory and are generally faster than DOM parser.

- *The size of the XML document*. The size of the XML document is a very important factor.

- *Parser setup time*. The parser setup time is a constant time which is required whenever a parser is initialised.
CHAPTER 7. A MODEL ENABLING XML OPTIMISATION

Heuristic rules

Note that the following heuristic rules are based on educated guesses.

- Whenever a document, passed from the content provider, is smaller than 1 KB it should be forwarded directly to the client by the proxy server without any processing.

- Do not send XML documents to the client which have a size larger than 30% of the clients available memory.
Chapter 8

Requirements

Chapter 7 describes a generic model allowing optimisation of XML in a mobile environment. This chapter takes a closer look at some of the functionality a system implementing the model needs to cover, this is done in Section 8.1. Section 8.2 lists the requirements elicited from the work done in Section 8.1.

8.1 Overall system functionality

This section describes some of the functionality the system has to support. Use cases are used to aid this description. The use cases are represented through use-case diagrams and elaborating textual use-cases. The notation used in the diagrams is based on the Unified Modelling Language (UML) as described by Fowler [21], while the notation used for the textual use-cases is described in Appendix F.

Five use-case diagrams are used to describe the main functionality required by the system. The use-cases describe the following functionalities:

- Initialisation and set-up of the system.
- Optimising an XML document.
- Sending an XML document to the client.
- Receiving an XML document from the client.

The following actors, active within the system, have been identified:

- Proxy-server
CHAPTER 8. REQUIREMENTS

- User
- Mobile client
- Content provider

Notice that not all messages sent within the system are required to pass through the proxy-server, only XML documents are required to do so. In other words, the mobile client can communicate directly with the content provider.

Use-case for the initialisation and set-up of the system

Figure 8.1 depicts the functionality needed to establish a connection between the participating actors. The intention of this diagram is to show that all actors need to establish a connection before any further action can be taken. A textual use-case elaborating the diagram is given in table 8.1.

Figure 8.1: A use-case diagram for the initialisation of the system
## CHAPTER 8. REQUIREMENTS

<table>
<thead>
<tr>
<th>Use-case name</th>
<th>Initialisation of the system</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iteration</td>
<td>Filled</td>
</tr>
<tr>
<td>Abstract</td>
<td>The system is set up through a registration process.</td>
</tr>
<tr>
<td>Trigger</td>
<td>Started by user.</td>
</tr>
</tbody>
</table>

### Main path

1: The user starts his mobile client.
2: The user establishes a connection with a content provider.
3: The user informs the content provider about the proxy server.
4: The content provider establishes a connection with the proxy server.
5: The proxy-server establishes a connection with the mobile client.

### Alternate path

**Register new user**

1^a: If the application on the mobile client is started for the first time, the mobile client has to register itself at the proxy-server.
1^b: The proxy-server stores information about the new user (mobile device) in a database.
1^c: The proxy-server informs the user about the registration.
1^d: Jump to point 2 in the main path.

### Exception path

2^a: If the connection to the content provider cannot be established, the user is informed and given a choice whether to cancel or to try again.
2^b: If cancel is selected the application on the mobile client quits. If try again is selected the application tries to establish a connection and if it succeeds; jump to 3.
4^a: If the content provider is unable to establish a connection, the user is informed and given a choice whether to cancel or to try again. If cancel is selected the application on the mobile client quits. If try again is selected the content provider tries to establish a connection and if it succeeds; jump to 5.

### Preconditions

The user has installed an application on the mobile client supporting the system.

### Author

Tobias Melcher

### Date

01.11.04
Use-case for optimising an XML document

Figure 8.2 depicts the functionality needed to optimise an XML document. The diagram shows which parts are needed to enable an optimisation of XML documents. A textual use-case elaborating the diagram is given in table 8.2.

Figure 8.2: A use-case diagram for the XML optimisation
## Use-case: XML optimisation

<table>
<thead>
<tr>
<th>Use-case name</th>
<th>Optimising an XML document</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iteration</td>
<td>Filled</td>
</tr>
<tr>
<td>Abstract</td>
<td>An XML document is optimised according to a set of rules.</td>
</tr>
<tr>
<td>Trigger</td>
<td>The proxy server receives an XML document from a content provider.</td>
</tr>
</tbody>
</table>
| Main path     | 1: The XML document is parsed by an XML parser.  
|               | 2: The structure of the document is analysed and compared to values stored in the rule-base.  
|               | 3: The document is transformed according to the heuristic rules.  
|               | 4: The document is forwarded to the mobile client. |
| Alternate path| 2*: If the documents structure implies no transformation, point 3 is skipped. |
| Exception path| -                           |
| Preconditions | a) The communication path between the content provider, proxy-server and the mobile client has been initialised.  
|               | b) The XML document is large enough to be considered worthwhile transforming. |
| Author        | Tobias Melcher              |
| Date          | 01.11.04                    |

Table 8.2: Use-case: XML optimisation
## CHAPTER 8. REQUIREMENTS

<table>
<thead>
<tr>
<th>Use-case name</th>
<th>Parsing a transformed XML document</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iteration</td>
<td>Filled</td>
</tr>
<tr>
<td>Abstract</td>
<td>A transformed XML document is parsed.</td>
</tr>
<tr>
<td>Trigger</td>
<td>The mobile client receives an XML document from the proxy server, or the other way around.</td>
</tr>
</tbody>
</table>
| Main path             | 1: It is determined whether the XML document has been transformed or not.  
                        | 2: The XML document is parsed by an XML parser.  
                        | 3: The information from the parsed XML document is utilized by some application. |
| Alternate path        | 3\textsuperscript{a}: If the document is received from the mobile client, the proxy server changes it back to its original composition if required.  
                        | 3\textsuperscript{b}: The document is forwarded to the recipient. |
| Exception path        | -                                   |
| Preconditions         | a) The communication path between the content provider, proxy-server and the mobile client has been initialised. |
| Author                | Tobias Melcher                      |
| Date                  | 08.11.04                            |

Table 8.3: Use-case: Parsing a transformed XML document
CHAPTER 8. REQUIREMENTS

Use-case for sending an XML document to the client

Figure 8.3 depicts the functionality needed to send an XML document to the mobile client from a content provider. The diagram shows which parts that are involved in the process and what the responsibilities of these parts are. Textual use-cases elaborating the diagram are given in table 8.4 and 8.5.

![Use-case diagram for the client receiving an XML](image)

Figure 8.3: A use-case diagram for the client receiving an XML
### CHAPTER 8. REQUIREMENTS

<table>
<thead>
<tr>
<th>Use-case name</th>
<th>Sending an XML document to the client</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iteration</td>
<td>Filled</td>
</tr>
<tr>
<td>Abstract</td>
<td>An XML document is sent from the content provider through the proxy-server to the mobile client.</td>
</tr>
<tr>
<td>Trigger</td>
<td>The content provider receives a request for an XML document from the client.</td>
</tr>
</tbody>
</table>
| Main path     | 1: The user chooses to download an XML document from the content provider.  
2: The content provider includes information about the destination and sends the XML document to the proxy-server.  
3: The proxy-server receives the document from the content provider and handles it as given in the use-case shown in 8.2.  
4: The proxy-server forwards the document to the destination provided by the content provider.  
5: The mobile client receives the XML document from the proxy-server.  
6: The document is parsed by the mobile client.  
7: A message confirming that the XML document has been received and information about how much time was needed for the process is sent to the proxy-server.  
8: Information, concerning the process of sending the XML document to the mobile client, is stored in a database. |
| Alternate path| -                                    |
| Exception path| -                                    |
| Preconditions | The communication path between the content provider, proxy-server and the mobile client has been initialised. |
| Author        | Tobias Melcher                       |
| Date          | 01.11.04                             |

Table 8.4: Use-case: Sending an XML to the client
### Use-case: Suspend communication

<table>
<thead>
<tr>
<th>Use-case name</th>
<th>Suspend communication</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iteration</td>
<td>Filled</td>
</tr>
<tr>
<td>Abstract</td>
<td>The connection is broken for some reason during the transfer of an XML document. Both the mobile client and the proxy-server save their current state to be able to resume the transfer at a later stage.</td>
</tr>
<tr>
<td>Trigger</td>
<td>The connection between the proxy-server and the mobile client is broken.</td>
</tr>
</tbody>
</table>
| Main path     | 1: The proxy server stops transmitting the XML file.  
2: The mobile client saves the received part of the file, and notes how much of the file is missing.  
3: The proxy-server notes the point in the XML document when communication was lost and saves the document to a database.  
4: After a couple of days the database entry is removed, and information about the transfer is saved in a database. |
| Alternate path| 4a: The mobile client resumes the connection with the proxy-server and is now able to download the rest of the XML document. |
| Exception path| -                     |
| Preconditions | a) The communication path between the content provider, proxy-server and the mobile client has been initialised.  
b) The proxy-server is sending an XML document to the mobile client. |
| Author        | Tobias Melcher        |
| Date          | 01.11.04              |

Table 8.5: Use-case: Suspend communication
CHAPTER 8. REQUIREMENTS

Use-case for receiving an XML document from the client

Figure 8.3 depicts the functionality needed to send an XML document from the mobile client to a content provider. The diagram shows which parts are involved in the process and what the responsibilities of these parts are. Textual use-cases elaborating the diagram are given in table 8.4 and 8.7.

Figure 8.4: A use-case diagram for the client sending an XML
CHAPTER 8. REQUIREMENTS

<table>
<thead>
<tr>
<th>Use-case name</th>
<th>Receiving an XML document from the client</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iteration</td>
<td>Filled</td>
</tr>
<tr>
<td>Abstract</td>
<td>An XML document is sent from the mobile client through the proxy-server to a recipient.</td>
</tr>
<tr>
<td>Trigger</td>
<td>The user wishes to return an XML document to the content provider.</td>
</tr>
<tr>
<td>Main path</td>
<td>1: The user chooses send an XML document.</td>
</tr>
<tr>
<td></td>
<td>2: The mobile client includes information about the destination and sends the document to the proxy-server.</td>
</tr>
<tr>
<td></td>
<td>3: The proxy-server receives the document from the mobile client and transforms it into a valid XML document.</td>
</tr>
<tr>
<td></td>
<td>4: The proxy-server forwards the document to the destination provided by the mobile client.</td>
</tr>
<tr>
<td></td>
<td>5: The content provider receives the XML document from the proxy-server.</td>
</tr>
<tr>
<td>Alternate path</td>
<td>3°: If the XML is valid upon arrival at the proxy-server, a transformation is unnecessary; jump to 4.</td>
</tr>
<tr>
<td>Exception path</td>
<td>-</td>
</tr>
<tr>
<td>Preconditions</td>
<td>The communication path between the content provider, proxy-server and the mobile client has been initialised.</td>
</tr>
<tr>
<td>Author</td>
<td>Tobias Melcher</td>
</tr>
<tr>
<td>Date</td>
<td>01.11.04</td>
</tr>
</tbody>
</table>

Table 8.6: Use-case: Receiving an XML from the client
### Use-case name

<table>
<thead>
<tr>
<th>Use-case name</th>
<th>Suspend communication</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iteration</td>
<td>Filled</td>
</tr>
<tr>
<td>Abstract</td>
<td>The connection is broken for some reason during the transfer of an XML document. Both the mobile client and the proxy-server save their current state to be able to resume the transfer at a later stage.</td>
</tr>
<tr>
<td>Trigger</td>
<td>The connection between the proxy-server and the mobile client is broken.</td>
</tr>
</tbody>
</table>
| Main path     | 1: The mobile client stops transmitting the XML file.  
|               | 2: The proxy-server saves the received part of the file, and notes how much of the file is missing.  
|               | 3: The mobile client notes the point in the XML document when communication was lost and stores the document in some local memory.  
|               | 4: The entry is removed after a given time. |
| Alternate path| 4a: The mobile client resumes the connection with the proxy-server and is now able to upload the rest of the XML document. |
| Exception path| -                     |
| Preconditions | a) The communication path between the content provider, proxy-server and the mobile client has been initialised.  
|               | b) The mobile-client is sending an XML document to the proxy-server. |
| Author        | Tobias Melcher        |
| Date          | 01.11.04              |

Table 8.7: Use-case: Suspend communication
CHAPTER 8. REQUIREMENTS

8.2 System requirements

This section lists a number of requirements to be addressed by the prototype system. The first two subsections describe the functional requirements, while the last describes the non-functional requirements.

8.2.1 Functional requirements for the proxy-server

The proxy server needs to address several requirements to implement the desired functionality. These requirements are listed in the following requirements, PROXY-FR1 through PROXY-FR8

PROXY-FR1 Establishing connections. The system has to be able to set up connections between the mobile client, the proxy-server and the content provider to enable communication between the involved parties. The proxy server has to be able to handle connection requests both from the content provider and the mobile client. The use-case depicting this functionality is given in Figure 8.1.

PROXY-FR2 Registering a user. A mechanism for registering information about a mobile client has to be supported by the proxy-server. Registering a client profile will support the adaptation of transferred XML documents. The use-case depicting this functionality is given in figure 8.1.

PROXY-FR3 Receiving and forwarding documents. The proxy server has to be able to receive XML documents, process them and forward them to the correct party. This is needed to allow the proxy server to deliver a service to a client. The use-cases depicting this functionality are given in Figure 8.3 and 8.4.

PROXY-FR4 Suspending communication. The proxy server has to be able to suspend a communication session. The wireless environment is exposed to frequent disconnections due to the environment, user movement and scarce resources in mobile devices. Asynchronous communication is a means to overcome these challenges, being able to suspend and resume communication is an important side effect allowed by this communication scheme. The use-cases depicting this functionality are given in Figure 8.3 and 8.4.
CHAPTER 8. REQUIREMENTS

PROXY-FR5 Resuming communication. The proxy server has to be able to resume a communication session. The use-cases depicting this functionality are given in Figure 8.3 and 8.4.

PROXY-FR6 Parsing an XML document. Parsing of XML documents needs to be supported by the proxy server. Being able to extract information from an XML document is absolutely necessary to allow transformation and adaptation of XML documents. The use-case depicting this functionality is given in Figure 8.2.

PROXY-FR7 Transforming an XML document. The proxy server needs to be able to change the XML document. The intention of the complete system is to optimise processing and transferring of XML documents in a mobile environment. The core part of this is transforming an XML document to a more appropriate and less resource consuming XML document. The use-case depicting this functionality is given in Figure 8.2.

PROXY-FR8 Storing a set of heuristic rules. The proxy server needs to be able to store a set of heuristic rules in an appropriate manner. This is necessary to allow the proxy server to determine when and how to transform an XML document based on a set of given rules. The use-case depicting this functionality is given in Figure 8.2.

8.2.2 Functional requirements for the mobile client

The mobile client needs to address several requirements to implement the desired functionality of a prototype system. These requirements are listed in the following requirements, CLIEN-FR1 through CLIEN-FR3.

CLIENT-FR1 Establishing a connection. The mobile client has to be able to connect to a content provider and to the proxy server. The use-case depicting this functionality is given in figure 8.1.

CLIENT-FR2 Receiving and sending an XML document. The mobile client has to be able to receive an XML document and send an XML document at will. In a system where information is fetched from distributed entities, it is obvious that some mechanism for sending and receiving documents is required.
CHAPTER 8. REQUIREMENTS

The use-cases depicting this functionality are given in figure 8.3 and 8.4.

CLIENT-FR3 Parsing transformed XML documents. When a document is received from the proxy server it might be in a transformed form. The mobile client needs to be able to parse and understand this transformed document just as well as a document which has not been changed. The use-case depicting this functionality is given in figure 8.2.

8.2.3 Non-functional requirements

A number of non-functional requirements are defined in the following subsections.

External interface requirements

This section addresses issues concerned with the different kind of interfaces the prototype system has to deal with.

INFR1 Standard menus, buttons and text fields are to be used within the application for the mobile client.

INFR2 The application developed for the mobile client should be easy to use and understand. 95% of all users should be able to learn and use the application within 5 minutes.

INFR3 The application developed for the mobile client shall support the Java 2 Micro Edition (J2ME).

INFR4 The proxy server is to be developed using Java 2 Standard Edition (J2SE).

INFR5 Sockets are to be used to enable communication between parties.

Performance requirements

This section would normally address performance issues such as the number of simultaneous users and the time needed to perform a given task. These kind of requirements are left out since the system will only be a prototype system.

PNFR1 The system shall obtain a performance gain of more than 10% compared to normal XML document passing and processing.
CHAPTER 8. REQUIREMENTS

Design constraints
This section summarizes the requirements imposed by constraints set as a result of the prescribed use of certain hardware or standards.

DNFR1 The maximum size of the mobile application should be no more than 60kB.

Software system attributes
This section describes requirements related to system quality.

SNFR1 None of the documents sent through the proxy server should disappear.

SNFR2 All classes, methods and variables shall have an explanation for future studies of the code.

SNFR3 It has to be possible to add, remove and change heuristic rules, parsers, and transforming algorithms

8.3 Summarizing the requirements
This section summarizes the requirements by providing a table mapping the functional requirements to their respective use-case, and by providing two tables which prioritise the functional and non-functional requirements.

8.3.1 Mapping the use-cases to the requirements
Table 8.8 maps the functional requirements to the use-case diagrams. This mapping is useful for future system testing as this aids the tester in finding which functionality has been covered by the system.

8.3.2 Prioritising the requirements
Table 8.9 and 8.10 summarize all requirements to the system. A short description is given together with the identifying number. The requirements are prioritised using the values; high (H), medium (M), and low (L). Not all requirements will be implemented in a first development of a prototype system and these requirements are postponed.
### CHAPTER 8. REQUIREMENTS

<table>
<thead>
<tr>
<th>ID</th>
<th>Initialisation</th>
<th>Optimisation</th>
<th>Client receive</th>
<th>Client send</th>
<th>Description</th>
<th>Priority</th>
<th>Postponed</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROXY-FR1</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>Establish connections</td>
<td>H</td>
<td></td>
</tr>
<tr>
<td>PROXY-FR2</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>Registering a user</td>
<td>L</td>
<td>X</td>
</tr>
<tr>
<td>PROXY-FR3</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>Receiving and forwarding messages</td>
<td>H</td>
<td></td>
</tr>
<tr>
<td>PROXY-FR4</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>Suspending communication</td>
<td>M</td>
<td>X</td>
</tr>
<tr>
<td>PROXY-FR5</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>Resuming communication</td>
<td>M</td>
<td>X</td>
</tr>
<tr>
<td>PROXY-FR6</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>Parsing an XML document</td>
<td>H</td>
<td></td>
</tr>
<tr>
<td>PROXY-FR7</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>Transforming an XML document</td>
<td>H</td>
<td></td>
</tr>
<tr>
<td>PROXY-FR8</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>Storing a set of heuristic rules</td>
<td>H</td>
<td></td>
</tr>
<tr>
<td>CLIENT-FR1</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>Establish connections</td>
<td>H</td>
<td></td>
</tr>
<tr>
<td>CLIENT-FR2</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>Receiving and sending an XML document</td>
<td>H</td>
<td></td>
</tr>
<tr>
<td>CLIENT-FR3</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>Parsing transformed XML documents</td>
<td>H</td>
<td></td>
</tr>
</tbody>
</table>

Table 8.8: Mapping use-cases to functional requirements

<table>
<thead>
<tr>
<th>ID</th>
<th>Description</th>
<th>Priority</th>
<th>Postponed</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROXY-FR1</td>
<td>Establish connections</td>
<td>H</td>
<td></td>
</tr>
<tr>
<td>PROXY-FR2</td>
<td>Registering a user</td>
<td>L</td>
<td>X</td>
</tr>
<tr>
<td>PROXY-FR3</td>
<td>Receiving and forwarding messages</td>
<td>H</td>
<td></td>
</tr>
<tr>
<td>PROXY-FR4</td>
<td>Suspending communication</td>
<td>M</td>
<td>X</td>
</tr>
<tr>
<td>PROXY-FR5</td>
<td>Resuming communication</td>
<td>M</td>
<td>X</td>
</tr>
<tr>
<td>PROXY-FR6</td>
<td>Parsing an XML document</td>
<td>H</td>
<td></td>
</tr>
<tr>
<td>PROXY-FR7</td>
<td>Transforming an XML document</td>
<td>H</td>
<td></td>
</tr>
<tr>
<td>PROXY-FR8</td>
<td>Storing a set of heuristic rules</td>
<td>H</td>
<td></td>
</tr>
<tr>
<td>CLIENT-FR1</td>
<td>Establish connections</td>
<td>H</td>
<td></td>
</tr>
<tr>
<td>CLIENT-FR2</td>
<td>Receiving and sending an XML document</td>
<td>H</td>
<td></td>
</tr>
<tr>
<td>CLIENT-FR3</td>
<td>Parsing transformed XML documents</td>
<td>H</td>
<td></td>
</tr>
</tbody>
</table>

Table 8.9: Prioritised functional requirements
## CHAPTER 8. REQUIREMENTS

<table>
<thead>
<tr>
<th>ID</th>
<th>Description</th>
<th>Priority</th>
<th>Postponed</th>
</tr>
</thead>
<tbody>
<tr>
<td>INFR1</td>
<td>Standard GUI</td>
<td>H</td>
<td></td>
</tr>
<tr>
<td>INFR2</td>
<td>Time to learn</td>
<td>L</td>
<td></td>
</tr>
<tr>
<td>INFR3</td>
<td>J2ME for mobile client</td>
<td>H</td>
<td></td>
</tr>
<tr>
<td>INFR4</td>
<td>J2SE for proxy server</td>
<td>H</td>
<td></td>
</tr>
<tr>
<td>INFR5</td>
<td>Sockets for communication</td>
<td>M</td>
<td></td>
</tr>
<tr>
<td>PNFR1</td>
<td>10% gain in performance</td>
<td>H</td>
<td></td>
</tr>
<tr>
<td>DNFR1</td>
<td>Mobile client; max 60Kb</td>
<td>M</td>
<td></td>
</tr>
<tr>
<td>SNFR1</td>
<td>Do not loose XML documents</td>
<td>M</td>
<td></td>
</tr>
<tr>
<td>SNFR2</td>
<td>Comment code</td>
<td>M</td>
<td></td>
</tr>
<tr>
<td>SNFR3</td>
<td>Adaptation of heuristic rules</td>
<td>H</td>
<td></td>
</tr>
</tbody>
</table>

Table 8.10: Prioritised non-functional requirements
Chapter 9

The prototype system

This chapter describes a prototype system of a system for XML optimisation. The intention of this prototype is to show how the model described in Chapter 7 can be implemented. This chapter also gives an answer to RQ3 defined in Chapter 2.

9.1 Architecture and design

This section describes the architecture and design of the prototype system. The architecture is described in an object-oriented way, using parts of the methodology described by Kruchten in [33], and UML for diagram representation. The architecture and design are important tools for further validation of the system.

9.1.1 Architectural goals and considerations

The intention of the prototype system is to show how the model proposed in Chapter 7 can be implemented.

System quality attributes

Identifying some of the qualities a finished system should have, will help me identify important parts which need to be addressed in the prototype system.

Performance Supporting a number of clients simultaneously is of utmost importance in a finished system. Introducing concurrency in the prototype system will allow for some testing with more than one user.

Modifiable The system is to be incorporated in a constantly evolving environment, and allowing the system to evolve is crucial. Enabling easy
incorporation of new modules to support new heuristic rules is an important part of this. Using known design patterns within the prototype architecture might help future developers to implement new parts to extend the system.

Architectural tactics

Section 9.1.1 identified two system quality attributes which need to be given special attention. Bass et. al. [5] provide a means to achieve these attributes. They list a number of design decisions, which they call tactics, to aid an architect in creating a design. The tactics suited for the prototype are listed in the following.

Performance tactics:

PT 1. **Bound queue size.** Only a certain maximum of event arrivals is allowed. Using this tactic helps limiting the resources needed to process requests. In other words, this means that only a certain number of clients may have their XML file processed at the same time.

PT 2. **Introduce concurrency.** By introducing concurrency several clients may be served at the same time.

Modifiability tactics:

MT 1. **Maintain semantic coherence.** “Semantic coherence refers to the relationships among responsibilities in a module.” Maintaining semantic coherence is achieved by gathering the responsibilities with semantic coherence in one module so that reliance to other modules is minimized. This can be done by abstract common services and provide specialized modules. The prototype system will include the following specialized modules; the parser and the transforming module.

MT 2. **Anticipate expected changes.** The goal of this tactic is to limit the set of modules which need to be modified when a change to the system is needed. The parts within the prototype system, which may change include; the set of heuristic rules, the parser, and the algorithms responsible for transforming the XML document.

MT 3. **Use an intermediary.** This tactic is concerned with minimizing the effect of a change by inserting an intermediary responsible for the activities associated with the dependencies between two modules.
CHAPTER 9. THE PROTOTYPE SYSTEM

Architectural considerations

A number of considerations must be taken into account when the architecture is to be designed. The most important ones are listed in the following:

**Keep it simple.** The limited resources of this project does not allow any extra features within the system. There is no time to write extra code.

**Make it cheap.** The financial resources are limited and using free software or software on student licences is important.

**Use mature open-source libraries.** Using open-source libraries allows adaptation if needed. Maturity is important to avoid unnecessary bugs, and to allow easier access to help files and similar.

**Use design patterns.** Design patterns should be used to increase the value of the prototype by introducing reusable parts. In addition, the experience captured within design patterns would ease the design and implementation phase.

9.1.2 Overall architecture

The prototype will consist of the same elements as outlined in the model presented in Chapter 7. To summarize; a mobile client, a proxy server, and a service provider. The main focus in the development of an architecture will be on the proxy server as this contains the core functionality required to build a system of the type defined in the main research question (see Chapter 2).

Figure 9.1 depicts a layered and distributed view of the systems architecture. The figure gives an overview of the core parts of the system. Among other, it depicts how different parts interact.

The system can be further decomposed into a number of modules responsible of delivering different services. The proxy can be divided into the following modules:

- a communication module enabling communication with content providers and the mobile device,
- a control module providing communication between the different parts of the proxies architecture,
- a parsing module parsing the received messages if this is needed,
• a transformation module transforming the XML messages, and
• a rule-base containing the implemented heuristic rules.

The different modules have different responsibilities. The communication module shall provide a framework for communication with the mobile client and the content provider. It has to listen for incoming data, while being able to send data. The module has to be able to distinguish between the different entities connected at any time. The control module’s responsibility is to control the flow of data within the proxy server. It has to instantiate the needed objects and provide for the required message passing between the objects and different modules. The parsing module is responsible for parsing the document. In the prototype system, the parser will be a model parser - e.g. DOM - which builds a model in memory. This allows the transforming module to operate directly on the model. The transforming module is responsible for making changes to the XML document - the model passed on from the parser - and return an optimised XML document. The rule-base module implements the rules which are to be used to make decisions such as whether to transform or not.

Figure 9.2 depicts how the pipe-and-filter architectural style is used in the proxy server. The XML document is passed through the parser and the transformator as a data stream. The architecture differs from the original architectural style as the data stream will not be continuous. The transfor-
CHAPTER 9. THE PROTOTYPE SYSTEM

The communication module will have to wait for the parser to finish parsing the XML document. Still, both modules act as filters as they produce different representations of the original XML document.

![Figure 9.2: XML document filtering](image)

The mobile client consists of the following modules;

- a communication module enabling communication with content providers and the mobile device,
- a parsing module parsing the received messages if this is needed and
- a user interface giving feedback to the user and receiving input from the user.

As with the proxy server, the different modules have different responsibilities. The communication module shall provide a framework for communication with the proxy server and the content provider. The module is responsible for transmitting and receiving. The parsing module is responsible for parsing the data received, and handing it over to the module responsible for the user interface. This module is responsible for presenting data to the user and allow the user to communicate with the application.

The content provider will also be implemented in the prototype system. This part does not however deliver any important functionality, and is merely developed to show how messages may be passed within the system.

9.1.3 Logical view

The proxy server can, as shown, be divided into five different modules. These modules consist of several classes in most cases. The logical view depicting the top level class diagram consisting of the five categories of classes - e.g. the
modules - is shown in Figure 9.3. The functional requirements are primarily supported by the logical architecture.

![Class Diagram](image)

Figure 9.3: The top level class diagram (notation is from [33])

The class categories may be decomposed. Figure 9.4 depicts a decomposition of the Parser category. The design uses the Factory Method pattern as described by Gamma et al.[23]. The pattern defines an interface for creating a parser object which is used by the ProxyController class to parse the XML document.

![Parser Diagram](image)

Figure 9.4: The parser

The Transformer class category is designed using the Abstract Factory pattern [23]. This pattern is used to allow the ProxyController class to be independent of how the transformer objects are created. This also allows the system to use different transformer objects together if needed. The design is
CHAPTER 9. THE PROTOTYPE SYSTEM

depicted in Figure 9.5.

Designing a good solution for the implementation of heuristic rules is important as these probably will be an element of rapid changes. To overcome this, the Strategy pattern [23] is used. This pattern allows to define a family of algorithms (the heuristic rules) which may be interchanged. The design of the implementation of the heuristic rules is depicted in Figure 9.6.
CHAPTER 9. THE PROTOTYPE SYSTEM

9.1.4 Process view

The process view depicts how operations are distributed, and which run simultaneously. In other words, it addresses the issues of concurrency and distribution within the system. The process view is primarily concerned with the non-functional requirements. Figure 9.7 depicts the process view for the prototype system.

![Diagram showing the process view](image)

Figure 9.7: Process view (notation is from [33])

The client has three processes; the Communicator process is responsible for the communication with the proxy server and the provider, the UI process is responsible for giving feedback to the user and listening for user input, and the Parser process parses incoming messages. The proxy server has two processes; the Terminal process is responsible for communication and for starting new Optimisation processes, and the Optimisation process is concerned with optimising the XML data.
CHAPTER 9. THE PROTOTYPE SYSTEM

9.2 Implementation

Implementing the prototype system is needed to validate the proposed model of Chapter 7. An implementation will allow future analysis of the architecture, which may lead to improvements. Unfortunately, only parts of the prototype system have been implemented. This is due to the limited resources and time constraints of this project. Implementing the full prototype and improving the architecture has to be done in future work.

This section presents the parts of the prototype which have been implemented and discusses the difficulties met.

9.2.1 Communication framework

Allowing the different actors within the network to communicate requires a framework for communication. Java provides an interface to the network through the java.net package. Sockets where selected as the means to communicate, and a set of messages allowing information exchange where set up. Table 9.1 and 9.2 depict the messages.

```xml
<?xml version="1.0"?>
<response>
  <header>
    <sender>[IP-address]</sender>
    <receiver>[IP-address]</receiver>
    <date>[date]</date>
    <type>[XMLFileList | XMLFile]</type>
  </header>
  <body>
    [XML file | XML filelist]
  </body>
</response>
```

Table 9.1: The response message is used by the content provider and the proxy server to answer client requests.
CHAPTER 9. THE PROTOTYPE SYSTEM

Table 9.2: The request message is used by the client to require data from the content provider.

The selection of sockets and a self-defined set of messages proved to be a poor choice, as socket programming is hard to handle and as the self-defined messages required parsing and interpretation. To solve these problems a standard communication protocol such as HTTP should be selected. Java supports HTTP-connections and handles messaging quite well. However, building a self-defined protocol might be effective if some work is put into it.

9.2.2 The mobile client

The mobile client is implemented using J2ME (see Appendix E.1) and the Mobile Information Device Profile (MIDP) 2.0. The parser used within the client is the kXML 1.2 parser. The current functionality of the client is:

Connection to a content provider. Based on input from the user, the client is able to connect to any content provider.

Requesting data from the content provider. As soon as the client is connected, it requests a list of XML files. This list is displayed to the user who can select a file to be downloaded. The client sends this request to the content provider together with information about the proxy server.
CHAPTER 9. THE PROTOTYPE SYSTEM

Displaying information to the user. The client is able to display information to the user, such as the downloaded list of XML file.

The current functionality of the client is very basic, and only sets up the initial parts of the system. No functionality supporting parsing and transformation of transformed XML documents has been implemented. This part of implementation is left until the proxy server is able to transform and optimise XML documents.

9.2.3 The proxy server

The proxy server is implemented using J2SE. At this moment, its only functionality is accepting connections, the rest of the proxy server is just an empty shell. However, an earlier version of the proxy server revealed some difficulties;

Message handling. Using a self-defined message framework seems to cause some problems as it is based on XML. The solution implemented in the early version required the proxy server to save the XML file to disk to allow further processing. This requires several io-accesses which might reduce performance.

Flow of events. Controlling the flow of events within the system is an important part. It also seems to be a demanding task which requires a large amount of consideration and testing.

Representing the XML file. How the XML file should be represented at the proxy server is a question which can only be answered through testing. Should it be a stream of data or should it be a file?

Implementing heuristic rules. Representing heuristic rules in a way that makes them interchangeable and at the same time offers enough information to the ProxyController is hard. A reasonable set of primitives providing the necessary information needs to be developed.

9.2.4 The content provider

The content provider is implemented using J2SE and the Xerces SAX parser. It is able to receive requests from the client and handle them appropriately. In other words, it accepts connections from the client and is able to connect to the proxy server.
9.2.5 Remaining work

Implementing a first working prototype requires some more work. The content provider provides the basic functionality and can be considered finished. Much work remains, however, on the proxy server. None of the functionality required to optimise an XML document has been implemented. As elaborated in earlier parts of this report, the proxy server needs to be able to use heuristic rules to optimise an XML document. In other words, it has to decide if the document should be transformed or not, and if, how. The client is not finished either. Implementing the part which is able to translate and understand transformed messages still remains.
Part IV

Evaluation and conclusion
Chapter 10

Evaluation and discussion

This chapter discusses and evaluates the work presented in this report.

10.1 Discussion

This discussion focuses on issues concerning the model suggested in Chapter 7 and the prototype described in Chapter 9. These issues include discussing the use of a central proxy server, performance, the development of heuristic rules and the mobile client.

10.1.1 The use of a central proxy server

The selection of a centralized proxy server to handle XML optimisation is a key issue worth discussion. This solution was selected because it should be fairly easy to integrate into existing systems. However, the chosen framework requires an infrastructure for communication, and thereby disallows the use in Mobile Ad-hoc Networks. It also introduces an extra step in the communication between a content provider and a client. Middleware and agent technology are possible solutions solving these problems. Future work should address these solutions.

As is, the mobile client communicates directly with the content provider, and only XML documents are routed through the server. However, as pointed out by my supervisor, Carl Fredrik Sørensen, it is possibly the simplest solution to actually use the proxy server as a router for all information. The server can thus serve as a connection point especially to the Internet. The proxy server can then also be extended to support a “home” service for the client which can handle eventual client disconnections and still be able to
Chapter 10. Evaluation and Discussion

continue to communicate with the world. This is especially interesting related to services that requires some interpretation, aggregation and a lot of coordinating traffic. This solution can support mobile agents by letting the proxy server receive agents coming home, and then transform such data to a format and form suitable for resource-weak mobile clients.

10.1.2 Performance issues

Another important issue is performance. One of the main objectives of this work is to improve the performance of XML data traffic in mobile environments. The model suggested in this report can be able to better utilize the limited bandwidth of wireless networks. However, there are some concerns. The use of a centralized proxy server introduces an extra step in the communication path which can increase the time needed to send data from a content provider to a client. This is a concern when the load of XML documents, to small to be optimised, is high. Another concern, is that the proxy server can become a bottleneck if it has to handle many large documents simultaneously. One solution would be to extend the number of proxy servers or in some other way the capacity of the proxy server. Another solution could be to limit the load on the proxy server and force the content provider to route the document directly to the client and inform the client about the problem. However, considering the content provider as a bottleneck can be considered a small problem, as this server probably will be able to serve clients fast compared to the performance of the wireless link. Limiting the number of io-accesses on the proxy server is also an important performance issue. The proxy server will probably have to store the XML document from the content provider on disk, and it will also depend on databases to store important information such as user profiles. Users normally expect a minimal response time, optimising the overall system performance is an important issue, and should thus be the subject of further research.

Following this line of arguments, another performance issue worth discussing, is the use of Java as the implementation platform for the proxy server and the mobile client. Currently, the most significant weakness of Java is its moderate performance. However, Java is an object-oriented language and has the capability of being executed on heterogeneous environments. In addition, the adoption of Java on mobile devices and the number of supporting libraries is good, compared to programming languages such as .NET. Considering this, Java was a natural choice to make. Nonetheless, it should be noted that the moderate performance of Java can be improved by making the code executable instead of interpretable. This can be done by using one
CHAPTER 10. EVALUATION AND DISCUSSION

of the several available just in time compilers for Java.

The last performance issue worth discussion is the choice of parser. This project has not been concerned on which parser performs better. However, the choice of parser is significant to the performance of the system and should be given some attention in further research.

10.1.3 The development of heuristic rules

The development of good heuristic rules is crucial to the success of a system like the proposed system. A set of such rules is needed to be able to make decisions on when an XML document needs to be transformed. However, defining such rules has proven to be hard, and hence a considerable amount of effort to develop good heuristic rules is required. A set of significant factors needs to be identified and incorporated in a framework for heuristic rules. Knutsen’s [32] covers this issue partly. An associated problem is how these significant factors should get their necessary data. The requirement PROXY-FR2\(^1\) is one possible mechanism which might solve this problem partially.

10.1.4 The mobile client

The mobile client is of course a central part of the model. In the prototype the parser is closely connected with the application providing the user interface. This is however not a general solution which would be preferable. A better solution would be to allow the parser/translator to work as a proxy on the client, meaning that all incoming XML data is routed through the parser before it is passed on to the applications needing the data.

10.2 Evaluation

In order to evaluate the project it is necessary to consider how the goal of this project was reached through answering the research questions. Further, it is necessary to summarize which requirements where met.

10.2.1 Answering the research questions

Looking at how well the research questions have been answered is a way to see if the goals of this project have been achieved or not.

\(^1\)PROXY-FR2 is concerned with the registration of new users, and hence the collection of client and user information
Starting with the main research question, stated in Section 2.1, this report tried to answer:

*How can we build a system based on mobile clients and supporting servers which optimise processing and transfer time of XML-based information between these?*

This question has been answered with a set of requirements, a suggested model and a prototype architecture. The model suggested in Chapter 7 provides a framework for optimising the processing and transfer time of XML-based information and the prototype described in Chapter 9 describes how a system based on the model can be built.

The main research question lead to the definition of several subquestions which determined the progress of this project:

**RQ1** *Current state:* Are there any efforts which can answer or help answer the main question?

The state-of-the art survey revealed that no research has been done answering the main question directly. Still, there are some efforts related to the problem addressed in this report. These efforts provided ideas for a solution which where incorporated in the model.

**RQ2** *Requirements:* What is the nature of XML messages to mobile devices and what kind of hardware restrictions exist? Then, what requirements do these impose on the optimisation in handling XML messages?

Part II described the nature of XML and the restrictions caused by the nature of mobile environments. It thereby, identified some challenges in the combination of XML and mobile environments. Section 7.2 summarized the requirements needed to be addressed in a model allowing optimisation of XML documents.

**RQ3** *Solution:* What is needed to provide a foundation for meeting the requirements? What kinds of technologies are suitable for an implementation?

Section 7.3 describes a model which is able to meet the general requirements. However, this model has several shortcomings as discussed in 10.1. Chapter 5 describes some of the technologies which might be used in an implementation.
CHAPTER 10. EVALUATION AND DISCUSSION

RQ4 Evaluation: How well does the solution solve the problem given in the main question?

The model proposed in Chapter 7 should be able to solve the problem stated in the main research question, though with some shortcomings as discussed in 10.1. Unfortunately, the implementation of the prototype was not finished and no tests regarding how well the problem was solved could be run. This is left for further research.

10.2.2 Meeting the requirements for the prototype

Table 10.1 shows how the functional requirements given in Section system-requirements are met by the prototype system. The table depicts whether they have been covered in the prototype’s architecture and in the implementation.
### Table 10.1: The table summarizes how the functional requirements have been covered by the prototype system.

<table>
<thead>
<tr>
<th>Description</th>
<th>Priority</th>
<th>Postponed</th>
<th>Architecture Implementation</th>
<th>Implemented</th>
<th>Transformed XML</th>
<th>XML document</th>
<th>Receiving and sending an XML document</th>
<th>Establishing connections</th>
<th>Suspending communication</th>
<th>Resuming communication</th>
</tr>
</thead>
<tbody>
<tr>
<td>Establish connections</td>
<td>H</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Registering a user</td>
<td>L</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Receiving and forwarding messages</td>
<td>H</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Suspending communication</td>
<td>M</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parsing an XML document</td>
<td>H</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transforming an XML document</td>
<td>H</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Storing a set of heuristic rules</td>
<td>H</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**ID**
Chapter 11

Conclusion and further work

This chapter summarizes the achievements of the work presented in this report and points out some directions for future work.

11.1  Conclusion

This section describes the major themes covered and summarises the contributions of this report.

11.1.1  Important themes

The diversity and heterogeneity in mobile environments highly motivates the need for a framework supporting communication in such environments. XML has been chosen for data transfer due to its support for heterogeneity. However, XML processing requires considerable amounts of resources, and in a mobile environment limited by its scarce resources, a system optimising XML data to minimise transfer and processing time is needed. This report looks at the possibility to develop such a system which provides better support for exchanging XML-based information in a mobile setting. The report studies several aspects in this area.

**The verbose and redundant nature of XML data.** An important topic that has gained some attention in this report, is the verbose and redundant nature of XML data. The objective in this report has been on how these properties can be better handled.

**The challenges imposed on mobile computing.** This report has also focused on topics related to challenges within mobile computing. In addressing this problem, a model is suggested where a proxy server is
responsible for optimising XML-based information so that the limited resources in mobile environments may be better utilised.

11.1.2 Contributions of this report

The major contribution of this work has been the presentation of a model supporting XML-based information exchange in mobile environments and the description of a prototype based on this model. Other contributions include the elicitation of requirements and the suggestion of alternate solutions.

11.2 Future work

This section describes the directions for future work including, finishing the prototype, extending the proposed model and further research.

11.2.1 Finish the prototype implementation

The directions for finishing the prototype are as follows:

Implement the rest of the proxy server. As mentioned in Section 9.2.5 the implementation of the proxy server is basically not more than an empty shell. Implementing the proxy should be a central part in future work as this will allow for testing and validation of the model.

Implement the rest of the client. The client cannot interpret transformed XML documents received from the proxy server, this part of the client should be finished in parallel with the proxy server. As with the proxy server, it is important considering testing and validation.

Cover the postponed requirements. The architecture of the prototype should be extended to cover the postponed requirements. This new architecture should eventually be implemented.

11.2.2 Extensions of this work

The directions for extending the model and the prototype are as follows:

All traffic through the proxy server. Routing all traffic from a mobile client through the proxy server will probably be a good solution allowing the proxy to serve as a home base.
CHAPTER 11. CONCLUSION AND FURTHER WORK

Implementing a specialized parser. Vaaland [58] shows how a specialized parser can increase performance. Implementing such a parser on both the proxy server and the mobile client will probably save a considerable amount of resources.

Aggregation and synthesis of XML messages. Allowing the proxy server to collect and store information on the XML documents passed through the system would allow for analysis of the data and as a next step allow aggregation and synthesis.

11.2.3 Further research

Several issues have been left for further research, and include the following topics:

Looking at other models. As discussed in Section 10.1, the proposed model has several shortcomings. Suggesting and testing other models might reveal better solutions. Some interesting directions include the use of mobile agents to support Mobile Ad-hoc Networks and the creation of a middleware system.

The creation of heuristic rules. As pointed out in Section 7.3.2 the creation of good heuristic rules is not straightforward. Future research should concentrate on identifying significant factors and determine which primitives may be used to decide how an XML document should be transformed. This can be done through mathematical modeling, simulations and testing on real prototypes or systems.

Learning and self calibration. A future improvement of the system might be to make it a learning and self calibrating system where the heuristic rules are evolving to serve different needs and to further improve the handling of XML messages. Research determining how this can be done in an efficient and valid way is needed.
CHAPTER 11. CONCLUSION AND FURTHER WORK
Bibliography


BIBLIOGRAPHY

[23] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, first edition, 1995.


BIBLIOGRAPHY


Appendix A

Glossary

**Actor** defines a role which interacts with respect to the system.

**Aggregation** is a method for combining information to create a total.

**API (Application Programming Interface)** is a library of pre-defined routines which can be used by a developer to ease the task of programming.

**ASCII (American Standard Code for Information Interchange)** is the most common format for text files. Alphabetic, numeric or special character is defined using a 7-bit binary number, it defines 128 different characters in total [64].

**Asynchronous communication** is a way of communication where the reply is not needed straight away.

**Awareness** is a term used to describe the knowledge about the environment.

**Bandwidth** defines the available speed at which data can be sent on a network.

**BREW (Binary Runtime Environment for Wireless)** is an open-source development platform for wireless devices developed by Qualcomm.

**CLDC (Connected Limited Device Configuration)** provides the core functionality of J2ME targeted at small and limited devices.

**Communication protocol** defines a set of rules which end-points in a telecommunication use when they communicate [64].
**APPENDIX A. GLOSSARY**

Compression is a mechanism for shrinking the size of data.

Content provider is a party in my model which makes XML data available to the client for download.

Design pattern is a way of describing solutions of a common and specific problem in object-oriented software design.

Distributed system is a collection of components distributed over several hosts connected through a network.

DOM (Document Object Model) is a model parser recommended by the W3C.

DTD (Document Type Definition) defines and constrains the markup used in XML or SGML.

Framework is a structure supporting something.

Functional requirements is a function which must be covered in a system.

Heterogeneous is a term used to say that something consists of many different things.

Heuristic rule is a rule defined by reasoning or past experience.

HTML (HyperText Markup Language) is a markup language used to define Web page’s.

HTTP (HyperText Transfer Protocol) is a communication protocol for transferring web documents.

IDI Department of Computer and Information Science at NTNU

Java is a object-oriented programming language developed by Sun Microsystems.

J2ME (Java 2 Micro Edition) is a Java platform for small and embedded devices with limited resources.

J2SE (Java 2 Standard Edition) is a Java platform intended for workstations and laptops.

JVM (Java Virtual Machine) is responsible for interpreting compiled Java binary code, enabling a processor to perform the instructions within the Java program [64].
APPENDIX A. GLOSSARY

LAN (Local Area Network) is a group of computers interconnected through a common communication line within a small geographic area [64].

Logical view describes an architecture using key abstractions such as objects or object classes.

MANET (Mobile Ad-hoc Network) is a wireless network without any infrastructure.

Markup language is a language using markup symbols or tags to structure or define information.

Middleware is a software layer residing between user applications, operating systems, hardware platforms, and communication protocols.

Mobile client is the user party in my model.

Modifiability a quality attribute determining how easy it is to modify a system or program.

MIDP (Mobile Information Device Profile) is a part of J2ME layered upon CLDC which enables application programming on resource constrained devices.

Mobile agent is a program performing some service able to move from host to host.

Mobile computing defined by [20] to be the use of a portable computing device capable of wireless networking.

Model parser is a parser which reads through the whole XML document and builds a model of it in memory.

MOWAHS (MOBILE Work Across Heterogeneous Systems) is a research effort conducted at NTNU.

.NET is a collection of programming support especially for Web services developed by Microsoft.

.NET CF (.NET Compact Framework) is a smaller version of .NET targeted at limited devices.

Namespace provide a means to distinguish between elements and attributes from different XML vocabularies with the same name [27].
APPENDIX A. GLOSSARY

Nomadic network is a wireless network where the wireless part is connected to a wired backbone.

Non-functional requirements are requirements concerned with other aspects than the functionality of a system, such as quality, interfaces and conformance.

NTNU Norwegian University of Technology and Science.

Parser is a program responsible for extracting information, check for logical errors, and to validate the XML document if required.

PDA Personal Digital Assistant.

Process view describes how different parts of a system may execute simultaneously.

Proxy server is the main party in my model responsible for optimising XML documents.

Pull parser is a parser which reads through smaller parts of an XML document as they are requested from the application using the parser.

Push parser is a parser which reads through the whole document and notifies a listener whenever certain parts of the document are encountered.

Redundancy is repeated information.

RPC (Remote Procedure Call) is a protocol enabling a program on one computer to request the service of an other program located at another computer connected through a network. The protocol hides the details of the network for the participating programs [64].

SAX (Simple API for XML parsing) is a push parser proposed by members of the DEV.XML mailing list to solve some of the problems with the DOM parser.

SGML (Standard Generalized Markup Language) is a standard of how to define a document markup language or a tag set [64].

SMTP (Simple Mail Transfer Protocol) is a TCP/IP protocol, based on ASCII and used to send email.

SOAP (Simple Object Access Protocol) is a W3C specification based on XML which provides messaging and RPC.
APPENDIX A. GLOSSARY

**Synchronous communication** is a way of communication where the communicating parties are required to respond instantly.

**Synthesis** is a method for aggregating and summarizing data.

**Ubiquitous computing** is a term introduced by Mark Weiser describing the trend of computing devices becoming a natural part of our lives.

**UML (Unified Modeling Language)** is a standard notation for modelling object-oriented programs.

**URI (Uniform Resource Identifier)** is a way of identifying a point of content.

**Use-case** is a methodology used to analyse a system to elicit requirements. A use-case is made up of a set of possible sequences of interaction between actors [64].

**W3C (World Wide Web Consortium)** is a forum for information and communication concerned with developing interoperable technologies to “lead the Web to its full potential” [61].

**WAN (Wide Area Network)** is a “geographically dispersed telecommunications network” [64].

**WAP (Wireless Access Protocol)** is a specification for a set of communication protocols enabling wireless devices to be interconnected [64].

**Web services** a mechanism enabling interaction between online applications.

**Wireless networking** is a way of connecting devices enabling them to communicate using technology such as radio frequency.

**WLAN (Wireless LAN)** is a LAN with wireless connections used for communication.

**WSDL (Web Services Description Language)** is a language based on XML which provides a formal description of Web services.

**XHTML (eXtensible HyperText Markup Language)** is a reformulation of HTML 4 in XML.

**XML (eXtensible Markup Language)** is a markup language defined by W3C and used to store and transfer structured data.
XML Schema is a way of describing the interrelationship between the attributes and elements of an XML document [64].
Appendix B

XML technologies

This chapter describes some of the technologies that have been layered upon XML.

B.1 The Extensible Stylesheet Language (XSL)

XSL is a language for expressing stylesheets [2]. It consists of a language for transforming XML documents (XSLT), an expression language used by the transformation language to access or refer to parts of an XML document (XPath) and an XML vocabulary for specifying formatting semantics (XSL-FO).

An XSL stylesheet is used to express how the content of an XML document or data file should be presented on some presentation medium. Table B.2 presents an example.

B.1.1 XSL Transformations (XSLT)

XSLT is a language for transforming an XML document into another XML document [13]. XSLT can for instance be used for transforming XML into HTML to enable viewing in a browser. An XSLT document contains template rules comprising of a pattern and a template.

During the transformation of an XML document the nodes in the documents tree-structure are compared to the template-rule patterns by an XSLT processor. Whenever a match is found, the template is written into an output tree which can be further serialized into an XML document or some other
Table B.1: A XML document example

```xml
<?xml version="1.0" encoding="UTF-8"?>
<class>  
  <student id=1357>
    <name>
      <first>John</first>
      <last>Smith</last>
    </name>
    <grade>A</grade>
  </student>
  <student id=1467>
    <name>
      <first>Jimmy</first>
      <last>S. Student</last>
    </name>
    <grade>C</grade>
  </student>
</class>
```

Table B.1: A XML document example

Table B.3 shows the result of applying the XSLT stylesheet, shown in table B.2, to the example XML document, shown in table B.1.

### B.1.2 XPath

XPath is a language for filtering and selecting values. It is used to identify particular parts of an XML document. XPath can indicate nodes in the XML’s tree-structure by position, relative position, type and content [27]. XPath is used by XSLT to match and select particular elements of the input document for copying into the output document.

### B.1.3 XSL Formatting Objects (XSL-FO)

XSL-FO is an XML application for describing the layout of text on a page. It uses elements that represent pages, blocks of text, horizontal rules and many more to achieve this.
APPENDIX B. XML TECHNOLOGIES

Table B.2: A XML stylesheet

```xml
<?xml version="1.0" encoding="UTF-8"?>
A graduate student
A graduate student
```

Table B.3: Result of applying the XML stylesheet given in table B.2.

## B.2 Future extensions

The W3C XML working groups are continuously working on extensions and improvements to XML. The following subsections describe two of the more relevant extensions.

### B.2.1 XInclude

To facilitate modularity a mechanism for inclusion is needed. XInclude is a W3C Proposed Recommendation [1] providing such a mechanism. In other words, XInclude provides a means for merging XML documents. Controlling this mechanism is done through the use of existing XML constructs - elements, attributes, and URI references.
APPENDIX B. XML TECHNOLOGIES

B.2.2 XFragment

XML documents can easily grow large, a mechanism splitting a large document into smaller pieces is hence needed. XFragment is a W3C Candidate Recommendation providing such a mechanism. XFragment consists of two parts; the fragment body and the Fragment Content Specification describing location and path to the fragment.
Appendix C

Enabling technologies for wireless networks

This chapter describes the underlying technology in networks.

C.1 Network Carriers

Network carriers define how data is transmitted in the network in terms of range, data rate and encoding schemes. In the following I try to give a brief overview of some of the existing network carriers for different types of wireless networks.

C.1.1 Cellular

Cellular technologies are capable of delivering voice and data services to users with handheld devices. During the last few years multimedia services on mobile phones have been introduced to end-users, and the cellular technologies are moving towards what is known as the third generation of mobile telephony services. The more important technologies are:

Global System for Mobile communication (GSM) is a widely deployed digital mobile telephony system which operates in the 900MHz and 1800Mhz frequency band. GSM is a circuit switched system based on TDMA (time-division multiple access).

General Packet Radio System (GPRS) complements the GSM system. GPRS is a packet-based service which provides data rates from 56 Kbps up to 114 Kbps.
APPENDIX C. ENABLING TECHNOLOGIES FOR WIRELESS NETWORKS

Enhanced Data GSM Environment (EDGE) is a faster version of the GSM system designed to deliver data rates up to 384 Kbps.

Universal Mobile Telecommunications Services (UMTS) based on GSM is a 3G standard. It supports bit rates up to 2 Mbps and is based on FDD (frequency division duplex) and TDD (time division duplex) [47]. The deployment of UMTS was originally planned to start around 2002, but it has been delayed because of some problems with the technology.

Code Division Multiple Access (CDMA) optimises the available bandwidth by allowing a number of signals to occupy a single transmission [64]. CDMA2000 is a 3G ITU standard. CDMA2000 1x supports packet data service up to 144 Kbps while CDMA2000 1xEV-DV promises data rates up to 3 Mbps.

C.1.2 Wireless Local Area Networks (WLAN)
A WLAN is the counterpart of a regular local area network, using radio waves instead of cable connections for data communication.

IEEE 802.11 is a family of IEEE standard for wireless communication, often referred to as WiFi. It uses either carrier sense multiple access, frequency-hopping spread spectrum or infrared pulse position modulation for data transmission. IEEE 802.11b is the most widely deployed standard of those contained in the 802.11 family. It operates in the 2.4 GHz ISM (industrial, scientific, and medical) band and delivers data rates up to 11 Mbps with a range of approximately 100 meters [4].

HiperLAN2 is being developed by the European Telecommunications Standardisation Institute (ETSI). It operates in the 5 GHz ISM band and supports data rates up to 54 Mbps with a range between 30 and 150 meters [4].

C.1.3 Short-range networks
Short-range networks can be used to set up personal area networks (PAN), which reside on one person and create a network of portable devices. One common application is the use of a mobile phone together with a handsfree set, both bluetooth enabled. Important enabling technologies include:
APPENDIX C. ENABLING TECHNOLOGIES FOR WIRELESS NETWORKS

Bluetooth is a short range industry standard for wireless communication. Its range is up to 10 meters (30 feet), it serves data rates up to 720 Kbps and operates in the unlicensed 2.4 GHz range of the radio spectrum [6]. Bluetooth gives you the ability to easily connect different devices, like your cell phone and your hands-free set. Bluetooth’s advantages are its relatively cheap pricing, its low power consumption, and its ability to be easily integrated into almost any electronic devise.

IrDA is a family of specifications defined by the Infrared Data Association [29] which operate in the infrared band. They can deliver data rates up to 4 Mbps and have a range of up to 10 meters. Unfortunately their line of sight requirements makes them unsatisfactory for many uses.

IEEE 802.15 is a family of standards defining low cost and short range wireless solutions. Bluetooth is included in this family (802.15.1). Other standards include the 802.15.3 designed for high data rates and 802.15.4 designed for very low data rates [4]. The main advantage of these standards is their cheap pricing.

C.2 Network protocols

The following describes some of the commonly used protocols on the Internet.

The Internet Protocol (IP): IP provides a means for addressing hosts in a network. The common IP version 4 allocates 32 bits for addressing, which gives an inadequate address space. Hence, IP version 6 was proposed to allow a larger address space. To meet the needs of mobile users Mobile IP version 4 was designed by the Internet Engineering Task Force (IETF) in 1996 [59]. However, it has not been widely deployed due to some shortcomings including the limited address space. Hence MIPv6 has been developed supporting a larger address space and allowing constant connection.

Transmission Control Protocol (TCP): TCP is the most widely adopted transport protocol on the Internet. It is intended for use on unreliable packet switched networks and is a highly reliable host-to-host protocol.

The User Datagram Protocol (UDP): UDP is a connection-less protocol allowing hosts to exchange messages with a minimum of overhead. UDP is an alternative to TCP when a guarantee for the delivery of each datagram is not required.
APPENDIX C. ENABLING TECHNOLOGIES FOR WIRELESS NETWORKS

C.3 Application protocols

A number of application protocols exist. The most successful is, without a doubt, the Short Message Service (SMS) which runs on top of GSM networks and provides the users with a mechanism for sending short messages between mobile phones.

The Wireless Application Protocol (WAP) has emerged as a standard wireless protocol and a browser for “limited-display-capable” devices, allowing Internet access to PDA’s and mobile phones. It has three major building bricks; The Wireless Markup Language (WML) is based on XML, and makes optimal use of small screens, the WMLScript makes WML pages dynamic and the default picture format Wireless Bitmaps (WBMP) [35]. WAP has had a slow start, but with the new networks supporting higher bit rates and the launch of WAP 2.0 it will probably become more popular.

Other application protocols include the Hyper Text Transfer Protocol (HTTP) which is a protocol for exchanging data.
Appendix D

Mobile device examples

As mentioned a number of different mobile devices exist, in this section we take a closer look at two of them, the PDA and the mobile phone.

D.1 Mobile Phone

Mobile phones have gained an enormous popularity, and as Mark Weiser foresaw in his vision on ubiquitous computing they have merged into our everyday life. In Norway there are approximately 4.4 million subscriptions which is more than the entire Norwegian population. We depend on mobile phones in our everyday activity.

Figure D.1: Sony Ericsson’s T610. One of the worlds most popular phones. Integrates both Bluetooth and IrDA.
APPENDIX D. MOBILE DEVICE EXAMPLES

Figure D.2: HP’s iPAQ h4155 Pocket PC. Comes with a 400MHz CPU and 64 MB of SDRAM. Integrates WiFi, Bluetooth and IrDA.

Figure D.3: Sony Ericsson’s P910i. Provides both PDA and mobile phone capabilities. Integrates both Bluetooth and IrDA.

The mobile phone has experienced an evolution without comparison during the last decade. They have moved from being impractical monsters with batteries weighing a couple of kilos to small, handy devices with cameras, music playback and other appropriate features. Mobile phones have become small computers allowing a range of new applications.

D.2 Personal Digital Assistant (PDA)

A PDA can be characterized as a handheld computer as it provides computing and information storage and retrieval. Typically they are used for personal scheduling, storing addresses and note-entering. Input is usually given through either small keyboards or touchscreens for handwriting and pointing.

Mobile phones and PDA’s are merging. Many mobile phones are showing up with similar capabilities as PDA’s such as storing and retrieval of information. Figure D.3 shows a typical example of a merger between the mobile phone and the PDA.
Appendix E

Platforms for mobile and limited devices

This chapter describes some of the most well known development platforms for mobile devices.

E.1 Java 2 Micro Edition (J2ME)

J2ME [53] is a Java platform, presented in June 1999 [28], specifically designed for consumer electronics, portable, and embedded devices. It was developed as a joint effort between Sun Microsystems and the Java Community Process (JCP)\(^1\). J2ME provides developers with a rich environment for the development of stand-alone applications for a range of resource constrained devices, hereby removing the need to develop native application for each device. The hardware demands have been greatly reduced compared to other Java platforms to accommodate limited devices. For instance the memory footprint has been reduced to a fraction.

E.1.1 Architecture

The J2ME architecture comprises of a number of configurations, profiles, and optional packages which a developer can choose from and combine to meet certain requirements. A configuration specifies a runtime environment and acts as the Java platform on a limited device [50]. It consists of a virtual machine and a set of basic class libraries. A profile is a specification defining a set of Java application programming interfaces (API) developed to address

\(^1\)an expert group representing a number of companies
APPENDIX E. PLATFORMS FOR MOBILE AND LIMITED DEVICES

specific types of devices. Helal [28] describes the J2ME architecture as a three layered architecture consisting of:

![Figure E.1: The J2ME architecture](image)

**The profile layer** defines a minimum set of Application Programming Interfaces (API) available on a particular device. It is built on top of the configuration layer to provide extra features (networking, persistent storage, user interface, etc.) not included in the core functionality defined by the configuration. As apposed to the configuration a device can support several profiles. The more common profiles of J2ME include the Mobile Information Device Profile (MIDP) [54] and the Foundation Profile [50]. Other profiles include the Information Module Profile, the Personal Basis Profile and the Personal Profile.

**The configuration layer** defines the runtime environment for a particular category of devices. It comprises a minimum set of core Java classes and a virtual machine. J2ME includes the Connected Limited Device Configuration (CLDC) [51] and Connected Device Configuration (CDC) [50].

**The Java Virtual Machine layer** is needed to enable Java byte-code to be executed on any operating system. The layer provides an interface between the binary code and the processor of the device, e.g. it translates the binary code into machine code. The virtual machine is customised for a particular devices OS and hardware to ensure performance. It supports a particular J2ME configuration [44, 28].
APPENDIX E. PLATFORMS FOR MOBILE AND LIMITED DEVICES

Virtual Machine (KVM) [51] supports the CLDC and the Connected Virtual Machine (CVM) supports the CDC.

E.1.2 Advantages

An important advantage of J2ME is its wide adoption as an application development platform on mobile phones. Nokia estimates that more than 250 million units supporting J2ME will have been sold by the end of 2003 [45], and Sun Microsystems lists hundreds of devices supporting the J2ME platform [52].

Another significant feature of J2ME is the ability to add optional packages bringing extra features to the platform. One example is the J2ME Web Services API.

E.2 .NET Compact Framework

The .NET Compact Framework (CF) is a Microsoft initiative which was generally available in early 2003. It offers a programming environment and has been designed for resource constrained devices. The .NET CF allows developers with experience from the full .NET Framework to easily port their knowledge to build .NET CF applications. The reason for this is that the CF is a subset of the full .NET Framework. Key features include the integration of XML Web Services as a platform component, and the ability to run client-side code in a safe environment through the managed code model [44].

E.2.1 Architecture

The .NET Compact Framework is designed to be portable and hence able to run on a number of operating systems. To enable this the architecture includes a Platform Adaption Layer (PAL) which can be adapted to match the common language runtime’s (CLR) requirements and the specific host operating systems capabilities [42]. Above the PAL the CLR is implemented which provides an execution environment. On top of the architecture the programming infrastructure itself is implemented. This infrastructure is highly adaptable allowing developers to extend the environment with class libraries adding extra features [18]. Figure E.2 shows how the .NET Compact Framework architecture is layered.
APPENDIX E. PLATFORMS FOR MOBILE AND LIMITED DEVICES

Figure E.2: The .NET Compact Framework architecture, comprising the Platform Adaption Layer, a Common Language Runtime and the programming infrastructure. (Adopted from [42])

**.NET Compact Framework** is the programming infrastructure. It includes base class libraries providing core functionality for all applications. These libraries are compliant with the ECMA Common Language Infrastructure (CLI) standard\(^2\). The programming infrastructure also includes higher level functionality such as XML Web Services, a graphics and Windows Forms library and data-access through the ADO.NET library.

**Common language runtime** is an execution environment which enables cross-language integration. The CLR can run the Microsoft Intermediate Language (MSIL). This is done using a just-in-time (JIT) compiler which translates the MSIL into machine code. The CLR also includes memory management and garbage collection.

**Platform Adaption Layer** provides an abstraction layer between the host operating system and the CLR. The layer enables portability of the

\(^2\)The CLI defines how to create applications with high-level languages avoiding the need to rewrite each application to accommodate different system environments [42].
APPENDIX E. PLATFORMS FOR MOBILE AND LIMITED DEVICES

.NET Compact Framework. This is done through adapting it to match the CLR’s requirements and the new host operating systems capabilities [42].

E.2.2 Advantages

The .NET Compact Framework has been designed from the ground up giving special consideration to XML Web services. This in turn equals the difference in the ability to access distributed data between stationary computers and smart devices.

Another advantage is clearly that developers familiar with the .NET Framework do not need to acquire new skills to develop Compact Framework applications.

E.3 Binary Runtime Environment for Wireless (BREW)

BREW [7] is an open environment developed by Qualcomm and released in the beginning of 2001. Initially BREW only supported CDMA chipsets and technology, but it is now independent of the wireless technology employed. BREW aims to be a complete solution for both development and distribution of wireless technology. Three key components support this goal;

BREW Software Development Kit (SDK) is used by developers to create applications which can run on BREW enabled devices. It is based on Windows and includes an emulator, documentation and the API.

BREW Distribution System (BDS) provides an easy way to distribute applications for developers. BSD takes care of billing, testing and application distribution.

BREW Application Platform is the execution platform required to run BREW applications on a mobile device. It is optimised to make downloads of BREW applications and the execution of them easier and more efficient.

In terms of market share BREW has barely reached the European market, counting one operator in Romania. In 2004 BREW operators exist in 24 countries mainly in Asia and America.
E.3.1 Architecture

The complete BREW architecture consists of a runtime environment, a development environment (the SDK) and a distribution system (the BDS). The runtime environment is a layer resident in a BREW enabled device. It is placed right above the hardware on top of the Application Specific Integrated Circuit (ASIC). This layer is hardware independent and it supports a number of operating systems. As can be seen in figure E.3 the device architecture can be extended to provide a number of services.

![BREW device architecture](image)

Figure E.3: BREW device architecture (Adopted from [8])

E.3.2 Advantages

The low memory footprint of only 150 Kbytes is BREW’s major advantage. The low footprint allows BREW to run on a number of handheld devices. This is probably also one of the main reasons for its wide adoption in Asia and America.

Another advantage is that every application to be run on BREW enabled handsets need to be tested and verified to obtain a unique key which enables operation on BREW enabled handsets.
Appendix F

Notation

This appendix gives an explanation to some of the notation used in this report. Table F.1 explains the textual use-cases used in Chapter 7.

<table>
<thead>
<tr>
<th>Use-case name</th>
<th>Initialisation of the system</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iteration</td>
<td>Filled or Facade. Filled is an elaboration of Facade (see Chapter 4 in [34] for a closer description.)</td>
</tr>
<tr>
<td>Abstract</td>
<td>A summary of the use-case.</td>
</tr>
<tr>
<td>Trigger</td>
<td>The event triggering this use-case.</td>
</tr>
<tr>
<td>Main path</td>
<td>1: A stepwise explanation of the events in this use-case. 2: ....</td>
</tr>
<tr>
<td>Alternate path</td>
<td>Explains, if any, alternate paths.</td>
</tr>
<tr>
<td>Exception path</td>
<td>Explains, if any, exceptions from the main path.</td>
</tr>
<tr>
<td>Preconditions</td>
<td>Describes the conditions which must be met to allow this use-case to happen.</td>
</tr>
<tr>
<td>Author</td>
<td>The author of this use-case.</td>
</tr>
<tr>
<td>Date</td>
<td>The date this use-case was written.</td>
</tr>
</tbody>
</table>

Table F.1: Use-case: Explanation of the textual use-cases