Developing a Conceptual Browser for Mobile Phones

Noel Serjik Zargarian
Developing a Conceptual Browser for Mobile Phones

Noel Serjik Zargarian

Master’s Thesis in Computer Science (20 credits)
Single Subject Courses
Stockholm University year 2006
Supervisor at Nada was Ambjörn Naeve
Examiner was Yngve Sundblad

TRITA-CSC-E 2006:041
ISRN-KTH/CSC/E-06/041-SE
ISSN-1653-5715
Abstract

This report discusses a new approach of structuring, managing and presenting digital information in the context of online learning environments on wireless devices, such as mobile phones. The information is structured using conceptual modeling – a visual modeling technique inspired by UML (Unified Modeling Language) – and presented as context-maps. A context-map consists of connected concepts filled with content materials which gives the user the possibility to explore the concept further. The discussion will also include emerging Semantic Web activities and our vision of extending this into what we call the Conceptual Web. One of the main objectives of this Master’s project has been creating means for displaying context-maps and exploring their concepts on wireless clients which we managed to fulfil by presenting a client-server solution.

Utveckling av begreppsbrowser för mobila telefoner

Referat

# Contents

1 Introduction .......................... 1  
   1.1 Administrivia .......................... 2  
   1.2 Background .......................... 2  
   1.3 Problem definition ................. 3  

2 The World Wide Web  ................. 5  

3 The Semantic Web .................... 7  
   3.1 XML .............................. 7  
   3.2 The Resource Description Framework: RDF ............ 7  

4 The Conceptual Web ................. 11  

5 Concept Browser ...................... 13  
   5.1 Definition of related terms .......... 13  
   5.2 Design principles of a Concept Browser ........ 14  
   5.3 Using ULM to Model Context Maps ........ 15  

6 The Conzilla Browser:  
   A First Prototype of the Concept Browser .......... 17  
   6.1 Demonstration ..................... 18  
   6.2 Feature summary ................. 20  

7 Mobile Concept Browser Analysis .... 21  
   7.1 Feasibility study .................. 21  
   7.2 Similar projects .................. 22  
   7.2.1 WAP browsers using WML .......... 22  
   7.2.2 WebViewer using HTML .......... 22  
   7.3 The Requirements .................. 23  
   7.3.1 Functional Requirements .......... 23  
   7.3.2 Communication requirements .......... 23  
   7.3.3 General requirements .......... 23  
   7.3.4 Semantic Web conformance: RDF based context maps .... 24
## Table of Contents

8 Platform and Tools 25 8.1 The Moplun 25 8.2 The J2ME 25 9 Design of the Mobile Concept Browser 27 9.1 Definitions 27 9.1.1 Concept 27 9.1.2 Concept-relation 28 9.1.3 Context-map 28 9.2 General design overview 29 9.3 The Communication 29 9.3.1 The request-response protocols 29 9.3.2 The request 30 9.3.3 The response 30 9.3.4 Wrapping context-maps: Data interoperability 30 10 The Application Structure and the Conceptual Data Model 31 10.1 The Java classes 32 10.1.1 Conceptual browsing: The abstract layer 32 10.1.2 Graphics 33 10.1.3 Communication 34 10.1.4 Miscellaneous classes 34 11 The Browser 35 11.1 Introduction 35 11.2 User interaction 35 11.3 Starting the application 36 11.4 Graphical user interface and functionalities 37 11.5 Flowchart: The browser’s possible states 41 11.6 Real test 42 11.7 Further improvements 42 12 Project Summary and Conclusions 43 12.1 Future work 43 Bibliography 45 Appendices 47 A RDF 49 A.1 RDF structure 50  A.1.1 RDF class and property 51 A.2 Ontologies 52
B The Protocol
   B.1 The Request .................................................. 53
   B.2 The response protocol ...................................... 54
Chapter 1

Introduction

The World Wide Web (WWW), or the Web, has over the recent years evolved into an ever growing global network, interconnecting a vast number of communities, allowing them to share experiences with each other in a rather convenient way. Various kinds of tools have been developed and marketed with the purpose of augmenting the web and enhancing our ability of using it, including tools designed to promote online learning environments and information management which is the core point of this project.

In spite of the fact that the market is being overwhelmed by rather sophisticated tools (applications), the development of online technologies still suffer from major obstacles which limit the fully usage of the Web’s enormous potential. Platform interoperability difficulties and lack of a rigid data structure which enables machines to understand and process data automatically are two of the main hindrances.

Another important, and to this thesis project a central, issue is the way information is being organized and presented which, considering the immense amount of information available on the Web, could be described as inefficient and chaotic.

In this thesis we will address these problems and discuss about related, ongoing activities within the communities involved. We will also present tools designed to support the dynamic nature of online learning environments.
CHAPTER 1. INTRODUCTION

1.1 Administrivia

This thesis project is a part of the distributive interactive learning environment being developed by the Knowledge Management Research¹ (KMR) group at Centre for user-oriented IT Design² (CID), Royal Institute of Technology (KTH), Stockholm.

1.2 Background

The Web, with its vast network infrastructure, has created an excellent environment for exchanging online information. This enormous data repository requires a new set of information-structure and management system in order to simplify and improve the Web’s usability.

The KMR group is involved in research fields concerning knowledge management and online learning environments. The research is based on Knowledge Manifold (KM) [2], an open educational architecture designed for individualized and interactive learning. The KM was introduced by Ambjörn Naeve who is the head of KMR group. In his publication [2], Naeve characterizes the school system as being layered, close architectures based on teacher centric, knowledge pushing principle, as opposed to learner centric, interest-oriented, knowledge pulling approach which he advocates. He describes his ideas and introduces a new kind of approach in ”The Garden of Knowledge as a Knowledge Manifold” [1].

The research done by KMR group aims to improve our ways of structuring and communicating information, laying great emphasis on developing a dynamic, learner-centric, educational architecture. As Naeve expresses in his writings [3]:

“The overall aim of the knowledge manifold framework is to provide patterns that support and enhance individualized learning in every possible way.”

To put these ideas into effect the KMR group has developed several tools, among them a powerful Concept Browser, called Conzilla which is designed to support exploring and organizing information. The information-management and presentation technique is based on Conceptual Modeling (see chapter 4). The technique which is developed by A. Naeve aims to capture our underlying conceptual thought patterns of a given topic in its actual context and present them in form of context-maps, somewhat similar to mind-maps but more structured.

During the last decade there have been remarkable improvements within the mobile phone technologies. Nowadays, mobile phones not only enable wireless communication but they also serve as a thin personal computer as well. This has given software developers better possibilities to create suitable applications for this segment of devices. Others, such as established software vendors, have moved their positions into the wireless world by introducing a stripped down version of their

¹http://kmr.nada.kth.se
²http://cid.nada.kth.se
1.3. PROBLEM DEFINITION

existing products. This strategy is an ideal opportunity also for the KMR group to expand their activity area to include the wireless community as well, which is why this project has been initiated.

1.3 Problem definition

The Conzilla browser is a powerful knowledge management tool. In spite of its simple, user friendly interface, it is built upon an extensive design structure which makes it unsuitable for the product category of limited resources, e.g. mobile phones. The main purpose of this Master’s project is to design and develop a concept browser prototype which enables conceptual browsing on mobile phones. This should conform to the design principles of a concept browser described in chapter 5.2 to such an extent that is attainable within the limits of this specific application domain.
Chapter 2

The World Wide Web

The Web, as it is today, has become a large but rather chaotic information repository where the process of retrieving a relevant piece of information is hardly as easy as depositing it. Perhaps the most annoying thing, while searching the Web, is the feeling of getting lost in this overwhelming, unorganized information flow which gradually paralyzes the individual’s ability to digest and integrate observed information. This, in combination with the radically increasing use of information over the Internet during the past few years, has made us to realize that the current structure of the Web is not powerful enough to fulfill our growing demands which seems to evolve constantly in complexity. Currently, documents of various kinds are being stored in this network with no rigid contextual based classification structure which makes the searching process almost unbearable. In the following examples we will try to expand some of these structural problems inherent on the Web.

Example 1: While searching the Web for a given topic you often end up with a search-result containing different combinations of the requested subject. Search engines (or agents as they often are called), simply search the whole Web in order to find a string matching your request, regardless of the context that you had in mind. Why? Because, we use word(s) as input data which semantically are comprehensible to us humans, but barely to the machines. For them, word(s) are just binary digits arranged in a certain order and all they are capable to do is to collect documents (links) in which they find a string matching the input data, i.e. our word(s). Furthermore, most of these documents, or hyper links, are probably irrelevant and should be discarded by the search engine.

Example 2: In another scenario, one can imagine what happens if the information we try to retrieve exists purely in an image format, i.e. with no textual description at all about the image. Assume that you are interested in knowing the geographical location of the Stockholm city which I happen to have on my homepage in a png\(^1\) format. Search engines will probably fail to capture this

\(^1\)Portable Network Graphics.
CHAPTER 2. THE WORLD WIDE WEB

document, of obvious reasons.

The first example on the previous page addresses the lack of machine-understandable data model (machine semantics) which corresponds to the human's interpretation of the same data, i.e. a data model that would make it possible for computers to understand the humans semantics.

In the second example, the information is implicit in rich media, and thus unrecognizable to the search engines. Both cases point out the lack of proper annotation mechanism with which different properties (=meta-data) of a document, such as author, genre, publisher etc., can be added (annotated) to the document by the creator.
Chapter 3
The Semantic Web

The World Wide Web Consortium (W3C) started the Semantic Web initiative with the vision of creating a universal medium for the exchange of data, making data understandable to and process-able by machines. The stated purpose of this initiative is to design a structural data model in which data are defined, organized and linked in a way that it can be processed automatically by computers. The pay-off is that data can be integrated and reused across various applications and platforms (=semantic interoperability), making it easier for computers and people to work in cooperation [25]. Semantic web is based on RDF (Resource Description Framework), which forms the data structure, and is serialised by XML.

3.1 XML

In an earlier attempt to overcome interoperability problems between different platforms and applications, W3C introduced eXtensible Markup Language — or XML for exchanging data over the Internet. XML is designed to describe/label and organize data in a tree-like structure.

Using XML has proved to be very efficient in exchanging of wide variety of data on the Web, but still, despite its proven functionality, there are some other fundamental issues which XML can not cope with by itself. For instance, XML uses DTDs (Document Type Definition), or even more advanced XML-schema, which provides a syntax for a XML document, but the semantics of DTD are implicit, meaning that elements defined in DTD or XML-schema can have different semantics. Thus, an exchange of XML documents works well if the parties involved have agreed to a DTD/XML-schema beforehand, but not otherwise which consequently limits the semantic interoperability.

3.2 The Resource Description Framework: RDF

RDF is a meta-data (=data about data) modeling language which can be used to describe resources on the web. RDF consists of three components [18]:

7
CHAPTER 3. THE SEMANTIC WEB

Resources: Everything described by RDF is a resource. A resource can be a web page, a book or a statement issued by, for instance, W3C, as long as it can be identified by an URI.

Properties: A property is an attribute, a characteristic of a resource with which it can be described. Author, Title and Publisher are examples of properties within the context of literatures. Each property has a specific meaning and a value. For instance, we can define the property author as the writer of a book and it can take the string literal, 'Roal Dahl', as its value. The property value can also be a resource with a unique URI, such as, “http://example.org/greatAuthors/Roal_Dahl”.

Statements: In order to describe data, RDF uses statements which are defined as a set of triples: A subject, a predicate and an object which correspond to Resource, Property and Property value.

RDF statements can be shown using directed labeled graphs where nodes (drawn as ovals) and arcs represent resources and predicates respectively. Figure 3.1 illustrates a basic RDF graph.

![RDF Graph](image)

Figure 3.1. A basic RDF graph representing the three parts of a statement.

One can interpret figure 3.1 as:

“<subject> HAS <predicate> <object>"."
3.2. THE RESOURCE DESCRIPTION FRAMEWORK: RDF

Describing data using RDF is a straightforward procedure. Let us begin with a simple example in which we wish to describe the resource report.html which is uniquely identified by its URL, “http://example.org/doc/report.html” (see figure 3.2).

![Figure 3.2. A RDF-graph.](http://example.org/doc/report.html)

Note that the object is drawn using a rectangle. This is an accepted convention whenever the resource is a string literal\(^1\). Furthermore, the arc always starts at the subject and points to the object of the statement. As it was mentioned before, a labeled arc is a resource which must be defined by an unique URI as opposed to string literals.

We can elaborate on this example by adding further information about this document, e.g. the date of publication or a short key words describing the document. Figure 3.3 demonstrates how this can be done in RDF.

![Figure 3.3. An extended RDF-graph.](http://example.org/doc/report.html)

---

\(^1\)Literals are used to identify values such as strings, numbers, dates etc. by means of a lexical representation [17].
Figure 3.3 represents three statements:

“The report.html is written by Noel Zargarian.”

“The report.html was published in 2004-02-20.”

“Key words describing report.html are: Concept browser, Semantic Web and RDF.”

It is also possible to uniquely identify each resource (see figure 3.4). For the moment, we use imaginary URI-references where “ex.” is an abbreviation of “http://example.org/elements/”.

![RDF-graph with its unique identifiers](image_url)

Figure 3.4. A RDF-graph with its unique identifiers.

This was a brief introduction to the RDF. The reader is referred to Appendix A where we continue to describe the structure of the RDF in details.
Chapter 4

The Conceptual Web

The emerging Semantic Web initiative presents a solution to some of the main problems inherent in the Web, including data compatibility and platform interoperability. These improvements, however, will not bring sufficient clarity to the ruling chaos of hyper-linked-enabled Web documents by itself. Still, a common user will not be able to grasp an overall contextual overview of the contents being studied, something that usually results in difficulties in absorbing and integrating the information into a contextual whole.

The KMR group presents a new approach to this dilemma which is called the Conceptual Web, the fundamental of which is based on conceptual modeling—a technique, which according to A. Næve [5], provides a human-understandable semantics for both abstract ideas and concrete resources. The idea of conceptual modeling is inspired by the way UML is used in designing object oriented softwares where classes and class relations can be thought of as concepts and concept relations, which, together they manage to capture the conceptual structure of the underlying information, e.g. Web contents. The concepts are linked together through concept-relations and presented in form of context-maps (typically UML diagrams). Exploring these maps in the Conceptual Web requires a Concept Browser. In the following section we will discuss the principles of such a browser.

The information structure of the Conceptual Web is based on RDF, and hence, it shares the advantages that come with it in terms of distributivity and scalability [5]. It creates a platform where the traditional content-oriented navigation through Web documents is replaced by concept-oriented approach with a clear separation between the content and the context.

The Conceptual Web is a long term vision, thought as an extension of the existing Web which fits naturally on top of the emerging Semantic Web as a complementary layer.
Chapter 5

Concept Browser

So far, we have given a basic description about the Conceptual Web and the tool which makes exploring concepts possible, i.e. the Concept Browser. In this chapter we will continue to describe the roles and definitions upon which the idea of conceptual browsing is built.

5.1 Definition of related terms

Some of the terms described below occurs frequently in this thesis and form an important part of the vocabulary used in the context of conceptual browsing. These definitions are taken from the report [4] written by A. Næve.

Concept: Representation of some thing.

Context: Graph containing concepts as nodes and concept-relations as arcs.

Context-map: Graphic representation of a context.

Content: Information linked to a concept or a concept-relation. This could be a Web page, a PDF file or a motion picture.

Resource: Concept or concept-relation or context or content.

Contextual- Neighborhoods and Topologies:

Let $S$ be a set of concepts, and let $C$ be a concept in $S$. A context involving a subset of concepts in $S$ that contains $C$ is called a contextual neighborhood of $C$ in $S$. The contextual topology on $S$ is the set of all contextual neighborhoods (in $S$) of concepts of $S$. If a concept $C$ has no contextual neighborhood involving other concepts from $S$, then $C$ is called an isolated concept in $S$ [4].

For example, assume that you are doing research on the subject ‘interest rate’ and its impact on the economy (= economical context). Another researcher might be interested in what kind of influences a variation in the interest rate might have on the political or the social context. Here, the concept ‘interest
rate’ appears in three different contexts – the economical, political and social context. Thus, according to the definition above, each of these three contexts form a contextual neighborhood of the concept ‘interest rate’ (in the set of all concepts involved in these three contexts).

5.2 Design principles of a Concept Browser

According to A. Naeve [4], a concept browser conforms to the eight major design principles which build the backbone of the Conzilla browser. These principles are described as:

i. Separate the content of a concept or a concept-relation from its contexts. This supports the reuse of conceptual content across different contexts.

ii. Describe each separate context in terms of a context-map, preferably expressed in the Unified Modeling Language process (UML) which is an international industry standard for this purpose [12].

iii. Allow neighborhood-based contextual navigation on each concept and concept-relation by enabling the direct switch from its presently displayed context into any one of its contextual neighborhoods.

iv. Assign an appropriate set of resources as the content-components of each appropriate concept and/or concept-relation.

v. Label each resource (concept, concept relation, context or content-component) by making use of a standardized data description (= meta-data) scheme.

vi. Allow meta-data based filtering of the content-components through context-dependent aspect-filters. This enables the presentation of content in a way that depends on the context.

vii. Allow the transformation of a content-component, which is also a context map, into a context (henceforth called contextualization).

viii. Support lateral thinking by introducing a concept bookmaker, which allows concepts as well as contexts to be interactively constructed from content according to a menu of different content-gathering principles.
5.3 Using ULM to Model Context Maps

A context map, as mentioned above, describes concepts from a certain contextual aspect using some kind of graphical representation technique. The logical information structure of concepts contained in a context map is developed by A. Naeve and is called ULM (Unified Language Modeling) [2], [3], which is a dialect of the UML. The technique— which is a project in progress—is intended for modeling and describing the interrelations between different concepts, clearly reflecting the visual semantic of the underlying contextual model. In addition to its features inherited from UML, the ULM technique also introduces a new kind of a dependency relation, called a Classification, which symbolizes the relationship between an instance and its corresponding concept-type. This new relation is depicted as a dashed arrow with filled triangle ' - - - - ' in the figure\textsuperscript{1} 5.1.

\begin{figure}[h]
\centering
\includegraphics[width=0.6\textwidth]{uml.png}
\caption{ULM vers. UML.}
\end{figure}

\textsuperscript{1}The figure is taken from the report [3].
Figure 5.2 shows how the Classification relation can be applied in a ULM diagram describing vehicles.

As it can be seen from the example in figure 5.2, the ULM uses a different terminology for describing certain relationships. For instance, the phrase is-a, which is often used to label a generalization/specification relationship between a superclass and its subclass in UML, is substituted with the phrase kind-of which, according to A. Naeve, reflects a correct and more adequate interpretation of the same relationship [2]. For a more detailed description about ULM, we refer to the report The Human Semantic Web\textsuperscript{3} written by A. Naeve [11].

\textsuperscript{2}The figure is taken from the report [3].

\textsuperscript{3}The report is to be published in the International Journal On Semantic Web And Information System.
Chapter 6

The Conzilla Browser: A First Prototype of the Concept Browser

Conzilla is a concept browser which is developed by Mattias Palmér and Mikael Nilsson under supervision of Ambjörn Næve at the KMR. The project originates from research done in the fields of knowledge management and structured information and which is referred to as the Knowledge Manifold.

A concept browser is a kind of knowledge-management tool in which information is presented and organized in a structured and coherent way using the principles described in the section 5.2.

While exploring a topic, Conzilla gives an overview of related concept(s) and concept-relations within the chosen context, efficiently separating content-components from their contexts. This feature is among the prime properties that distinguishes a concept browser, such as Conzilla, from other common knowledge management tools. Since each concept can be seen from different aspects or appear in various contexts, concepts are grouped into appropriate containers called context-maps. The technique not only enables context-dependent navigation through different concepts, it also allows users to view specific content-components, such as a HTML page, attached to a concept. Consequently, one can view and explore several concepts/concept-relations and even corresponding content-components\(^1\) without losing the contextual overview. This way, users are prevented from the confusion, or surfing-sickness, which often arises while searching (=surfing) the Web using common Web browsers.

Conzilla utilizes RDF to build the underlying information structure of the context maps while the visual presentation is carried out by ULM. Furthermore, all information contained in a context map is wrapped in RDF/XML encoding syntax, as it is described in Appendix A. This means that Conzilla not only supports the

\(^1\)This is done by presenting actual content in a separate displaying tools, e.g. the Netscape browser.
emerging Semantic Web technology but it also pushes the progress even further by supporting conceptual browsing.

Conzilla is also equipped with an editing tool for creating/modifying context maps. Creating a context map is very similar to the way we use graphical drawing programs, such as Dia\textsuperscript{2} or Visual Paradigm\textsuperscript{3}, to create UML-based class diagrams.

6.1 Demonstration

After this initial presentation of the theoretical structure behind a concept browser, we intend to demonstrate how the Conzilla browser works in action, illuminating the way all these ideas are applied.

The screen-shot in figure 6.1 illustrates the Conzilla browser displaying a context map\textsuperscript{4}.

![Context Map](image)

**Figure 6.1.** A context map where the concept Environment appears in the context of Mathematics.

The user can then select a concept and perform one of the following commands:

1. Right-click on the mouse and select one of the following entries from the pop-up menu (see figure 6.1):

\textsuperscript{2}\url{http://www.gnome.org/projects/dia/}
\textsuperscript{3}\url{www.visual-paradigm.com}
\textsuperscript{4}This is an applet version of Conzilla ("http://www.conzilla.org") which can be loaded on Web browsers using a java plug-in.
6.1. DEMONSTRATION

Surf: This command allows you to navigate into a contextual neighborhood of the chosen concept.

View: This command allows you to view all content material available within the chosen concept.

Info: This command presents meta-data information associated with the concept.

2. Double-click presents a detailed view—in form of a new context map—of the chosen concept.

Selecting surf → Mathematics exploratorium will lead us to the view illustrated in figure 6.2 which is a contextual neighborhood of the concept Environments.

![Context Map Illustration]

Figure 6.2. Viewing a context map in the Conzilla. The pop-up window beneath displays LOM meta-data information of the concept Mathematics

When the new map is loaded, the concept Environments will be highlighted indicating the user’s navigational path\(^5\). This is a useful feature which guides the user in his contextual orientation through context maps. If the view entry is chosen, a list of content-components will be displayed.

\(^5\)The highlighting disappears when the pointer is moved to another concept, in this case to the concept Mathematics.
CHAPTER 6. THE CONZILLA BROWSER: A FIRST PROTOTYPE OF THE CONCEPT BROWSER

The meta-data information related to the concept will be given by choosing the info entry from the pop-up menu, as illustrated in figure 6.2. The default meta-data used in this example confirms to the IEEE\textsuperscript{6} set of Learning Object Metadata (LOM) which is the emerging standard within the e-learning communities.

Double-clicking on the concept Environments will display the detailed map of this concept which in this case, i.e. within this particular context map, is the previously shown context map (see figure 6.1). Hovering above a concept will result in displaying a tooltip with a short description about the concept. Finally, if the user decides to view a certain content of a concept, (s)he can simply select one of the content items from the content menu bar which will be activated using the appropriate plug-in. This is, of course, conditioned to the existence of the necessary plug-ins and the platform for which the content is designed.

6.2 Feature summary

The list below summarizes some of the several advantages that Conzilla possesses.

- Clearly Separating content-components from the context.
- Helping the user to build an overall contextual understanding of the underlying concepts which makes it easier to study and integrate visually observed information.
- Supporting navigation through contextual neighborhoods.
- Allowing context-dependent aspect filtering of content-components which lets you discard the irrelevant information (avoiding surfing-sickness), see vi.

It is worth knowing that the Conzilla is a work-in progress. This means that different parts of the project are subject to modifications and improvements which is why we have been brief in describing certain features in detail. This includes the filter aspect which is not yet fully implemented.

\textsuperscript{6}http://www.ieee.org/
Chapter 7

Mobile Concept Browser Analysis

The task of developing a mobile concept browser is divided into client- and server side sub-projects with this Master’s project being responsible for the client-side solution. Henceforth, the browser application (or the client) will be referred to as the Mobile Concept Browser or MCB.

The analysis is divided into the following major parts:

- A feasibility study regarding mobile phones’ technical capabilities.
- Studying similar projects.
- The requirements.

7.1 Feasibility study

Hand-held digital devices, such as PDAs, in general and mobile phones in particular belong to the category of personal computers with limited resources some of which are [23]:

- Limited memory and processor capacity
- Battery power limitation
- Smaller display size and limited user input facilities

Additional constraints which arise due to the connectivity issues are:

- Limited Bandwidth
- Delaying problems in communication
- Uncertain connection stability

Considering the constraints mentioned above and in order to obtain an optimal user interaction with the system, we should strive to minimize:
The amount of transferred data: this will result in reduced loading- and pro-
cessing time on the device.

The computational operations on the client-side: the client should be relieved
of doing heavy computational operations, saving processor time and battery
power. Thus, for a given situation two scenarios are possible: either this
information is included in the buffer in advance, or the client will initiate an
additional server-connection, asking it to perform the task. This, of course, is
a question of optimization. Adding more information to the buffer will enlarge
the size of data, increasing loading- and processing time as result. On the other
hand, an additional connection means a longer delay in the user’s interaction
with the system.

The number of client-to-server connections.

The number of button-pushing in order to use most common functions.

And maximize:

The amount of information that can be retrieved based on data received on a
single connection: this allows the user to avoid unnecessary connections and
deal-time in-between as result.

7.2 Similar projects

Studing similar projects is often a good way of improving your knowledge about the
subject you are about to examine. For this reason, we studied two projects closely
which we considered to be relevant to our Master’s project.

7.2.1 WAP browsers using WML

The first micro browsers on wireless clients were developed using Wireless Application
Protocol technology or WAP\(^1\) which made creating Web applications possible
for mobile phones using WML (Wireless Markup Language) instead of HTML [23].
WAP is based on Internet standards, e.g. HTML, XML and TCP/IP. As matter a
fact, the WML is inherited from HTML, but it is packaged in XML as it is the case
with RDF or XHTML.

7.2.2 WebViewer using HTML

WebViewer is a new micro browser recently introduced to the market by Reqwireless\(^2\)
which enables Java-capable mobile phones or PDAs to view HTML based Web pages.
It is also possible to explore various kinds of content-components, such as PDF files,
GIF images, Word documents etc.

---

\(^1\)The WAP technology was initiated by the WAP Forum: http://www.wapforum.org/.
\(^2\)http://reqwireless.com/webviewer.html
7.3. THE REQUIREMENTS

WebViewer has several interesting features. Being a content oriented Web browser based on Sun’s J2ME platform, this product can be considered as a suitable complementary tool to our concept oriented Web browser.

7.3 The Requirements

The requirements are divided into four major parts: Functional Requirements, Communication requirements, General requirements and Semantic Web conformance.

7.3.1 Functional Requirements

The list below presents a short summary of the expected browser’s functional capabilities.

i. Enabling network connection.

ii. Presenting context-maps in which concepts and concept-relations are clearly illustrated according to the ULM specifications.

iii. Providing means for browsing (surfing) through contextual neighborhoods as well as the detailed map of a concept.

iv. Displaying meta-data information attached to a concept/concept-relation.

v. Exploring possibilities and, if feasible, providing means for exploring content materials. Naturally, creating facilities for displaying different types of content-components can, of obvious reasons, not be included within the boundaries of this project, except for the case where the content-component itself is a context-map. This is termed as contextualization of content-components (see chapter 5.2 role vi).

vi. Enabling navigation through a history list of explored context-maps.

vii. To be able to bookmark a context-map.

viii. Saving and reloading a context-map.

7.3.2 Communication requirements

Many of the demands stated above presuppose a working client-server communication setup. For this reason, an appropriate communication protocol will be needed.

7.3.3 General requirements

Mobile phones come in different varieties depending on the vendor producing the device. They adopt different technologies on their products leaving the application developers to come up with device specific solution for each product category.
CHAPTER 7. MOBILE CONCEPT BROWSER ANALYSIS

We intend to step over these obstacles and design an application which gains a vast spread among mobile phone users, regardless of the platform implemented on the device. In short, the browser application should be:

- Platform independent: runnable on different platforms.
- Easily extendable.

7.3.4 Semantic Web conformance: RDF based context maps

Context-maps designed for Conzilla are based on RDF/XML, and as such, the browser conforms to the emerging Semantic Web technology. This a design goal that we also wish to achieve. We do, however, realize that a similar approach, i.e. constructing RDF/XML based context-maps intended for mobile phones, will put a great amount of strain on the device. This is because of extensive structure of the RDF/XML based context-maps and the corresponding parser needed to interpret these documents. Nevertheless, we will take this matter under careful consideration during the application design phase, trying to find a suitable solution.
Chapter 8

Platform and Tools

An important part of this project has been the task of choosing suitable development platform. While evaluating possible candidates, the question that arose was: given a certain platform, how are we going to test this application. In other words, we will need a mobile phone emulator in which the browser application can be run and examined. Adding this extra condition to our requirements, we had, at the time, only two candidates available to choose from.

8.1 The Mophun

The Mophun was introduced by Synergenix\(^1\). It is an optimized runtime environment designed mainly for developing graphic-intensive applications, such as mobile games, using the C programming language. Unfortunately, we didn’t have the opportunity to verify this platform in detail, as we failed to install the emulator on our Linux machines.

8.2 The J2ME

Developed by Sun Microsystems\(^2\), J2ME is a java-based optimized runtime environment targeting a wide range of hand-held devices, such as pagers and mobile phones. The edition comes in a complete SDK\(^3\) containing a compiler, an emulator (Wireless Toolkit WTK) and an API. The J2ME is a layered structure consisting of configurations, profiles and optional packages. Configurations define a minimal set of class libraries and a virtual machine. Currently, there are two configurations: the Connected Limited Device Configuration (CLDC), and the Connected Device Configuration (CDC). The CDC targets hand-held devices with higher capacities, e.g. a PDA, whereas the CLDC, which is the smaller of the two configurations, targets devices, such as mobile phones, which typically have either 16- or 32-bit CPUs,

\(^1\)http://synergenix.com  
\(^2\)http://www.sun.com  
\(^3\)version WTK 2.0 Beta 2, CLDC 1.0.4 and MIDP 2.0
CHAPTER 8. PLATFORM AND TOOLS

and a minimum of 128 KB to 512 KB of memory available for the Java platform implementation and associated applications.

The configurations combined with a set of higher level APIs, or profiles, will help the application developers to access device specific properties. The MIDP or Mobile Information Device Profile is designed for mobile phones which combined with CLDC provides a complete runtime environment that leverages the capabilities of hand-held devices [27].

The J2ME platform maintains following advantages [8].

- Built-in consistency across products in terms of running on any device: this fulfils the requirement of being platform independent.

- Portability of the code: this means that the compiled code generated by the java compiler can run on any other machine that has a java runtime environment installed.

- Leveraging the same Java programming language: J2ME is upwardly scalable to work with the J2SE and J2EE platforms.

- Safe network delivery: our application is highly dependent on consistent and reliable network connectivity.

- Industry standard: java has gained great respect among a large number of software developers and members of the industry community.

- Object oriented programing language: an excellent tool for writing extendable applications.

In addition, we have the opportunity to leverage the knowledge gained from the Conzilla project, as it is also written in java.

As the project went by, we noticed several similar tools emerging into the market, among them a customized version of Sun’ J2ME from Sony Ericsson⁴, (version WTK 1.0.4_01, CLDC 1.0 and MIDP 1.0). This gave us the opportunity to run our application on a device specific emulator.

⁴http://developer.sonyericsson.com/site/global/docstools/java/p_java.jsp
Chapter 9

Design of the Mobile Concept Browser

While designing the structure of the browser, the ambition has been building a simple, optimized model which not only meets the device constraints, but also conforms to the design principles/requirements discussed earlier.

9.1 Definitions

In previous chapters we explained the theoretical background of conceptual browsing and how this was applied in the Conzilla browser. We also defined some elementary terms, e.g. concept, context-map etc., which we will here specify further in order to demonstrate what information they stand for in the context of the Mobile Concept Browser (MCB).

9.1.1 Concept

In MCB a concept is represented by the following properties:

- A Title: a string representation.
- An URL: an unique identifier of a concept.
- A detail-map URL: this is used for a deeper study of the concept.
- A description: a short note (=meta-data) describing the concept.
- Contextual-Neighborhoods: a list of context-maps in which the concept appears. The list include both the title and the URL of the context-maps.
- Content-components: a list including title, mime-type and URL of content related to the concept.
- General visual attributes: position, size and the graphical shape of the concept.
CHAPTER 9. DESIGN OF THE MOBILE CONCEPT BROWSER

9.1.2 Concept-relation

A concept-relation, or more generally speaking, an association, describes how concepts are related to each other. Per definition, a concept-relation is a kind of concept, and as such, it inherits the properties of a concept. In addition, a concept-relation also has its own special characteristics as described below.

- The type of (ULM-based) association: The design should support general types of associations as described in chapter 5.3, most importantly the types: generalization, aggregation and classification.

9.1.3 Context-map

In MCB, a context-map can be seen as a container, holding all necessary information regarding its main components, i.e. concepts and concept-relations, as defined above, but it also contains context-map-specific information consisting of:

- An URL: which uniquely identifies the context-map.

- A Title: the title of the context-map.

- Map position: the relative position of the extracted map used for rendering purposes.

- Number of concepts: this information is added only to reduce the amount of computational operations on the client side. Technically, one can also retrieve this information from the data buffer simply by counting the number of the concepts.

- Number of concept-relations: the same reasoning as above.
9.2. GENERAL DESIGN OVERVIEW

9.2 General design overview

Although mobile phones are gradually getting more powerful, many applications are still too resource-intensive for most of these devices. To overcome this obstacle we decided to set up a portal server to which the mobile phone (the browser application) can delegate complex tasks. Figure 9.1 illustrates a general overview of the system’s structure.

![Diagram](image)

*Figure 9.1. The General network structure. MCBP stands for the Mobile Concept Browser Protocol.*

As we can see from figure 9.1, the server’s responsibility extends from locating and gathering RDF/XML based context-maps to preprocessing and reformatting them for presentational purposes which is finally carried out by the client. This also illustrates the browser’s coupling to the Semantic Web which has been achieved through the parallelly developed proxy server. For a thorough explanation, the reader is encouraged to study the server-side report, Server-side-solution for Conceptual browsing, written by Fredrik Enoksson [9].

9.3 The Communication

The browser application will extend J2ME’s network connectivity support for remote communications. More specifically, the application will use the HttpConnection class in order to maintain application portability among different clients because this class is mandatory for all java-enabled devices [26].

9.3.1 The request-response protocols

As described earlier, establishing communication between the client and the server requires a predefined communication setup. This set of request-response protocols is referred as the Mobile Concept Browser Protocol (MCBP), a complete reference of which is given in the Appendix B.
9.3.2 The request

On the client side, the request part of the MCBP (wrapped in HTTP) consists of following components:

- The URL of the requested context map.
- A collection of user specific data, or user’s profile, which enables the server to deliver tailored context-maps, responding to the user’s specification. Currently, this specification includes:
  - Language: the preferred language used in context-maps. There is a default fall-back.
  - Presentation: preferred presentational options. The idea is to let the user choose among different presentational possibilities. Currently, our mission is to deliver a graph-based context-map, but in a future extension, however, one might prefer a voice-based context-map, or even both.
  - Display size: of the device.
  - Version: Version of the browser application in use.

9.3.3 The response

The parallelly developed server [9] is responsible for locating, assembling and delivering requested context-maps. The response part of the MCBP (see Appendix B) is mainly composed of:

- Connection message: a server message informing the client about (un)successful localization of the requested context-map.
- Version: the version of the protocol in use.
- context-maps: consisting of concepts, concept-relations and context-map specific data.

9.3.4 Wrapping context-maps: Data interoperability

As mentioned before, a context map designed for Conzilla contains a considerable amount of data. This is partly due to the employment of the RDF/XML encoding technique which gives the context-maps a rigid, but rather extensive, structure. In our design, however, we have chosen a different approach. Instead of a direct employment of RDF/XML on the client side, we decided to let the server function as the bridge which will maintain data interoperability between the client and other network parties. The encoding syntax used for serializing the context-maps is rather simple and is described in Appendix B.
Chapter 10

The Application Structure
and the Conceptual Data Model

Based on the analysis performed in the previous sections, we will present and discuss the structure of a conceptual data model which is designed to capture and embody the principles of conceptual browsing and the functional requirements presented earlier. The UML-based class diagram in figure 10.1, describes our approach in structuring this model, and as it reveals, the data model is divided into three major parts: The conceptual browsing, The user interface and The communication.

- The conceptual browsing: Being at the heart of the browser application, this is where the designed context-maps with its concepts/concept-relations are actually built. In this model, the entities context-map, concept and concept-relation are considered as Resources\(^1\). Furthermore, each concept/concept-relation might contain sets of content-components. Note, that a concept, as defined in chapter 5, is thought as a free entity, i.e. independent from any contextual aspect. However, this feature is not entirely reflected in our designed data model, as the concepts and concept-relations can only exist within a given context-map.

- The user interface: This part is responsible for rendering graphics, including created context-maps. It will also handle user’s interaction with the application. The graphics package of J2ME provides simple 2D geometric rendering capabilities. Drawing primitives are provided for text, images, lines, rectangles etc.

  From the user’s perspective, the primary objective is to get an overall understanding of the displayed context-maps. Knowing that context-maps can appear in different sizes and forms, some basic features, e.g. zooming, panning and mouse pointer, are added into the application to improve the user’s accessibility to the maps. A more detailed description about these features is given on chapter 11.

\(^1\)See the definition of the term Resource on chapter 5.
The J2ME-2.0 provides limited support for exploring certain types of content-components designed for mobile phones, including a Multimedia API which enables devices to: play sound, view images\(^2\) and video sequences\(^3\). Naturally, this will not remedy the problem of exploring all sorts of content materials but, as stated before, it is not our intention to equip the browser with various kinds of content viewers. Our main ambition is to create a suitable interface towards possible content-oriented applications. We have, however, conducted a simple test in which the user is able to view a video sequence, just to verify the browser's functionality.

Today's Web browsers support a variety of online documents, such as PDF-files and java-scripts, through appropriate plug-ins. Conzilla, for instance, extends this capability by utilizing existing Web browsers, such as Netscape, in order to display online content-components. In a similar fashion, the MCB browser could make use of a thin content browser, e.g. WebViewer (see chapter 7.2.2) to perform similar tasks.

- The communication and data buffering: Fetching context-maps is done by sending a HTTP-request to the proxy server which locates, packages\(^4\) and returns the requested context-map back to the client. The browser application will then re-assemble the context-map by parsing the received data buffer and, finally, render it on the display.

10.1 The Java classes

The three major parts introduced in our data model are directly represented in the java classes described below (see also figure 10.1). The total size of the application is about 22 KB, which is an acceptable size considering the available resources.

10.1.1 Conceptual browsing: The abstract layer

Resource: an interface representing Resources, including context-map, concepts/concept relation and content-components.

Concept: an abstract super-class representing our definition of concept/concept-relations.

ConceptBox: this class is a realization of the predefined concept.

Relation: this class is a realization of the predefined concept-relation.

ContextMap: this class is a realization of the predefined context-map.

\(^2\)PNG format, as specified by the PNG Specification, Version 1.0.
\(^3\)Supported video formats are: WAV and MPG.
\(^4\)As defined in MCBP.
10.1. THE JAVA CLASSES

Figure 10.1. The java class hierarchy.

10.1.2 Graphics

Map: this class handles graphics, including rendering context maps and the mouse pointer. It also carries out user commands by implementing the java class CommandListener.

Forms: the class Forms provides facilities of various kinds, such as read-only text fields, editable text fields, editable date fields and choice groups. Like the Map class it also implements the CommandListener class to perform user commands.

Content: this class is generally intended to be the browser application’s interface towards diverse content-components. By using the mime-type, it is the implementation duty to decide which application or plug-in should be initiated in order to display the content-component. This class is also responsible for contextualizing (see role vii in chapter 5.2) the content-component, i.e. if a content-component is a context-map.
10.1.3 Communication

HttpConn: this class maintains network connectivity support.

Parser: this class is responsible for parsing the stored data buffer as well as structuring the context-maps.

10.1.4 Miscellaneous classes

Initiate: The main class responsible for initiating the application.
Chapter 11

The Browser

11.1 Introduction

In this section we intend to describe the overall structure and the functional properties of the designed browser. In brief, we have developed a concept browser, referred to as Mobile Concept Browser (MCB), which enables downloading and displaying context-maps, through a proxy server\(^1\). The concepts/concept-relations contained in a context-map are hyperlink-enabled, giving the user the possibility to explore the concepts further. It is also possible to view certain types of content-components, for instance a textual description, a video sequence or a PNG-image.

11.2 User interaction

The user’s interaction with the application happens through the keyboard and since devices available on the market come in various forms and with different keyboard formats, we need to include additional demands on keyboards. For this reason, and because we aim to design an application which functions properly on a broad number of devices, the keyboards should have the following characteristics:

- Containing the keys 1–9.
- At least one soft key which is usually placed right under the display window. This key, even named as Option or Menu key, is used to offer the user a choice menu.
- The up and down arrow keys. These keys make selection from the choice menu possible.
- An Enter key with which the user can confirm and execute a selection.

\(^1\)See Server-side-solution for Conceptual browsing, written by Fredrik Enoksson [9]
11.3 Starting the application

As mentioned earlier, the application is developed using Sun’s emulator kit, WTK2.0\textsuperscript{2}. On the lines below, we assume that the reader has basic knowledge of using the emulator.

Start the emulator and open the browser project. Compile the code and then press the Run button to launch the application. A mobile phone emulator will appear (see figure 11.1) displaying the title of the project. Now, you can start the application by pushing the Launch command button from the menu bar. The browser will now start to operate asking the user to enter the URL (of a context-map).

The user can then send the request to the server by selecting Go from the Menu and then pushing the Enter button. In the upcoming texts, we will assume that the user always pushes the Enter button after selecting an item from the Menu list or whenever it involves selections of any kind.

![Image of Sun's WTK2.0 emulator](image)

Figure 11.1. The Sun's WTK2.0 emulator.

Assuming that nothing goes wrong, the browser will render the context map on the display (see figure 11.2).

\textsuperscript{2}http://java.sun.com/j2me/docs/wtk2.0/
11.4 Graphical user interface and functionalities

Generally, the browser can be in two states, the map-mode or the form-mode. The browser is in the map-mode if it displays a context-map, and in the form-mode in all other cases. Below, we describe the created user interfaces and the functional properties of the browser. The notations seen inside the brackets [ ] refer to the Design Principles (DP) or Requirements (Req.) announced earlier in this report.

- Meta-data description [DP sec. 5.2 role v]: as described previously, each concept might contain a short textual description (=meta-data). There are two different ways of viewing this information.
CHAPTER 11. THE BROWSER

- Hovering above the concept with the mouse pointer will trigger a pop-up window event, displaying the corresponding concept description. The lifetime of the pop-up window is limited by the application. Note that the browser will stay in the same mode, i.e. the map-mode, before and after that the event has been triggered. This functionality, i.e. pop-up window event, is not enabled for concept-relations.

- Select a concept from the Menu. A textual description will then appear on the display. This action will change the browser state from map-mode into form-mode as opposed to the previous choice.

- Navigation through contextual neighborhoods [DP sec. 5.2 role iii]: in order to navigate through contextual neighborhoods of a concept, the user can select a concept from the Menu, whereby, the browser will display the meta-data description. Selecting the Menu button again and choosing the Surf entry, will list a choice group of all available contextual neighborhoods which can be explored one at the time. This action requires network connection.

- Exploring a detailed map [DP sec. 5.2 role ii]: there are two ways of exploring a detailed map. The first approach uses the mouse pointer: hovering the pointer above a concept and then pushing the key number five will result in displaying the detailed map of the chosen concept. The second method is carried out in the same way as for exploring contextual neighborhoods, i.e. while the browser is showing the meta-data description, the user can select the Detailed map entry. Note, that the mouse functionality is only used to explore detailed maps and to view meta-data description. Exploring Detailed maps requires network connection.

- Viewing content-components [DP sec. 5.2 role i,ii and Req. sec. 7.3.1 role v]: again, as described for exploring contextual neighborhoods, select a concept from the menu and then choose the contents entry which will list belonging content-components. This action will require network connection.

- Back [Req. sec. 7.3.1 role vi]: traversing backwards through the navigation history is enabled through this function. Generally, using this function will result in displaying the previous screen but the Back button has a dual functionality, meaning that its performance varies depending on the browser's actual state. The application follows a certain logic in which the browser's map-mode is given higher priority than the form-mode. This means that pushing the Back button, while in the map-mode, the browser will remain in the same mode regardless of the previous browser-state, i.e. it will only display the last context-map prior to the present one. On the other hand, if the browser is in the form-mode, pushing the Back button will change the browser-state to the prior one. This logical priority is illustrated in the drawings in figure 11.3.

38
11.4. GRAPHICAL USER INTERFACE AND FUNCTIONALITIES

![Diagram](image)

- **Forward** [Req. sec. 7.3.1 role vi]: this button is used to move forward through the history list and it functions using the same logical principles as described for the Back button.

- **Panning**: a context-map might exceed the limits of the displaying area in which case only a part of the map will be visible to the user. Using the arrow keys Up, Down, Right, Left the user is able to pan across the whole map in either of four directions.

- **Zooming**: this feature enables users to zoom into- and out of a context-map. The application limits the zooming action to some useful degree.

- **Mouse Pointer**: the mouse pointer functions through keypads 1–9 and is used to trigger several events, some of which have been described earlier. In addition, the mouse pointer can trigger a panning-event whenever it points at the boundaries of the display window.

- **Save file** [DP sec. 5.2 role viii]: this function is used to temporarily save a context map which is being displayed. To save a file, select Save file from the Menu. After saving a file, an message window will be displayed confirming the action. The files are stored temporarily which means that they exist only during the application life cycle. No persistent storage capability has yet been implemented.

- **Open file** [DP sec. 5.2 role viii]: opens a stored file by choosing Open file... from the Menu.

- **Bookmark map** [DP sec. 5.2 role viii]: while in the map-mode users are able to temporarily bookmark the displayed context-map. After a book-marking event, a message window will confirm the action.

39
CHAPTER 11. THE BROWSER

- Bookmarks [DP sec. 5.2 role viii]: this function enables users to load an existing bookmarked context-map. This will require network connection.

- Alerts: an Alert is a temporary screen used to present messages to the user. The intended use of Alerts is to inform users about possible errors and also to confirm certain user interactions.
11.5. FLOWCHART: THE BROWSER’S POSSIBLE STATES

11.5 Flowchart: The browser’s possible states

The browser’s possible states and the state transition methods are summarized in figure 11.4.

Figure 11.4. A state diagram of the application. The notation \( \rightarrow (x) \) indicates next transition mode.

41
11.6 Real test

The browser application was implemented and tested on an Ericsson mobile phone Z1010 by Richard Wessblad from DatorDoktor⁷ with satisfactory results.

11.7 Further improvements

The browser application should be considered as a first prototype which needs to be reviewed and improved (as well as debugged) in many of its aspects, including:

- The communication: This part needs to be more steady and reliable, as we often faced problems while trying to maintain connection between the client and the server.

- The mouse pointer: There are bugs in the pointer’s functionality which is related to panning and pop-up windows. In some occasions the mouse triggers a panning event although it does not point at any of the edges of the display.

---

⁷http://www.datadoc.se
Chapter 12

Project Summary and Conclusions

The KMR group is actively participating in several projects involving research in knowledge management. The vision of the Web-of-concepts is among one of their activity areas which is yet to become a reality. Meanwhile, they continue to develop tools suitable for these purposes, including this light version of a concept browser (MCB).

The overall aim of this project has been enabling conceptual browsing on thin wireless clients, such as mobile phones. A concept browser is meant to be used in the world of Conceptual Web where concepts are free entities. A set of concepts, in a given perspective or context, form a context-map. The main purpose of a concept browser is to provide a suitable environment for organizing and exploring information based on context-maps.

The main project has been divided into two separate sub-projects which is referred as the client- and the server-side projects with this Master’s project being responsible for the client-side solution. Together, we have managed to design and test a browser application in an emulator environment, as well as on a real mobile phone. The browser supports the design principles i, ii, iii, iv, v and vii, all specified in chapter 5.2. Point viii is partly supported, as the implementation only allows to bookmark the whole context-map and not a particular concept by itself. There is no real support for principle vi, i.e. meta-data based filtering, although the client request contains presentational options, such as a preferred language. The browser application supports the Semantic Web through the parallelly developed proxy server which uses the Conzilla interface for gathering RDF-based context-maps.

12.1 Future work

During the course of this project many ideas, regarding the future development of this application, have emerged. These ideas are outlined on the lines below.

- Structured data: currently, a simple set of data exchange protocols is used for client-server communication. A future, full-fledged version of this proto-
CHAPTER 12. PROJECT SUMMARY AND CONCLUSIONS

col should follow the W3C\textsuperscript{1} recommendation regarding data structuring as announced by Semantic Web activity group. A first step towards this goal would be a direct employment of XML encoding syntax in the protocols. Currently, there are a handful of open source XML parsers available, designed for hand-held devices. Among them is kXML 2.0 alpha, created by Enhydra\textsuperscript{2}.

- Further optimization of request-response protocols: a suggestion is to replace long URI strings, contained in context-map, with, for instance, integers, relieving the client from dealing with large text masses as well as reducing memory footprint. This can be done by letting the server keep track of all URI-s in a database.

- Segmentation of large maps: some context-maps may contain a large number of concepts and concept-relations, difficult or even impractical for a thin client to display. In such cases, segmentation of the context-maps by the server into appropriate parts would be a possible way of overcoming this hindrance.

- The graphical user interface: considering the small display size of mobile phones, and since the MCB application leans heavily on graphical representation technique, further improvements of user interfaces are strongly suggested. This includes improvements in the selection process of the concept being explored.

- Concept representation: another suggestion considers the graphical form by which concepts are represented. Currently, the application uses a rectangle and a descriptive title to illustrate a concept. We believe that the technique of concept-representation needs to be more general and not limited by a few predefined shapes.

\footnotesize
\textsuperscript{1}World Wide Web Consortium, http://www.w3.org
\textsuperscript{2}http://kxml.enhydra.org/
Bibliography


BIBLIOGRAPHY


[26] Network Programming with J2ME Wireless Devices,
http://www.wirelessdevnet.com/channels/java/features/j2me_http.phtml,
last visited November 2005.

[27] White Paper The K virtual machine (KVM),
last visited November 2005.
Appendix A

RDF

RDF extends XML model- and syntax specification for describing resources in a machine understandable way. The RDF statements shown in figure 3.4 in chapter 3 can be encoded in XML syntax as demonstrated below.

```xml
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns"$
xmlns:ex="http://example.org/elements/"$
<rdf:Description rdf:about="http://example.org/doc/report.html">
  <ex:writer>Noel Zaragrian</ex:writer>
  <ex:date>2004-02-20</ex:date>
  <ex:subject>Describes process for planting</ex:subject>
</rdf:Description>
</rdf:RDF>
```

The first line of the XML-RDF encoding above indicates that this is an XML document. The xmlns attribute on the second and third lines indicates two name spaces\(^1\), i.e. rdf and ex, with rdf being the default name space. All the properties described between the `<Description>` tags will refer to one of these name spaces. The `about` attribute on `rdf:Description rdf:about=...` line indicates the resource `report.html` which is to be described by three properties `ex:writer`, `ex:subject` and `ex:date`.

---

\(^1\)Name spaces are used in XML in order to avoid name conflicts [14].
APPENDIX A. RDF

Now, let us continue to expand the previous example even further.

![Diagram of RDF concepts](image)

Figure A.1. Several statements describing the same resource.

In figure A.1, further information about the creator is added. At the same time the resource or the person Noel Zargarian is now uniquely identified as “http://example.org/staff/staffID\#25566”.

A.1 RDF structure

The Resource Description Framework is a model that can be utilized by anyone in order to describe resources and resource relationships, using named properties and property values.

RDF itself does not provide any mechanisms for describing properties, or the relationships between properties and resources. That is a task for the RDF vocabulary description language, RDF Schema, which defines a set of classes and properties that can be used to describe other classes and properties [20].

RDF and RDF Schema define a set of core resources, such as: rdf:Type, rdfs:Property, rdfs:Class, rdfs:Resource and rdfs:Literal, which together build the basics of further extensions [19], analogous to the way object oriented applications are built upon already existing classes. The class rdfs:Resource is defined as the superclass of all classes (similar to the class Object in Java). Consequently, the set of all resources of a specific class is always a subclass of rdfs:Resource. The abbreviation rdf: and rdfs: denote the URI-references: “http://www.w3.org/1999/02/22-rdf-syntax-ns\#” and “http://www.w3.org/2000/01/rdf-schema\#”, respectively [19].
A.1. RDF STRUCTURE

A.1.1 RDF class and property

In RDF Schema, resources are divided into groups of RDF-classes and properties. In RDF Schema, classes may be described using resources rdfs:Class, rdfs:Resource and the properties rdf:type and rdfs:subClassOf where as properties may be described by RDF class rdf:Property and the RDF Schema properties rdfs:domain, rdfs:range, and rdfs:subPropertyOf. The RDF property rdf:type can be used to indicate that a resource is an instance of a class. The rdfs:range property is used to indicate that the values of a particular property are instances of a designated class, whereas rdfs:domain property indicates that a particular property applies to a designated class [22]. An example will perhaps help to clarify the statements above.

Imagine that in our previous example, figure A.1, the document was written in a collaboration of several participants and that Noel Zargarian was assigned the job, technical writer. How we can build our own schema on top of the RDF Schema is illustrated in figure A.2. In addition, it also shows how the properties rdfs:range and rdfs:domain can be used.

![Diagram](image)

Figure A.2. Extending RDF classes and properties using RDF Schema.
A.2 Ontologies

The RDF Schema does not specify any vocabulary of application-specific classes. The definition of different properties or vocabularies are performed by so called meta-data description communities, such as Dublin Core Meta-data Initiative\(^2\) (DC) or IEEE-LOM\(^3\). Each community defines elements (vocabularies) concerning its own knowledge domain which is called an Ontology. The term Ontology can be described as a formal specification of the concepts and relations of some domain. The DC working group are engaged in defining vocabularies that exist within the world of libraries (similar to a library card), e.g. creator or publisher\(^4\). Others, such as the LOM group are involved in describing meta-data concerning learning environments.

Another interesting ontology is Web Ontology Language (OWL), initiated by the W3C. The OWL specifies a Web-specific schema, i.e. it describes the semantics of classes and properties used in Web documents [15]. Of course, the OWL extends RDF Schema as any other schema based on RDF. The OWL is also one of the basic elements of the Semantic Web [16] giving applications the structure necessary for processing the content of documents instead of just presenting them to humans.

\(^2\)http://dublincore.org/
\(^3\)http://www.iee.org/
\(^4\)http://dublincore.org/documents/2002/10/06/current-elements/
Appendix B

The Protocol

In this chapter we will describe the designed client-server protocol, version 0.1 of MCBP, in detail.

B.1 The Request

The request is composed of following parts:

- Request = Head, Language, Display_size, Presentational_Options;
  - Head = Transmission_Protocol, Submission_method, Target, Browser_Version;
    * Transmission_Protocol = “HTTP”;
    * Submission_method = “Post”
    * (* The URL of the requested map *) Target = Terminal-string;
    * (* Version of the browser*) Version = Browser_name, Version_nr;
      - Browser_name = “MCB”, ’/’;
      - Version_nr = Integer, Line Feed;
    (* The <Language> defines preferred language option. The default setting is Swedish (se) *)
  - Language = [Terminal-string], “;”;
  - (* The display size in pixels, (Width, Height). *) Display_size = (Integer, ’,’ Integer, ”’); (* Preferred presentational options. (the default setting is Context_Map *)
  - Presentation = [Terminal-string], ”’Line Feed, Line Feed;
B.2 The response protocol

The response from the server is composed of: Response = Head, Context_map

- Head = Lines, Connection_message, Protocol_version;
  - (* Number of lines in the buffer *) Lines = Integer, ';', Line Feed;
  - (* Server message. OK = 1, Error = 0 *) Connection_message = Integer, ';', Line Feed;
    * Protocol_name = "MCBP", '/';
    * Protocol_version = Integer, Line Feed;

- Context_map = Head, Concepts, Concept_relations;
  - (* Context-map specific data *) Head = Map_title, Map_URL, Map_position, Concepts_nr, Concept_relations_nr;
    * Map_title = Terminal-string;
    * Map_URL = Terminal-string;
    * (*The relative position of the context-map (x,y) where x and y reveal the map’s lower right-hand corner *) Map_position = Integer, ',', Integer, '', ';
    * (* Number of concepts *) Concepts_nr = Integer, ';';
    * Concept_relations_nr = Integer, ';';

- Concepts = Concept_URL, Title, ConceptBox_position, Title_position, [Contextual_neighborhoods], [Detail_map_URL], [Concept_description], Graphical_shape, [Content_components], Line Feed;
  * Concept_URL = Terminal-string
  * Title = Terminal-string
  * (*)(Start_x, Start_y, Width, Height) ConceptBox_position = (Integer, ',', Integer, ',', Integer, ',', Integer, ',', ',');
  * (*)(Start_x, Start_y, Anchor) Title_position = (Integer, ',', Integer, Integer, Integer, '', ');
  * Contextual_neighborhoods = Context_nr, [Title, URL];
    - (* Number of the contextual neighborhoods *) Context_nr = Integer, ',
      - Title = Terminal-string, ',';
      - URL = Terminal-string, ',', '$', ',';
  * Detail_map_URL = Terminal-string, ',';
  * Concept_description = Terminal-string, ',';
  * Graphical_shape = Terminal-string, ',';
B.2. THE RESPONSE PROTOCOL

* Content-components = Content_nr, [Title, Mime_type, URL];
  · (* Number of the content-components *) Content_nr = Integer,‘;’;
  · Title = Terminal-string, ‘;’;
  · Mime_type = Terminal-string, ‘;’;
  · URL = Terminal-string, ‘;’, ‘$’, ‘;’ Line Feed;
- Concept_relations = Relation_URL, Title, Line_sequences, Line_position,
  Title_position, Connected_concepts, [Detail_map_URL], [Description],
  ULM_symbol, Symbol_start_position, StrokeStyle, Line Feed;
  · (* The number of connected lines which represent a concept-relation *) Line_nr = Integer, ‘;’;
  · (* A connected sequence of lines where the first line’s end position
    is taken as start point for the next line, (Start_x, Start_y, End_x,
    End_y). *) Line_position = (Integer, ‘;’, Integer, ‘;’, Integer, ‘;’,
    Integer), ‘;’, ‘;’;
  · (* Position of title (x,y,anchor) *) Title_position = Integer, ‘;’,
    Integer, ‘;’, Integer, ‘;’, ‘;’;
  · (* <Connected_concepts> reveals which two concepts are connected
    by this concept-relation (Concept_1, Concept_2) *) Connected_concepts
    = (Integer, ‘;’, Integer), ‘;’, ‘;’;
  · Detail_map_URL = Terminal-string, ‘;’;
  · Description = Terminal-string, ‘;’;
  · ULM_symbol = ULM_type, Symbol_direction;
    · (* <ULM_type> The type of ULM-based association: ‘0’ =
      Association, 1 = Aggregation, 2 = Generalisation, 3 = Classification,
      4 = Example_of *) ULM_type = [‘0’ | ‘1’ | ‘2’ | ‘3’ | ‘4’];
    · Symbol_direction = (Integer, ‘;’, Integer), ‘;’, ‘;’;
  · Symbol_start_position = (Integer, Integer), ‘;’, ‘;’;
  · (* The stroke style of the line: 0 = solid, 1 = dotted *) StrokeStyle
    = [‘0’ | ‘1’], ‘;’