Using Metadata and Context Information in Sharing Personal Content of Mobile Users

Panu Vartiainen

Helsinki, 27.2.2003

Master's Thesis

UNIVERSITY OF HELSINKI

Department of Computer Science

HELSINGIN YLIOPISTO – HELSINGFORS UNIVERSITET – UNIVERSITY OF HELSINKI

Tiedekunta/Osasto – Fakultet / Sektion – Faculty Faculty of Science	Laitos – Institution – Department Department of Computer Science

Tekijä – Författare – Author

Panu Vartiainen

Työn nimi – Arbetets title – Title

Using Metadata and Context Information in Sharing Personal Content of Mobile Users

Oppiaine – Läroämne

Computer Science

Työn laji – Arbetets art – Level Master's Thesis	Aika – Datum – Month and year 2/2003	Sivumäärä – Sidoantal – Number of pages 66+1

Tiivistelmä – Referat – Abstract

The thesis discusses possibilities for using metadata and context information in annotating, sharing, and searching user-created content in the mobile domain. The first part of the thesis discusses metadata, ontologies, context information, and imaging. The latter part of the thesis describes a prototype system for classifying and annotating digital photographs and storing context information as metadata of the photographs in a mobile phone. Another role of the prototype system is to perform context- and ontology-based information retrieval using a mobile phone user interface.

The prototype system contains a limited RDF metadata engine and an ontology browser for mobile phones, as well as a server-side metadata and content repository. The implementation demonstrates that a part of the creation-time context, such as the location and temporal context, can be automatically gathered in a mobile phone, and stored as metadata for the content. In addition, the same parts of context information can be used for searching. The content and the metadata can be stored on a server and shared with other users. The prototype is built around a tourism scenario that works as an example of how these technologies can be used in a mobile phone.

Classifications (ACM Computer Classification System 1998):
H.3.5 [Information storage and retrieval]: Online information services – Data sharing, Web-based services; H.5.2 [Information interfaces and presentation]: User Interfaces – Graphical user interfaces, Input devices and strategies, Interaction styles; H.2.8 [Database management]: Database applications – Image databases, Spatial databases and GIS; I.2.4 [Artificial intelligence]: Knowledge representation formalisms and methods – Semantic networks

Avainsanat – Nyckelord – Keywords

Image classification, metadata, ontology, context information, mobile user interfaces, mobile imaging

Säilytyspaikka – Förvaringställe – Where deposited

Library of Department of Computer Science, serial number C-2003-

Muita tietoja – Övriga uppgifter – Additional information

This thesis was written under the commission of Nokia Research Center.
The thesis was declared secret until 1st April 2003.

Abbreviations

API **– Application Programming Interface**

DIG **– Digital Imaging Group**

DAML+OIL **– DARPA Agent Markup Language + Ontology Inference Layer**

DBMS **– Database Management System**

GIS **– Geographic Information System**

GPRS **– General Packet Radio Service**

GSM **– Global System for Mobile Communication**

J2ME **– Java 2 Micro Edition**

J2SE **– Java 2 Standard Edition**

JDBC **– Java Database Connectivity**

JPEG **– Joint Photographic Experts Group**

JSR **– Java Specification Request**

LIF **– Location Interoperability Forum**

MIDP – **Mobile Information Device Profile**

MLP **– Mobile Location Protocol**

MMS **– Multimedia Messaging Service**

OWL **– Web Ontology Language**

PNG **– Portable Network Graphics**

RDF **– Resource Description Framework**

RDFS **– RDF Schema**

SIR – **Semantic Information Router**

SMS **– Short Message Service**

URI **– Uniform Resource Indicator**

W3C **– World Wide Web Consortium**

WAP **– Wireless Application Protocol**

WGS **– World Geodetic System**

XMP **– Extensible Metadata Platform**

XTM **– XML Topic Maps**

Contents

1 Introduction

The recent mobile phones with color screens, cameras, and multimedia messaging capabilities open up new possibilities for user-created content. With a camera phone, it is possible to take a photo and send it to a friend in a multimedia message. However, if the user wants to store a lot of photos and share them with a bigger group of other users, it is necessary to annotate and categorize the photos to be able to find them efficiently.

Photos can be automatically categorized to high-level categories, like “city” and “landscape” or different forms of landscapes, according to their visual content with an accuracy of up to 90–95% per categorization level [VFJ01]. To perform more accurate categorization, manual annotation is needed. When searching photographs, the context of the photo, which is not always seen in the picture, can be an interesting search criterion. Additional metadata is needed for making computerized search possible.

Mobile devices have their own limitations in browsing and entering information. These include smaller screen size and slower text input than in desktop systems. On the other hand, pieces of context information, such as location, can be used in mobile phones and networks. New browsing and annotation methods that are designed especially for mobile phones are needed.

1.1 Scope of the thesis

This thesis is based on the following research problems:

1. Find out some scenarios and use cases for using metadata in sharing user-created content in the mobile domain

    1.1. How much and what kind of metadata can be generated automatically?

    1.2. Which simple methods of user input can be used in annotation?

    1.3. How can the user be motivated for annotating his images?

2. Evaluate potential technologies for sharing personal content of mobile users

    2.1. How do the existing metadata and ontology technologies work in the mobile environment?

3. Design and implement a prototype of using metadata for a selected scenario: tourism

    3.1. Design and implement a user interface for annotating images with metadata

    3.2. Design and implement a user interface for the information retrieval process using metadata

    3.3. Evaluate advantages and disadvantages of using metadata and ontologies in the selected scenario

    3.4. Find out possible and suitable formats of metadata in the mobile domain

Two scenarios are presented later in this section. Methods for automatic generation, manual creation and management of metadata are discussed later in this thesis. In order to find feasible solutions to the research problems, I have designed and implemented a prototype of an image sharing system with a user interface for mobile phones. In the prototype, tourism is used as an example use case, in the spirit of city highlights in the Virtual Tourist service[1]. The prototype system gathers context information (especially location) in the situation of taking the photograph, lets the user annotate and semantically classify the image in the phone, and stores the collected information as the metadata for the image. Moreover, in the search process the current context is used together with the user-selectable classification for searching photographs.

The prototype includes a graph-based ontology browser and editor for mobile phones. It can be used for annotating the content, extending ontologies, and performing ontology-based search. The mobile phone client uses a limited, lightweight RDF metadata engine that was created as a part of the system. The content and the metadata are stored in a content and metadata repository server.

1.2 Scenarios

Here are the two main scenarios that were chosen to be on the background of the prototype application design. The tourism scenario is fully supported in the prototype, except for sending the multimedia message, which is a basic functionality of the phone. The event notifications and user profiles that are used in the Hobby peer group scenario were not implemented in the prototype.

1.2.1 Tourism

Peter, a student who is spending a week of his vacation in Germany, is walking on a marketplace in Berlin. It’s half past nine in the morning of a quiet Tuesday in November. The marketplace is quite empty. Peter is wondering what this marketplace looks like on some other time of the year and what happens there.

[1] See http://www.virtualtourist.com/

Peter starts the Travel Browser on his mobile phone. The application connects to a travel portal. The service performs a query according to Peter's current location. Peter can see photos of the festival that takes place each September in the city. He picks one of the photos, and sends it in a multimedia message (MMS) to his friend.

After a while, Peter feels like a lunch. He opens the Travel Browser, selects the category "Restaurants" and gets a list of restaurants nearby, sorted by distance. To get more precise results, Peter selects "Restaurants → Fish" from the browser, and gets a list of fish restaurants. Peter selects a restaurant that looks nice in the photo and is described to have reasonable prices. The system shows a map with Peter's current location and the location of the restaurant.

In the afternoon, Peter finds a nice café and wants to share the information about it on the service. He takes a photo, and categorizes the photo as "Restaurants → Cafeterias → Modern". He types in the name of the café and describes it briefly using the keypad. The image is submitted to the server together with the metadata including the current location and the information Peter just entered. Other people can now find the cafeteria from the service.

1.2.2 Hobby peer group

Jack, 17, has been doing judo for four years. He is going to attend a judo competition in Bradford and wants to share the photos he takes at the tournament with the local judo club.

Jane, a judo coach from Bradford, is one of the editors of a judo ontology. She has added "Bradford Judo Competition" as a subclass to "Judo → Competitions" using the ontology editor. She has also created a time, location, and user profile based event notification of the judo competition.

Jack travels to Bradford. Since he has claimed judo as a hobby in his personal profile, he gets an advertisement of the competition when he arrives to the city. The advertisement message contains a link to the home page of the competition. Jack can see the list of competing pairs on the competition web site.

Jack adds descriptions to the photos he takes (e.g. "Matt Johnson and Damon Smith competing for gold"). The photos will be connected with the "Bradford Judo Competition" –category that Jane had created.

1.3 Structure of the thesis

The thesis contains two parts. First we discuss metadata, ontologies, context information, and imaging from the perspective of mobile computing. In Section 2 we discuss metadata and ontologies in general and some of the available technologies for using semantic metadata. There is an overview on some metadata languages, tools and APIs, and desktop ontology editors / browsers. In Section 3 we discuss the possible uses of different type of context information. Section 4 deals with the specialties of imaging and metadata as well as researches on automatic image classification.

The latter part of the thesis discusses the content sharing prototype system. Section 5 contains the description of the architecture and the implementation of the prototype. The prototype is evaluated in Section 6. Finally, in Section 7, the conclusions of this thesis are drawn.

2 Metadata and ontologies

This section contains an introduction to metadata and semantic information. In addition, some of the available metadata and ontology languages, tools and toolkits, as well as metadata query languages are discussed. Some ontology browsers and editors for desktop environment are presented in the end of the section.

2.1 Metadata and semantics

Metadata is data about data. For example, in a traditional library catalog there is a metadata card about each book. It usually contains the author, the year of publication, the classification of the book, and possibly some keywords, a brief description, number of pages, and some other data.

In the world of computers, there are different types of metadata. One type of metadata describes the technical properties of the media, like the dimensions of an image in pixels, or the length of a file in bytes. These kinds of technical properties have been embedded into the files or the file system for decades, and they are widely used. Another type of metadata describes the content and the context of the piece of information. This can include classifications and brief free-text descriptions of the actual content, as well as the date of creation, the location of creation, the author, etc.

The content description metadata and context metadata help searching, especially when the content is not textual. Furthermore, a textual description in natural language is not always enough for effective search.

In natural languages, a single word does not always represent a single concept. For example, the word *sense* does not define an unambiguous concept. It may be interpreted as a verb or as a noun with one of the many different meanings or roles in phrases (e.g. “sense of humor”). Sometimes we need to describe the relations between different concepts or items. The use of cross-linked *vocabularies* and hierarchical classifications in a computer-readable form helps us to express the *semantics* of a concept and speak about things with common “words”. Vocabularies can be described with, e.g., the RDF Schema language, which is discussed below.

2.2 Metadata and ontology languages

For representing semantic metadata, there are various languages. The discussion of this thesis concentrates on the languages that are recommended by W3C. These include RDF, RDF Schema, and OWL that is a revision of DAML+OIL [McH03]. As a comparison point, the Topic Maps language is discussed briefly. For a more detailed view on metadata languages, see [HHV02].

In the prototype that is described in this thesis, RDF is used for managing metadata in both the terminal and the server. The advantage of RDF is that its basic structure is simple enough to be managed in a mobile terminal. In the future, OWL (Lite) could possibly be used for richer semantics and ontology mapping.

2.2.1 RDF and RDF Schema

The Resource Description Framework (RDF) is a language for representing metadata about Web resources, or about anything that can be identified with a uniform resource identifier (URI). RDF provides a framework for expressing the metadata so that it can be exchanged between applications without losing the semantics. This information is expressed in *statements* that have the form of *Subject – Predicate – Object -triples.* Any part of a triple can be a URI representing a resource, and the object can also be a literal [KCM03]. A triple forms an arc in a directed graph, where the subject and object represent the two ending nodes of the arc, and the predicate represents the labeled arc that connects the two nodes.

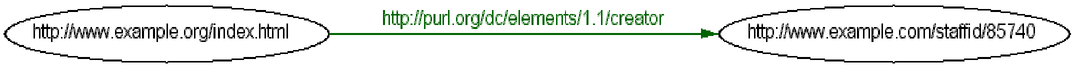
(a) “The creator of http://www.example.org/index.html is John Smith”:

(b) Subject: http://www.example.org/index.html

Predicate: http://purl.org/dc/elements/1.1/creator

Object: http://www.example.com/staffid/85740

(c)


http://www.example.org/index.html
http://purl.org/dc/elements/1.1/creator
http://www.example.com/staffid/85740

(d)

```
<?xml version="1.0"?>
 <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
          xmlns:dc="http://purl.org/dc/elements/1.1/">
  <rdf:Description rdf:about="http://www.example.org/index.html">
   <dc:creator rdf:resource="http://www.example.com/staffid/85740"/>
  </rdf:Description>
 </rdf:RDF>
```

Figure 1. A simple statement written (a) in English, (b) as an RDF statement, (c) as an RDF graph, and (d) in RDF/XML [MaM02] serialization format.

As an example, we can form an RDF statement from an English sentence (Figure 1). The sentence constituents are represented with URIs in the RDF model.

Using the RDF Schema language [BrG02] we can describe vocabularies that can be used in RDF documents. The vocabularies can include classes and properties. The RDF Schema language supports multiple inheritance in creating subclasses and subproperties. Compared to the XML Schema language [TBM01, BiM01], the RDF and RDF Schema languages concentrate on expressing the semantics of the metadata instead of the structure of the data [PaS02].

The recommended format for RDF serialization is XML-based RDF/XML [Bec02]. The semantics of RDF/XML is different from the semantics of an ordinary XML document. The triples can also be represented in other formats, such as N-triple[2] that is easy to parse on a computer, or Notation 3 (N3) [3] that is easier to write for humans. In addition, triples can be stored in relational databases.

RDF is defined in a collection of W3C documents. The only one that has gained the state of a W3C recommendation (until the end of the year 2002) is RDF Model and Syntax Specification [LaS99], and a version of the RDF Schema specification [BrG00] has been published as a candidate recommendation. The latest RDF document set (November 2002) includes six working draft documents, ranging from an introduction [MaM02] to the model-theoretic semantics [Hay02]. The November 2002 RDF specifications include a support for XML Schema datatypes.

RDF vocabularies

There are many RDF vocabularies and schemas in use. The *Dublin Core Element Set*[4] describes 15 predicates, e.g. `creator` (see the example above), `title`, `subject`, `language`, `description`, and `rights`. It also provides plain-text descriptions about the use of these predicates.

A specialized version of the Dublin Core element set is described in *PhotoRDF*, which is published as a W3C note [LaB02]. It uses a modified Dublin Core schema and adds technical and content schemas. The document describes a system that demonstrates how RDF and the Dublin Core elements could be used in describing photographs on the WWW. The system includes an application called RDFPic that embeds PhotoRDF metadata into JPEG image files by using comment blocks of a JPEG file [LaB02].

[2] See http://www.w3.org/2001/sw/RDFCore/ntriples/

[3] See http://www.w3.org/DesignIssues/Notation3

[4] See http://dublincore.org/documents/dces/

The *dmoz Open directory project*[5] defines a global cross-linked classification for web resources that is used e.g. in the Google search engine. The classification is huge: it takes over 35 megabytes of space in gzip-compressed RDF/XML format without the links to the categorized content.

The *W3C Composite Capability/Preference Profiles* (CC/PP) [KRW02] are also described in RDF. The profiles can be used for describing the properties and capabilities of a terminal, e.g. the supported content types, the screen resolution and the color capabilities. CC/PP profiles can be used for adapting the content to fit the properties of the terminal.

The *RDF Site Summary* (RSS)[6] is a lightweight, extensible metadata description and syndication format. Minimally, an RSS-document describes *channels* that can contain URL-retrievable items with titles and descriptions. The channels can be used, e.g., for delivering news feeds. RSS 1.0 includes Dublin Core, Syndication and Content modules. There are also many proposed modules available that include audio, company (stock), email, search, and streaming modules.

In plain RDF/RDF Schema there are no ways of mapping vocabularies directly to each other. Different URIs are interpreted as different elements, even if they have the same meaning. The mapping must be done at the application level, if the subclassing procedure of RDF Schema is not enough. For this problem, OWL provides a solution with ontology mapping (i.e. expressing equality of classes and properties) [SMV02].

2.2.2 DAML+OIL and OWL

The American ontology language project called DAML (DARPA Agent Markup Language) and the European project OIL (Ontology Inference Layer) joined

[5] See http://www.dmoz.org/

[6] See http://purl.org/rss/1.0/spec

forces to produce a common ontology language called DAML+OIL [CHH01]. It is an ontology language, based mainly on RDF and RDF Schema, with a support for XML Schema data types. The DAML+OIL specification was published as a set of W3C notes.

In 2002, the W3C Web Ontology working group used DAML+OIL as a basis for the design of the OWL Web Ontology Language [DCH02]. The working group decided to keep OWL more compatible with the RDF language, and waited for the decisions of RDF datatyping, which was first published in the November 2002 RDF working drafts. The XML Schema datatypes can be used in OWL like in RDF.

Using OWL we can describe classes, individuals, and properties with more complex structures than in RDF. The properties can be defined as symmetric or transitive properties, or inverses of other properties. Classes can be described as subclasses of other classes like in RDF Schema (`rdfs:subClassOf`). In addition, classes can be described with ontology mapping (`owl:sameClassAs`), disjoint operations (`owl:disjointWith`), set operations (`owl:unionOf`, `owl:interserctionOf`, `owl:complementOf`), and property restrictions. [DCH02]

The OWL language specifications describe three versions of the language: *OWL Lite*, which is a limited version, *OWL DL* (Description Logics) that includes the complete OWL vocabulary but is interpreted under a number of simple constraints, and *OWL Full* that includes all the properties of the language [SMV02].

OWL Lite supports transitive properties as well as equality, inequality, and datatypes. It supports cardinality restrictions with values limited to 0 and 1. The set operations for classes and the disjoint classes are not supported. Engines and tools for OWL Lite should be easier to implement than tools for the more complex versions of OWL. [SMV02]

An example of an OWL ontology is presented in Figure 2. It is a part of an ontology describing animals and humans. The example describes that the class `Person` is a union of the classes `Man` and `Woman`, and that `Man` and `Woman` are disjoint. An inference engine that uses the ontology can now tell that if Mary is an instance of `Person` and Mary is an instance of the negation of the class `Man`, Mary must be an instance of the class `Woman`. Furthermore, if the inference engine using the example ontology is told that Mary `hasChild` Joe, then it can infer that Joe `hasParent` Mary.

The OWL specification suite consists of six documents (November 2002 situation). For an overview on the documents, see the *Web Ontology Language (OWL) Guide* [SMV02] that is an overview for OWL.

2.2.3 Topic maps and XML Topic Maps (XTM)

The Topic Maps standard [ISO13250] specifies the SGML representation for topic maps. In addition to that, there is an XML version of the topic maps paradigm called XML Topic Maps (XTM) [PeM01], which is specified by the topicmaps.org consortium. Topicmaps.org is an independent consortium that aims in developing the applicability of the topic maps paradigm to the WWW. Members from companies including Empolis, InfoLoom, Mondeca, Ontopia, and Sun Microsystems have been working for the specification of XTM.

In the topic maps paradigm, there are *topics*, *occurrences,* and *associations* [Pep00]. A topic can describe a concept, a person, an entity, or anything we can talk about. A topic can have one or more *types*, which represent the class of a topic. An occurrence can be, for example, an image that displays the topic, or a description of a topic. Occurrences can have different roles, e.g. “article” or “illustration”. An association attaches one topic to another. Associations have types, e.g. “Helsinki **is the capital of** Finland,” states an association between the topics “Helsinki” and “Finland” with the type “is_capital_of”. In addition,

```
<rdf:RDF
  xmlns:rdf ="http://www.w3.org/1999/02/22-rdf-syntax-
    ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns     ="http://www.w3.org/TR/@@/owl-ex#"
>
<owl:Class rdf:ID="Animal">
  <rdfs:label>Animal</rdfs:label>
</owl:Class>
<owl:Class rdf:ID="Male">
  <rdfs:subClassOf rdf:resource="#Animal"/>
</owl:Class>
<owl:Class rdf:ID="Female">
  <rdfs:subClassOf rdf:resource="#Animal"/>
  <owl:disjointWith rdf:resource="#Male"/>
</owl:Class>
<owl:Class rdf:ID="Man">
  <rdfs:subClassOf rdf:resource="#Person"/>
  <rdfs:subClassOf rdf:resource="#Male"/>
</owl:Class>
<owl:Class rdf:ID="Woman">
  <rdfs:subClassOf rdf:resource="#Person"/>
  <rdfs:subClassOf rdf:resource="#Female"/>
</owl:Class>
<owl:ObjectProperty rdf:ID="hasParent">
  <rdfs:domain rdf:resource="#Animal"/>
  <rdfs:range rdf:resource="#Animal"/>
</owl:ObjectProperty>
<owl:Class rdf:ID="Person">
  <rdfs:subClassOf rdf:resource="#Animal"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasParent"/>
      <owl:allValuesFrom rdf:resource="#Person"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#Person">
  <owl:unionOf
    rdf:parseType="Collection">
    <owl:Class rdf:about="#Man"/>
    <owl:Class rdf:about="#Woman"/>
  </owl:unionOf>
</owl:Class>
<owl:ObjectProperty rdf:ID="hasChild">
  <owl:inverseOf rdf:resource="#hasParent"/>
</owl:ObjectProperty>
```

An Animal can be either Male or Female

Domain: Only animals can have parents
Range: Only animals can be parents

A Parent of a Person must be another Person

Every person is a Man or a Woman

x hasParent *y* ⇔ *y* hasChild *x*

Figure 2. A part of an OWL ontology describing Animals and Persons. The excerpt was taken from an OWL ontology example (http://www.daml.org/2002/06/webont/owl-ex).

associations have roles, e.g. the “is_capital_of” –association has the roles of “city” for Helsinki and “country” for Finland. The type of an association role is also a possible topic.

```
  <topic id="hamlet">
  <instanceOf><topicRef xlink:href="#play"/></instanceOf>
  <baseName>
    <baseNameString>Hamlet, Prince of Denmark</baseNameString>
  </baseName>
  <occurrence>
    <instanceOf>
      <topicRef xlink:href="#plain-text-format"/>
    </instanceOf>
    <resourceRef xlink:href=
      "ftp://www.gutenberg.org/pub/gutenberg/etext97/1ws2610.txt"/>
  </occurrence>
</topic>

<topic id="tempest">
  <instanceOf><topicRef xlink:href="#play"/></instanceOf>
  <baseName>
    <baseNameString>The Tempest</baseNameString>
  </baseName>
  <occurrence>
    <instanceOf>
      <topicRef xlink:href="#plain-text-format"/>
    </instanceOf>
    <resourceRef xlink:href=
      "ftp://www.gutenberg.org/pub/gutenberg/etext97/1ws4110.txt"/>
  </occurrence>
</topic>

<association>
  <instanceOf><topicRef xlink:href="#written-by"/></instanceOf>
  <member>
    <roleSpec><topicRef xlink:href="#author"/></roleSpec>
    <topicRef xlink:href="#shakespeare"/>
  </member>
  <member>
    <roleSpec><topicRef xlink:href="#work"/></roleSpec>
    <topicRef xlink:href="#hamlet"/>
  </member>
</association>
```

Figure 3. This example contains a part of an XML Topic Map that contains two topics that represent William Shakespeare’s plays, and an association between the topics of “Shakespeare” and “Hamlet”. The example was taken from [PeM01].

In topic maps, there are also the concepts of *identities*, *facets*, and *scopes*. While topics (e.g. countries) can be referred to by different names in different topic maps, we need a way to combine those different names under the same *identity*. This can be done using a public subject, for example a standardized list of country codes. We can use *facets* for assigning properties to information resources. *Scopes* allow limiting the range where an association is valid.

An example of representing topic maps in XTM format is presented in Figure 3.

2.3 Tools, APIs, and frameworks

In this section we discuss some of the open source toolkits available for RDF and RDF-based languages. The discussion concentrates mainly on the toolkits and APIs for the Java platform. Additionally, two RDF server or repository systems (Joseki and Sesame) are presented. In addition to the presented tools, there are many other implementations for e.g. C, Lisp, and Perl languages. Descriptions of more tools can be found in [SiH02].

Jena[7] [Bri02] is an RDF and DAML+OIL framework for Java by the Semantic Web activity from Hewlett Packard Labs. It includes support for storing RDF in relational database systems. The software package is big, and the required JARs take several megabytes (Jena 1.6.0). The software is developed continuously. It provides APIs for generating, managing and querying RDF and DAML+OIL models. In our prototype system, the Jena framework is used on the server side.

The *RDF API*[8] by Sergey Melnik from Stanford University provides basic functionality for managing RDF models in Java applications. The latest release for this API is from January 2001. The API is smaller than the Jena RDF libraries, and its JAR distribution takes about 300 kilobytes of storage space.

*KAON (*the *Karlsruhe Ontology and Semantic Web Tool Suite)*[9] is an open source ontology management tool suite. It includes an API for building ontology-based applications, and tools for ontology creation and management. It focuses on business applications. KAON uses an ontology language based on RDF with proprietary extensions for symmetric, transitive, and inverse relations, relation

[7] See http://www.hpl.hp.com/semweb/jena.htm

[8] See http://www-db.stanford.edu/~melnik/rdf/api.html

[9] See http://kaon.semanticweb.org/

cardinality, modularization, meta-modeling, and explicit representation of lexical information.

Joseki[10] is an implementation of an experimental WebAPI (former RDF NetAPI) for the remote query and update of RDF models. It includes a client API and a server, which can be run as standalone, embedded in a Java program, or inside an application server. The following operations are designed to be implemented in the RDF NetAPI [Sea02]:

1. Operations on the contents of a model: query and update
2. Operations on whole models: get and put
3. Operations about capabilities provided by a server:
   capabilities and range of operations on a particular RDF store.

Similar operations (query, update, and get) are implemented in the servlets of our prototype system. Our prototype implementation does not use the Joseki server or the RDF NetAPI, since we need to use location-based queries.

Sesame[11] is an open source RDF Schema based repository and querying facility that supports RDQL and RQL. It handles the requirements of the RDF model theory and supports the transitivity of the `rdfs:subClassOf` –property. The Sesame server requires a DBMS with JDBC support.

None of the presented tools and frameworks can be used in the J2ME applications on mobile terminals because of the software and memory requirements do not match the limitations of the mobile environment. However, the server applications of our prototype system use the Jena toolkit for managing the RDF metadata.

[10] See http://www.joseki.org/

[11] See http://sesame.aidministrator.nl/

2.4 Metadata query languages

There is not yet a standard for RDF query language, but there are many suggestions for an RDF query language by different parties. This section discusses few of the available RDF query languages. More information on RDF and Topic Map databases, as well as Topic Map query languages, can be found in the article [RSH02]. The available implementations of the presented query languages work on top of a relational database management system and a specific RDF toolkit.

Our prototype system does not use any of the query languages. Instead of that, we use the metadata model management methods provided in the Java API of the Jena framework.

RDQL is an RDF query language that looks closely like SQL. It is derived from the *SquishQL* query language. RDQL does not support transitive closures of subclasses and subproperties [MSR02]. This is one of the reasons why RDQL was not used in the prototype.

As an example, below is an RDQL query. It requests the items (?a) whose property `category` has a value that is a direct subclass of `attraction`. The items and their category classes (?b) are returned as response.

```
SELECT ?a, ?b
FROM   <http://example.org/rdf/data.rdf>
WHERE  (?a, <loc:category>, ?b)
AND    (?b, <rdfs:subClassOf>, <loc:attraction>)
USING loc FOR <http://example.org/rdf/loc.rdf#>,
      rdfs FOR <http://www.w3.org/2000/01/rdf-schema>
```

RQL [KAC02] is another SQL-based approach that was developed in the FORTH Institute of Computer Science. RQL supports the RDF and RDF Schema semantics including transitive closures of subclass and subproperty hierarchies

[KAC02]. It uses the SELECT-FROM-WHERE –query structure of SQL, and it adds its own syntax for expressing classes, instances, and relations.

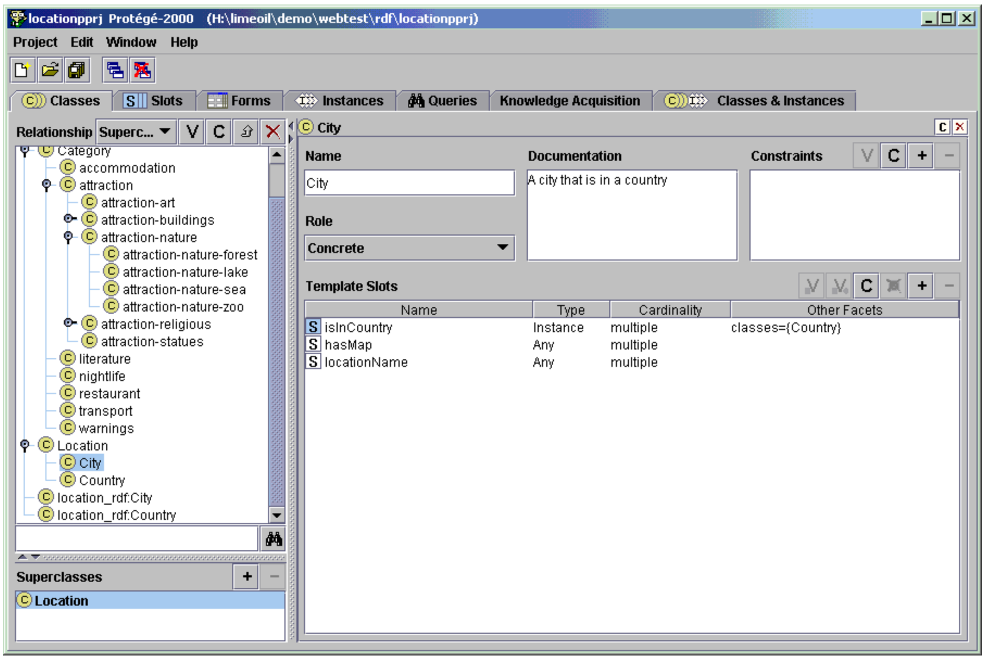
Below is an example of RQL query. When executed, it gets the items (X) whose property `category` has a value that is a transitive subclass of `attraction`. The query returns the items and the category classes ($Y) of the items:

```
SELECT X, $Y
FROM   {X}category{$Y}
WHERE  {$Y}<(attraction)
```

This kind of queries could be highly usable in the implementation of the server part of a community-shareable system.

TRIPLE [SiD02] is a query, inference, and transformation language for the Semantic Web. It was designed especially for querying and transforming RDF models. It allows semantics of languages to be described as rules, and this way it can use RDF Schema, Topic Maps or UML semantics. It includes also support for DAML+OIL.

The query language in *Profium SIR* (Semantic Information Router) is called *RDFQL*. It supports interactive queries, timed queries, and persistent queries [Saa02]. Timed queries are performed according to a schedule. For example, a stock query can be automatically performed every 30 minutes, and if there are interesting results, the end-user can be notified. Persistent queries can be configured to notify the user whenever information that matches the query is feed into the system.



Relationship: Superc...

Name: City

Documentation: A city that is in a country

Role: Concrete

Template Slots

Name	Type	Cardinality	Other Facets
isInCountry	Instance	multiple	classes={Country}
hasMap	Any	multiple	
locationName	Any	multiple	

Superclasses: Location

Figure 4. Protégé-2000 ontology editor from the Stanford University has various features, but unfortunately the user interface requires the user to study how to use the program.

2.5 Ontology editors, browsers, and viewers

For desktop computers, there are various ontology editors available. These, however, require a big screen size and are not portable “as is” to the world of mobile devices. This section discusses some properties of graphical ontology editors and ontology-based browsers. More ontology editors are described in [SiH02].

The *Protégé-2000* ontology editor[12] provides a tree-view access to the ontology (see Figure 4). It has various plug-ins, like a visualization tool that uses the open source *GraphViz* graph-rendering engine. It uses a proprietary internal representation format and it can import and export data in the RDF and RDF Schema format.

[12] See http://protege.stanford.edu/

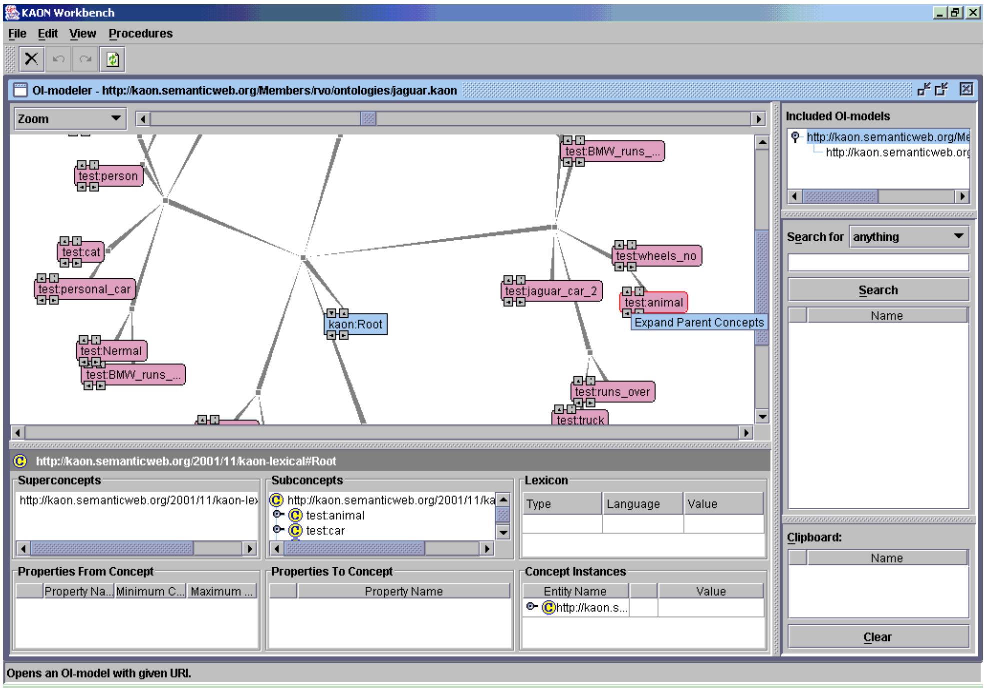
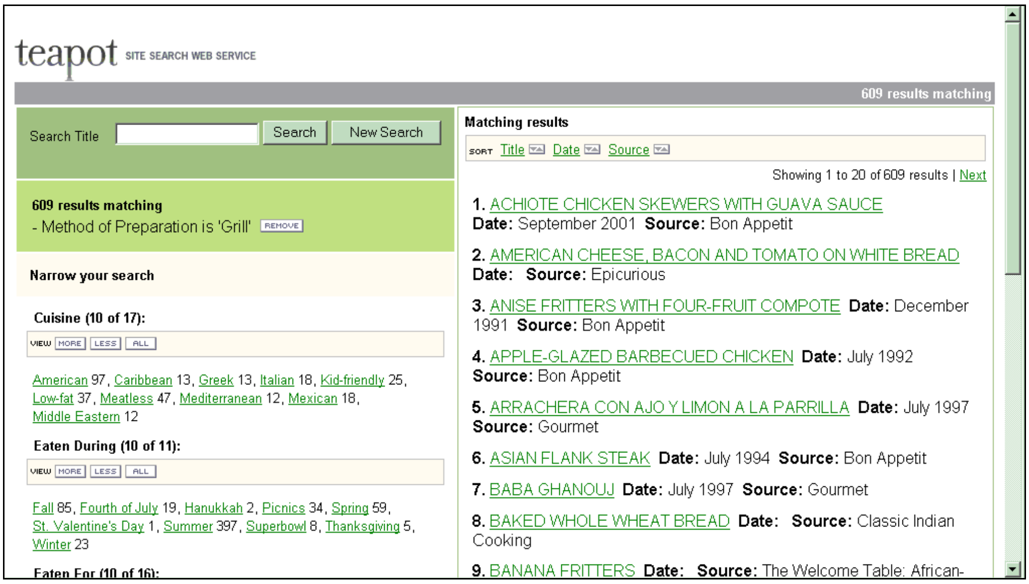


Figure 5. KAON OI-modeler (Ontology-Instance-modeler) provides both (hyperbolical) graph-based view and tree views for editing ontologies.

The *KAON Tool Suite*[13] includes an ontology editor, that supports graph- and treeview based editing (see Figure 5). The graph view has hyperbolical graph features, and it re-organizes itself during editing. *IsaViz*[14] is a graphical ontology editor that is based on the GraphViz rendering engine. Editing large ontology graphs in IsaViz seems quite cumbersome, because of the small fonts that require a detailed zoom to be readable.

[13] See http://kaon.semanticweb.org/

[14] See http://www.w3.org/2001/11/IsaViz/



teapot SITE SEARCH WEB SERVICE

609 results matching

Search Title Search New Search

609 results matching
- Method of Preparation is 'Grill' REMOVE

Narrow your search

Cuisine (10 of 17):
VIEW MORE LESS ALL

American 97, Caribbean 13, Greek 13, Italian 18, Kid-friendly 25, Low-fat 37, Meatless 47, Mediterranean 12, Mexican 18, Middle Eastern 12

Eaten During (10 of 11):
VIEW MORE LESS ALL

Fall 85, Fourth of July 19, Hanukkah 2, Picnics 34, Spring 59, St. Valentine's Day 1, Summer 397, Superbowl 8, Thanksgiving 5, Winter 23

Eaten For (10 of 16):

Matching results

SORT Title Date Source

Showing 1 to 20 of 609 results | Next

1. ACHIOTE CHICKEN SKEWERS WITH GUAVA SAUCE **Date:** September 2001 **Source:** Bon Appetit
2. AMERICAN CHEESE, BACON AND TOMATO ON WHITE BREAD **Date:** **Source:** Epicurious
3. ANISE FRITTERS WITH FOUR-FRUIT COMPOTE **Date:** December 1991 **Source:** Bon Appetit
4. APPLE-GLAZED BARBECUED CHICKEN **Date:** July 1992 **Source:** Bon Appetit
5. ARRACHERA CON AJO Y LIMON A LA PARRILLA **Date:** July 1997 **Source:** Gourmet
6. ASIAN FLANK STEAK **Date:** July 1994 **Source:** Bon Appetit
7. BABA GHANOUJ **Date:** July 1997 **Source:** Gourmet
8. BAKED WHOLE WHEAT BREAD **Date:** **Source:** Classic Indian Cooking
9. BANANA FRITTERS **Date:** **Source:** The Welcome Table: African-

Figure 6. The BPallen ontology browser provides a web user interface for filtering the search results by selecting from categories according to multiple classifications.

Web-based ontology browsers, such as the ones demonstrated by *Endeca*[15] and *BPallen*[16] (see Figure 6), provide filtering by selecting categories from different classifications. The image retrieval and yellow pages systems from the Semantic Computing group (SeCo) of HIIT (Helsinki Institute of Information Technology) include web-based ontology browsers that provide tree-view access to multiple ontological categorizations for filtering the search [HVH02, HSS02]. Thoméré et al. have presented a web-based ontology editing and browsing system that uses a Java applet for displaying the graphs of ontologies [Tho02].

The presented ontology browsers make use of the whole desktop PC screen size. None of them can be easily ported to the small screen of a mobile phone. Later in this thesis I present a graph-based mobile phone user interface component that is suitable for ontology-based filtering and annotation, as well as for simple ontology editing.

[15] See http://www.endeca.com/demos/demo_text.html

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

3 Context information in the mobile domain

The research area called *context-aware computin*g concentrates on the use of context information. A lot of context information is already available for application programmers in the current smart phone category mobile phones. In this section we concentrate on the context information in the mobile environment and discuss the issues of creating metadata from the context information.

In some fields of context-aware computing, e.g. *pervasive computing*, it is important to model the quality properties of the context information [HIR02]. There can also be multiple sources and multiple values for a single piece of context information, but these questions are out of the scope of this thesis. In the design of the prototype system we assumed that exactly one instance of each of the required aspects of context is available with adequate accuracy.

3.1 Different types of context

In this section, the context information is categorized into four categories. I use the semantic categorization by Schilit, Adams, and Want [SAW99] with additions from Chen and Kotz [ChK00]. Schilit's categorization contains *Computing context*, *User context,* and *Physical context*. Chen and Kotz added *Time context* as a separate category.

Some of the examples of using the different context categories in mobile phones are gathered from a white paper from Nokia, called Mobile Web Services Interfaces [Nok02]. It presents candidates for mobile web services interfaces that could be provided by the mobile operators for the third-party service providers.

3.1.1 Computing context

Computing context describes the resources available for devices: network connectivity, communication costs, and communication bandwidth as well as available printers, displays, and workstations [ChK00].

Terminal profile specifies the technical capabilities of the terminal. These include:

- Size of the screen (in pixels)
- Type of the screen (monochrome or color, number of colors)
- Support for features, e.g., WAP and Java
- Maximum memory sizes for WAP decks and Java MIDlets
- Support for different content types

Terminal profiles could be expressed as CC/PP device profiles [KRW02]. The use of terminal profiles could help in content adaptation, like the transmission of images in the correct resolution and format.

3.1.2 User context

User context can contain the user's profile, location, people nearby, and even the current social situation [ChK00].

Location

The most widely used piece of context information is *location*. There are different possibilities for sensing the current location of a device. The GPS satellite positioning system gives quite accurate results outside. Unfortunately the GPS signal is lost when the device enters a building [MaS00]. On a mobile phone it is possible to use the Cell-ID of the current cellular network base station to identify the current location. Many mobile phone operators, including Radiolinja and Sonera in Finland, already provide location-based services that use the information of current network cell [Rad02, Son02]. Vodafone is also building its own location-enabled service platform in many European countries [Vod02]. Location information may cost when the mobile network operators provide it, but the future price is an open question.

The most widely used global coordinate system is WGS-84. It is used in GPS and many of the mobile location services. However, the structure of the location information varies between services. For example, LIF (Location Interoperability Forum)[17] is a global industry initiative, which develops and promotes industry common solutions of Location Based Services. LIF has delivered a Mobile Location Protocol (MLP) specification, which defines an application-level protocol (with XML-syntax messages) for querying the location of mobile stations.

MLP serves as the interface between a Location Server and a location-based application. The protocol specification contains support for e.g. emergency messages, standard messages, and triggered messages. The specification supports inaccuracy zones for location information, and multiple coordinate systems, of which WGS-84 is obligatory.

User profile and presence

There is long-term and short-term information about the user available. A long-term *user profile* can contain for example nickname, occupation, gender, marital status, interests, and language preferences of a user. Short-term properties can contain information on the current end-user context (in a meeting, at work, at home, on a holiday).

[17] See http://www.locationforum.org

Gathering user profiles for each service separately by querying the user can be annoying. Most users don’t want to personalize themselves, even if they wanted to use personalized services. Using a centralized user profile could help in solving this problem. [Haa01]

User profiles could be stored on the phone or on a server maintained by e.g. a network operator or a third-party service provider. Mobile operators could gather user profiles, for example, when the user registers a mobile phone subscription.

Another way of gathering user profiles is to use social filtering methods. In social filtering, the users’ opinions about different items (books, food, wine, music) are gathered centrally when using the system, and compared between each other. Recommendations are made according to the opinions of similar users. [Haa01, p. 10-13]

Access management for the profile information is also needed. There is a W3C recommendation, P3P (Platform for Privacy Preferences), for managing the privacy of profile information for services on the WWW [Cra02]. With P3P, application programmers can create agents that can control the availability of, e.g., contact information for marketing purposes. In the mobile environment, the operator could offer the user profiles for third party services, according to the privacy preferences of the user.

In the Mobile Web Services Interfaces white paper [Nok02], *presence* is defined as a short-term, dynamic user profile. Presence information could contain the user’s contact information, as well as the current social context and the willingness to participate in a certain type of communication. In the current mobile phones, the user can manually profile her mobile phone by selecting a desired notify level from a list (general, meeting or silent) according to the current social situation. This sets some properties of the mobile phone, mainly the types (e.g. ringing tone, beep, vibration, or none) and volume of user notification sounds for different types of events (e.g. incoming call, calendar alerts, or incoming message). The same method can be used in context-aware messaging applications.

3.1.3 Physical context

Physical context contains lighting, noise levels, traffic conditions, and temperature [ChK00]. Physical context information can be gathered from sensors like thermometers (temperature), light sensors (lighting, inside/outside, pocket/table), microphones (noise levels), or accelerometers (gestures, movement, position). The information of these sensors could be used in, e.g., guessing the current social situation, identifying gestures and movement, and determining the location of the device. [Tuu00, p.53]

3.1.4 Timed context

Timed context contains time of the day, week, month, and the time of year [ChK00]. It can be used together with the other context classes to produce context-based notifications and service search. At least, searching for sports events, exhibitions, opening times, and such could benefit from the use of the timed context. In countries that have a big difference between seasons, such as in Finland, it is not relevant to promote e.g. skiing possibilities in the middle of summer or water sports in the middle of the winter.

Our prototype system stores the time information as part of the metadata of the image, but it does not use it as a search criterion.

3.2 Context information as metadata

Current smart phones have the potential of attaching context information to the media created with them. Since we are able to get the location information and identify the user in a mobile phone, it is quite straightforward to embed this data into the photographs. By storing the available context of the image (or a part of the context) as metadata, we can use it later for searching.

In the scope of relevance of search, I think that the most informational parts of context to be embedded into images would be the user context and time context. Computing context becomes useful when we think of content adaptation and delivery. The use of physical context could help in automatically classifying the content.

Chen and Kotz describe the term *Context history* for time-stamped recording of the computing context, the user context, and the physical context [ChK00]. The prototype application that is presented later in this thesis, gathers pieces of the context history as metadata of images.

When handling context information, the privacy needs to be concerned [HIR02]. There may be situations where pieces of the context history of other people can be identified, even if they didn't want to. This can happen, for example, if we store the identifiers of surrounding people when taking a photo, and later share the photo (including the embedded metadata) with other people.

One solution for managing the privacy of context information could be to use the Platform for Privacy Preferences (P3P) [Cra02]. P3P is a W3C recommendation for managing privacy rules and preferences. The service providers give information about the privacy policies of their services. The user can select, which kind of information can be given to whom and for which kind of use. The user agent can store a set of preferences about privacy policies using a language called APPEL (A P3P Preferences Exchange Language).

3.3 Researches on context-aware tourist information applications

There are some context-aware prototype implementations of tourist information guides that use location information on Web Pads and PDA's. These include the GUIDE project [CDM00] and the Cyberguide project [AAH97]. Both of these systems provide tourist information for people visiting a city or a site.

In the GUIDE system [CDM00], the user borrows a terminal from the tourist information booth of a city. The terminal is a Pad PC that has a grayscale screen of 800 by 600 pixels and runs the Windows 95 operating system. The GUIDE system uses WLAN cell-based location information and photos for determining the user's location. The user can plan a route for his trip, and display web pages according to the current location. The system also maintains history information about the user's path.

The Cyberguide system gives navigation aids for indoor and outdoor use [AAH97]. The system uses PDA devices with external positioning equipment as a terminal. The Outdoor-Cyberguide is designed to work as a context-aware tourist guide on the city. The Indoor-Cyberguide is designed to work only inside, e.g. on open house days, and it uses infrared beacons (made of remote controllers) for getting location information. In addition to aiding navigation, Cyberguide allows displaying information about items at the current location, and interaction by using email. There is also a version called CyBARguide that allows the users to modify the database of services and add their own comments and ratings.

For more examples of context aware applications see the survey by Chen and Kotz [ChK00].

4 Imaging and metadata

In this section I describe some properties of metadata in imaging, mobile imaging, and recent research in automatic image classification and non-traditional ways of annotation.

4.1 Metadata in image files in current applications

Metadata (or non-payload data) is embedded into image files in many of the current image-creation applications. There are some competing frameworks for managing metadata in images. I present some examples of how metadata is embedded into image files in the frameworks.

W3C has presented a scheme for annotating photos. The system, called *PhotoRDF* [LaB02], consists of a server and an annotation application. The annotation application, RDFPic, inserts Dublin Core metadata elements as RDF into the comment block of the JPEG file. The server side of the system uses Jigsaw web server platform[18] for Java that supports RDF metadata extraction from JPEG image files. PhotoRDF uses an RDF vocabulary based on the Dublin Core element set.

Adobe XMP (Extensible Metadata Platform) has its own block-encodings for embedding RDF-based XMP-data into JPEG, TIFF, PNG, GIF, and PDF files. All the current Adobe products have support for XMP metadata[19]. XMP uses the RDF data model, and it adds e.g. a proprietary datatyping system. XMP schemas describe properties that handle technical data about different types of media, as well as rights management and media management. The XMP platform manages the granularity of media by allowing annotations on parts of a document.

[18] See http://www.w3.org/Jigsaw/

[19] See http://www.adobe.com/products/xmp/main.html

The XMP specification describes means for embedding the metadata in different types of media files. In a JPEG file, the XMP metadata is stored inside an XML Packet (XPacket) that is put into an APP1 block of the JPEG file.

DIG-35 [DIG01] is an XML-based image metadata framework. The *JPX* metadata format, which is used in JPEG 2000, is based on it [CIH02]. The specification includes categories for image creation, content description, history, intellectual property rights, and image identification metadata.

Image creation metadata describes the technical properties and settings of the camera or scanner. Content description metadata describes the caption, the location, and the persons and things in the photo. DIG-35 can even be used for describing events that happened during the creation of the media. In addition, the content description metadata can contain an audio stream with a spoken description of the photo.

In the *Java ImageIO framework*, which is included in J2SE 1.4 and later, there is a package called `javax.imageio.metadata` that provides support for metadata in images. The metadata in images is considered to be “anything that is not pixels”, e.g. technical metadata in an image format specific way, and content metadata. The ImageIO framework handles metadata in XML-format, and provides a DOM interface for application programmers.

4.2 Mobile imaging

In the year 2002, a number of mobile phones with digital cameras were brought to the market. Some of the phones are delivered with integrated cameras, and some have a camera as an external accessory. Digital images can be transmitted between phones using multimedia messaging, or uploaded to a server, like in the Club Nokia Photo Zone service.

The WAP-browsers in the current mobile phones have support for images. The MIDP Java platform in mobile phones has also made it possible to use images in user-created applications.

In the Nokia 7650 mobile phone, the Camera application stores photos in JPEG format with VGA resolution (640x480 pixels). It stores the date and time information as a line-feed delimited ASCII string into the JPEG comment block of the image file. It also creates a thumbnail of the image and stores it as a separate file. The user can manage the images on the phone by renaming the files and storing them into folders. Image management is done in the Images application that displays a list of images with thumbnails, filenames, and date stamps.

Currently, the Nokia 7650 mobile phone attaches the following metadata to the created image files[20]:

<Filename, date and time, phone model>

The prototype application described in this thesis stores the following information, and provides means for adding more properties:

<Title, description, category, date and time, location, author, …>

A mobile phone equipped with a camera has the potential for gathering context information and for attaching it to the created images or other media. While we can use location information and efficiently identify the user in a mobile phone, it would be straightforward to embed these into the images taken with the camera of the phone. By storing the available context of the image, we can use that context information later for searching.

[20] In addition, the Club Nokia Photo Zone service on the WWW/WAP allows entering a caption, a plain text description, and keywords for the images that are stored in the system.

4.3 Alternative approaches to image categorization and annotation

While our prototype uses manual categorization and annotation, there are also other ways for classifying and annotating images. This section discusses some of the recent research on alternative methods of categorization and annotation. The presented methods could be feasible in a mobile phone environment in the future, but the issues with the required computing resources should be carefully estimated.

4.3.1 Automatic image classification

According to recent research, a timed sequence of photographs can be *automatically segmented* to events according to the content of the images [StL02]. There has also been research on automatically grouping single images from a photo library into semantically meaningful categories using low-level visual features. There are researches on *semantic classification* of photographs into indoor and outdoor scenes [SzP98], city versus landscape and different forms of landscapes [VFJ01]. The accuracy of semantic classification in the current systems can be up to 90–95% in City/Landscape and Indoor/Outdoor classification [VFJ01].

Face detection methods can be used for finding human faces from photographs [RBK98]. *Face recognition*, in addition, can be used for recognizing the person in the photo. At least some of the face detection and face recognition methods use neural networks, which brings out speed issues, at least on a mobile phone implementation.

4.3.2 Annotation by speech

There are researches of annotating images with speech. A system called SRIM, developed at the Media Technology Laboratory of the Helsinki University of Technology, uses the Java Speech API and Microsoft speech recognition engine. The software embeds the recognized speech as metadata into PNG image files [Kiv01]. The system stores the description from the speech and looks for pre-defined keywords to determine e.g. the filenames and folders. However, in the usability evaluation of the system [Vak02], problems were found with the speech recognition. If there was any background noise, the system failed to recognize the speech input. There were also problems with pre-recorded speech. These problems seemed to be caused by the properties of the speech recognition software, not the annotation method itself. More developed speech recognition software that is capable of recognizing speech even in a noisy environment would make the speech annotation method more usable.

5 Prototype system for mobile content sharing

A prototype system was created as a part of this thesis work for demonstrating the use of metadata in sharing mobile content. The prototype provides means for using RDF metadata in a mobile application and storing information about the current context as metadata of the content.

The prototype system is configured to work as a tourism portal that allows users to share images by manual semantic classification and automatically gathered context information. By changing the ontology, the same system could be used for sharing images of any specific interest area.

In this section I describe the requirements for the design of the prototype and the architecture of the prototype system. I present the user interface of the system, the implementation issues, and the use of ontologies in the prototype.

The presented prototype system is evaluated and compared with other means of mobile information retrieval in the next section.

5.1 Requirements and design for the prototype system

On the background of the prototype design there were the research problems and the scenarios presented in Section 1. The properties of the Nokia 7650 mobile phone were used as a base for the design. One of the main problems was to fit the ontology browsing and management user interface into the screen size of the mobile phone. Another goal was to keep the amount of obligatory textual user input small, yet being able to define the search criteria efficiently. A third problem was to keep the memory requirements small enough for the terminal end, to make the system feasible with real mobile phones.

These problems were solved as follows. A graphical ontology browser that displays the surroundings of one node at a time in a graph and is navigable with a 4-direction joystick was designed and implemented. The content and metadata were chosen to be stored on a server-side repository. A limited lightweight RDF engine for the mobile phone was implemented. The system uses location information together with the joystick-navigable ontology browser for filtering the search results.

5.2 System architecture

The prototype system is built of the following parts (see Figure 7):

A. The terminal. On the terminal, we run a custom camera application with annotation capabilities and a browser application for searching annotated content.

B. Mobile web services. These include the Location web service and the User Profile web service. Mobile web services are simulated on a normal web service server.

C. A repository server. The server stores and manages both content data and metadata.

The different parts of the system have no implications to future Nokia products, even if some of them are named after current components in some Nokia products and white papers.

5.2.1 Mobile terminal

The terminal part of the prototype was first decided to be implemented on the J2SE platform, because of the limitations of the J2ME. In the beginning the user interface properties of the Nokia 7650 mobile phone were simulated in a J2SE application with bitmap skins and buttons. This made the design and the implementation of the user interface components more flexible, and a ready-

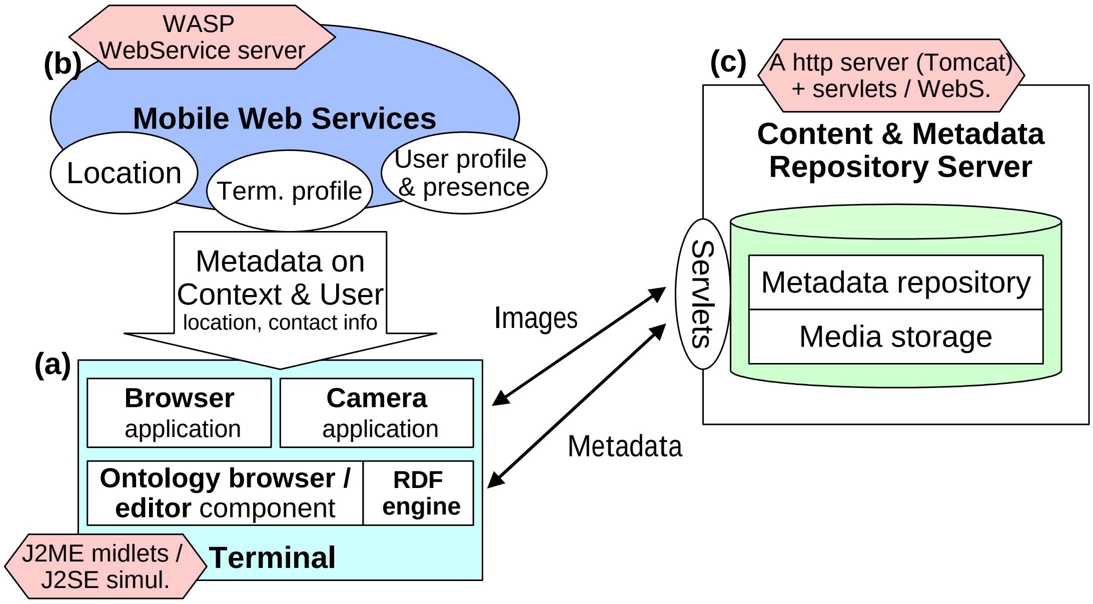


Figure 7. The prototype environment is built of (a) the terminal end applications, (b) the mobile web services simulation, and (c) the repository for content and metadata.

made RDF engine could be used. It helped in finding out the requirements for a possible phone implementation.

The self-made terminal user interface simulator is a J2SE application with a bitmap skin of the Nokia 7650 phone. It supports on-screen simulation of the joystick and the action buttons of the phone. This way the input methods and the properties of the mobile phone screen can be simulated. The Camera and the Browser applications were created on the top of the simulator. The Camera application includes annotation capabilities and the Browser application can be used for searching images based on classification and location.

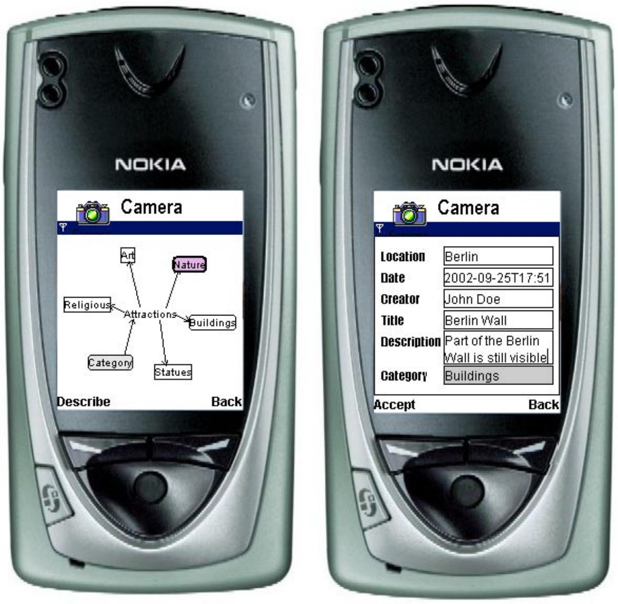
After getting knowledge with the terminal simulation, the terminal end applications were quickly implemented as J2ME midlets that can be run on the Nokia 7650 mobile phone. For that we needed to create a simple, limited RDF engine that is able to input and output triples and to perform simple queries according to the fields of the triples.

The terminal-end components and applications are described in the following paragraphs.

Ontology browsing and editing

I have created a mobile ontology browser component (Figures 8a and 12) that allows the user to navigate through an ontology class structure on a mobile phone screen using the four-direction joystick of the 7650. The screenshots were taken from the J2SE version of the software.

The ontology browser allows the user to classify media or enter search criteria from the ontology. The component also includes a simple editor that allows extending the class structure of ontology, if the user is allowed to modify the ontology. The access management problem was left outside the scope of the prototype implementation.



Location: Berlin
Date: 2002-09-25T17:51
Creator: John Doe
Title: Berlin Wall
Description: Part of the Berlin Wall is still visible
Category: Buildings

(a) (b)

Figure 8. Annotating photographs by using ontologies. (a) Navigating through the ontology graph to select the category. (b) Adding free-text descriptions.

Camera and Browser applications

When taking a picture, the Camera application allows the user to describe the content by selecting a category (Figure 8a) and by entering free-text descriptions for the photo (Figure 8b). The software gets the location information (coordinates and location name) automatically from the Location web service, and the user contact information (name, e-mail address) from the User profile web service. The application uploads the image and the gathered metadata to the metadata repository.

The Browser application sends queries to the metadata repository. The repository engine performs the query, gets the requested items and sorts the items according to the distance. The user can select a node from the category graph in the ontology browser to get a list of items that match the selected category. After selecting an item in the list of results, the user can select to display the content (the actual image) and the detailed metadata of the item, or a map that shows the current location of the terminal and the location of the selected item. The map image is generated on the fly by the GIS module that can be used through the Location Web Service.

5.2.2 Mobile web services

The system uses three web services that are named after the mobile web services interfaces in the Nokia white paper [Nok02].

The *Location web service* is implemented in the prototype by using a map window where we can point and click the "current location". The terminal-end applications request the current location from the location web service. The current coordinates (in the WGS-84 coordinate system) and the current location name are transmitted in the response.

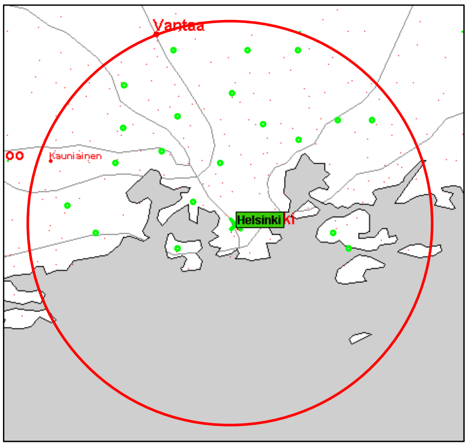


Figure 9. The browser displays a list of the items that are located within the selected geographical distance and match the selected category.

The user name and contact information are requested from the *User profile web service*. In the future, this service could include the interest of the user, preferred food, language preferences, etc.

The *Terminal profile web service* could be used in the future for receiving the dimensions and color properties of the terminal screen, as well as the supported image formats.

5.2.3 Repository for content and metadata

The repository stores the content (digital photos), the metadata describing the content, and the ontologies that are used in the metadata description. The terminal applications request ontologies or parts of them from the server (get and query functions), and send new metadata of the photos to the server (update function).

The query can include the current location (or location of interest) as well as a selected category. The result set includes only items within a reasonable distance, which is 30 kilometers in the prototype by default (see Figure 9).

When a user with sufficient access rights edits the ontology, the ontology editor component sends the updates to the repository server. When new images are added, the metadata is also sent to the repository server.

The metadata server uses some functions similar to the ones that were presented in the RDF NetAPI [Sea02]. The server needs to support location-based queries and RDF Schema transitivity, which are not supported in the Joseki implementation of the RDF NetAPI server, so we needed to create a server of our own.

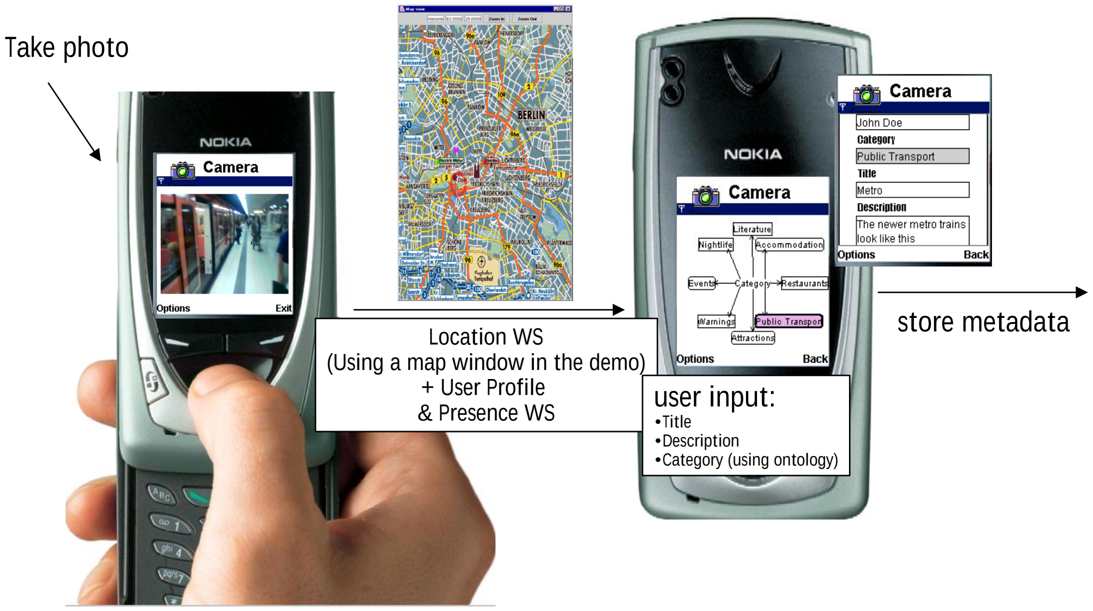
5.3 User interface

The user interface issues are divided into three parts. Firstly, there are two main use cases. One is entering the content and metadata. The other one is searching for items in the system. Secondly, both of these use cases utilize the ontology browser, for classifying data or for filtering the search results.

5.3.1 Entering content and metadata to the system

The prototype includes a content description tool. It is implemented in a Camera application –look-alike midlet (see Figure 10 for details). Another possibility would have been to only store the automatically gathered metadata (location and author information) in the Camera application and to integrate the free-text description part to an Image management application. In the real Images application of the Nokia 7650 phone, the images can currently be renamed and put into folders and the user can even upload selected images to Club Nokia Photo Zone service[21].

[21] The Club Nokia Photo Zone is a service that can be used on the WWW and WAP. It allows entering a caption, a plain text description, and keywords for the images that are stored in the system. The images can be uploaded directly from certain phone models to the service. See http://www.club.nokia.com/ for details.



John Doe
Category
Public Transport
Title
Metro
Description
The newer metro trains look like this

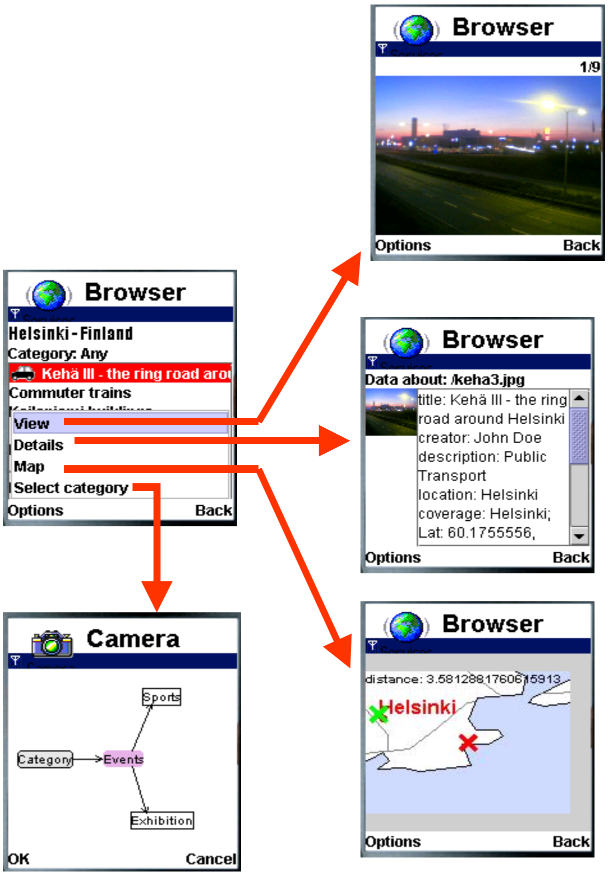
Figure 10. Annotating the photographs. The user takes a photo. The system retrieves the current location from the location web service. The user selects the corresponding category from the ontology browser, and enters a free text description. The system automatically gathers the location and author information. The user accepts the metadata, and it is stored together with the image to the repository.

5.3.2 Searching for images

The prototype includes a browser application for searching the content (images) that is fed into the system (Figure 11). It filters the items by the following criteria:

1. Distance between the current location and the location of the item
2. The category that can be selected with the ontology browser component

The system allows icons to be used for representing each category. This way we can display a visual hint of the category in the list of results for better human-eye filtering of the results. However, creating such icons is currently not possible in the prototype, and the icons should possibly be created and attached by using a desktop system.



Data about: /keha3.jpg

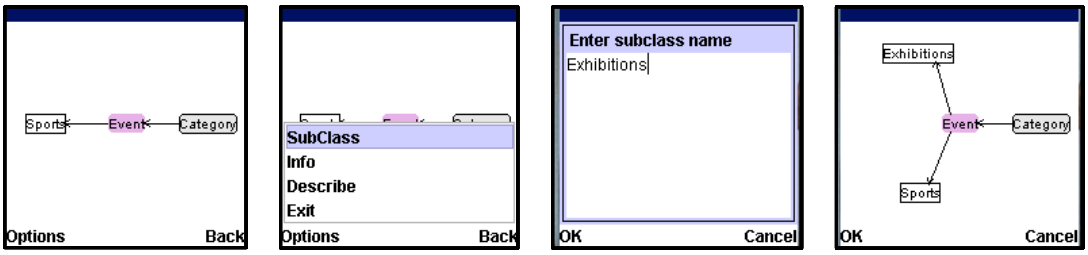
title: Kehä III - the ring road around Helsinki
creator: John Doe
description: Public Transport
location: Helsinki
coverage: Helsinki;
Lat: 60.1755556,

Figure 11. The browser application. From the list of results the user can select to view the image, the details or the map. The user can use the ontology browser to select a category for filtering search results.

If the ontology had icon representations for each class, we could use the icons for browsing the ontology on smaller screens. Each class could even have a vocal representation in addition to the textual and visual symbols.

5.3.3 Browsing and editing ontologies

The prototype includes a user interface component for mobile management of ontologies. The component allows users to navigate through an ontology graph on a mobile phone screen using the four-direction joystick of the 7650. The user can classify media or enter search criteria from the ontology by using the component. In addition, the component includes a simple editor that allows extending the class structure of the ontology by creating subclasses. The editing feature of the component is show in Figure 12.



Enter subclass name

Exhibitions

Figure 12. The ontology editor. The user with appropriate access rights can use the ontology editor for extending the ontology by adding subclasses.

5.4 Implementation

The class structures of the system with UML class diagrams are presented in this section. A sequence diagram for the Search use case is presented in Appendix A.

5.4.1 Client-side class structure

In this section, I describe the J2ME (midlet) version of the client software. The client uses a self-made limited RDF engine implementation for handling RDF statements.

There is a Mobile Media API for J2ME[22], which has camera support, but it is not included in the MIDP 1.0 that is implemented in the 7650 phone. Because of this the camera functionality is simulated. The location coordinates of the terminal must be retrieved and handled on the server side, since in the current versions of J2ME there is no way to find out the current location. The J2SE version of the client uses web services for requesting the current location, but this feature is not implemented in the current J2ME version of the client software.

[22] See http://java.sun.com/products/mmapi/

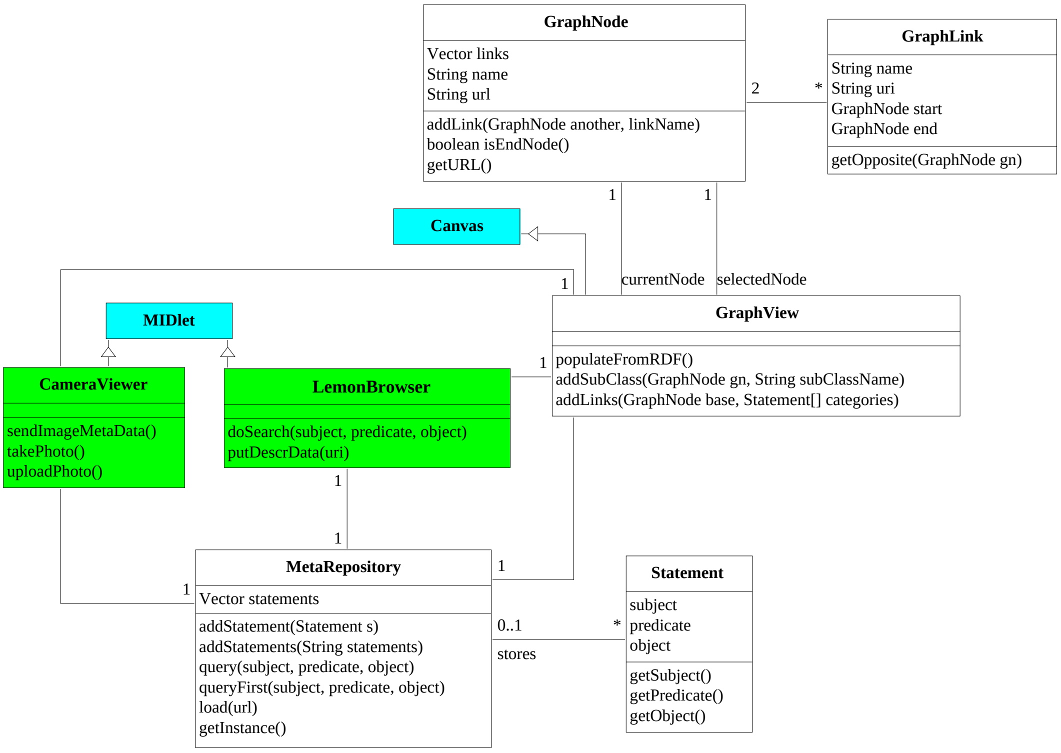


Figure 13. Client side class structure. `LemonBrowser` and `CameraViewer` are the main MIDlet classes. `GraphView` is the ontology browser component. `MetaRepository` is the limited lightweight RDF engine implementation.

Figure 13 depicts a class diagram for the J2ME implementation of the terminal end applications. The limited RDF engine has two classes: `MetaRepository` that stores RDF statements and handles the input of the statements in the N-Triple format, and the class `Statement` that represents the data structure of an RDF statement. The RDF engine stores subclass hierarchies and can perform simple queries, but it does not implement all of the features in the RDF model theory.

The graphical ontology browser consists of three classes: `GraphView`, which contains the painting and navigating methods for the ontology browser, and `GraphNode` and `GraphLink`, which represent the nodes and arcs in the data structure of the RDF graph.

LemonBrowser is the main class for the search tool midlet. It binds together the user interface of the browser application, and manages displaying the search results. It uses the selected classification and the location information for filtering the search of interesting items.

CameraViewer is the main class for the image annotation tool midlet. It simulates the Camera application of the Nokia 7650 phone with extended annotation capabilities. It gathers the context, and uses the ontology browser / editor component for classifying the content. Moreover, it uses the MetadataInputForm class for requesting the free text descriptions from the user and displaying the metadata.

5.4.2 Repository server class structure

The repository server runs on top of the Tomcat web server. The repository functionality is implemented by using three servlets. In Figure 14 you can find the servlet classes.

In MetaServlet, we process the metadata operations. Internally it uses the memory-based DAML+OIL model from the Jena toolkit for managing RDF data. The DAML+OIL-version of the model is used, because the system uses the

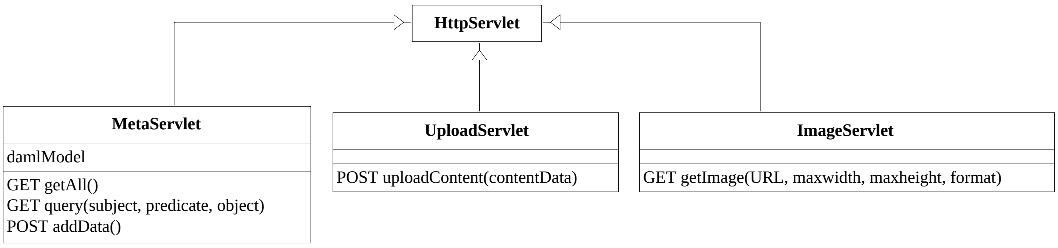

HttpServlet
MetaServlet
damlModel
GET getAll()
GET query(subject, predicate, object)
POST addData()
UploadServlet
POST uploadContent(contentData)
ImageServlet
GET getImage(URL, maxwidth, maxheight, format)

Figure 14. There are three servlets in the system. MetaServlet handles the metadata of images. UploadServlet takes care of incoming images. ImageServlet manages the content-adaptation for the user's device by adjusting the image size and format.

transitivity of subclasses, which is not supported in the RDF model of Jena. The servlet is used for storing metadata about items and for querying metadata. After adding metadata, the servlet stores the whole model into a pre-defined file using the RDF/XML serialization. The Jena toolkit includes support for storing the RDF metadata into a relational database, but I chose to keep the server lighter and not to include a DBMS in the system. For communicating with the terminal, `MetaServlet` uses the N-triple syntax of RDF, since it was easier to implement on the terminal end than the official RDF/XML format.

`UploadServlet` is designed for storing the content on the repository server. It stores the uploaded photographs on the web server to a pre-defined path, and returns the URL for the stored resource, so the client can create RDF triples with that resource URI.

`ImageServlet` handles the content adaptation when the client needs to display photographs. The images are shrunk to fit the screen of the mobile device, and transmitted in a requested format. This helps in keeping the communication costs small, since it reduces the byte size of the transmitted images significantly. On the other hand, the J2ME environment does not even support image scaling, since it could be quite a heavy operation for many mobile terminals.

5.4.3 Web Services class structures

The web services use the WASP Server for Java by Systinet[23]. The developer version of the server is available as a plug-in for the Sun ONE Studio for Java. The web services are presented in Figure 15 and described below.

`LocationWS` is the web service that handles the location information and geographical maps. The prototype implementation of `LocationWS` receives the

[23] See http://www.systinet.com/products/wasp_jserver/overview

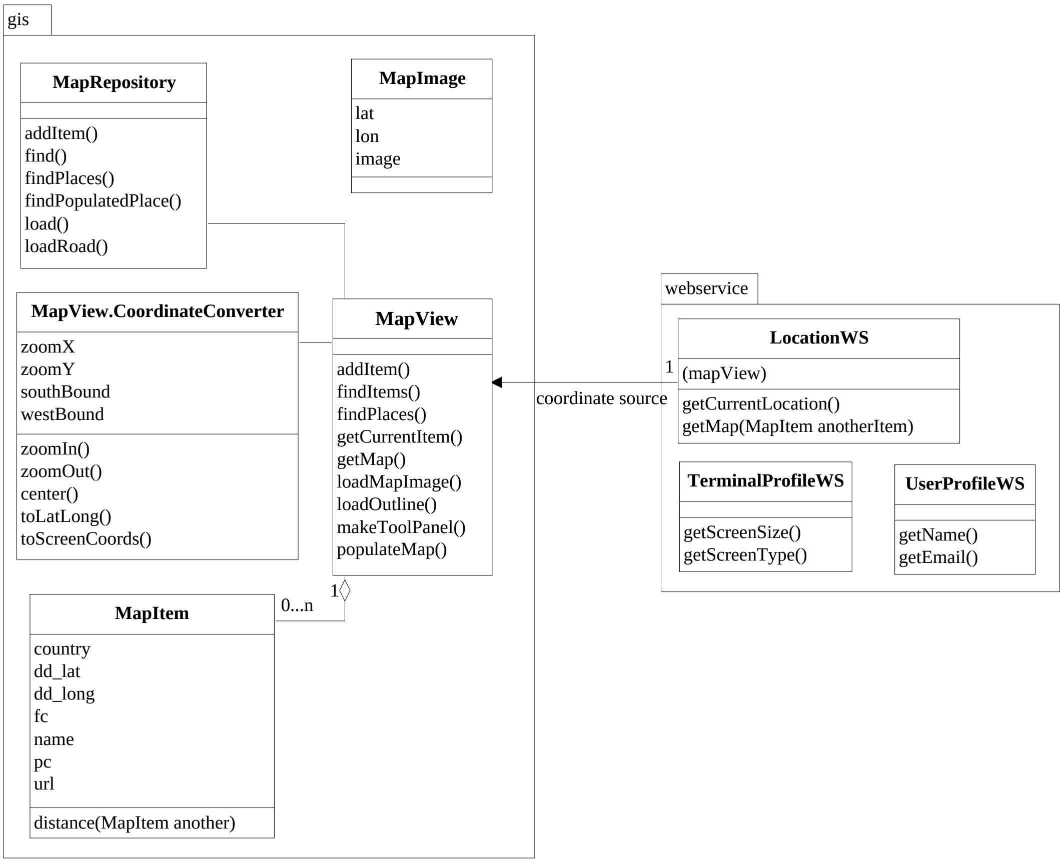


Figure 15. The class structure of the web services and the GIS module. The class `MapView` is used for simulating the current location with an on-screen map. It also produces bitmap images from vector map data to be displayed in the client.

current location from the class `MapView`. The user can enter the “current location” by clicking in a map window, and the location web service gets the coordinates and the name of the location. In a real implementation, the network operator could provide the location web service, and the location information could be retrieved from the cellular network. The current prototype implementation of the location web service allows requesting a map with the current location and the location of the selected item highlighted.

`UserProfileWS` handles the contact information of the user. It just returns the name and email-address of the user. In the future, this could contain more usable information of the user, like the hobbies, profession, gender, age, and marital status. These could be used for sorting the search results.

`TerminalProfileWS` allows retrieving the terminal properties on the server side. These include the size and type of the screen of the terminal.

5.4.4 GIS module

The server part of the prototype includes a simple Geographic Information System (GIS) module. It uses geographic names freely downloadable from the Geographic Names Database[24], maintained by the U.S. National Imagery and Mapping Agency. For vector maps, it can use the data available from the Digital Chart of the World (DCW)[25] in ArcView export format. The GIS module was implemented for experimenting the use of location information in the simulation. It is able to render maps for selected regions, if the vector map data is available. It can also display objects on a map, and draw a map with the current location and a selected item. The classes of the GIS module are presented in Figure 15.

5.5 Ontologies in the prototype

The prototype system uses the Dublin Core properties for expressing the title, description, creator, and location data. It also presents a vocabulary of its own for categorizing and expressing relations between the categories.

The image annotation part of the prototype creates seven statements for each photograph. In the ontology editor, three statements are created for each new subclassed category. See Figure 16 for an example of these. The available memory of the device limits the maximum number of statements that can be handled at once on the terminal.

[24] http://www.nima.mil/gns/html/

[25] http://www.maproom.psu.edu/dcw/

```
1:  <rdf:Description rdf:about='&loc;transport'>
2:    <rdf:type rdf:resource=&rdfs;Class'/>
3:    <loc:icon xml:lang='en'>transport.png</loc:icon>
4:    <rdfs:comment xml:lang='en'>Public Transport</rdfs:comment>
5:    <rdfs:subClassOf rdf:resource='&loc;Category'/>
6:  </rdf:Description>
7:  <rdf:Description rdf:about='&loc;transport-Station'>
8:    <rdf:type rdf:resource=&rdfs;Class'/>
9:    <rdfs:subClassOf rdf:resource=&loc;transport'/>
10:   <rdfs:comment>Station</rdfs:comment>
11: </rdf:Description>
12: <rdf:Description rdf:about='http://myhost/pics/railwaystation-
    2.jpg'>
13:   <dc:title>Helsinki Railway Station from the street</dc:title>
14:   <loc:category rdf:resource='&loc;transport-Station'/>
15:   <dc:date>2002-10-03T13:12</dc:date>
16:   <dc:description>The station was designed by the Finnish
    architect Eliel Saarinen</dc:description>
17:   <dc:coverage>Helsinki; Lat: 60.20243, Long:
    24.93216</dc:coverage>
18:   <loc:location rdf:resource='&loc;Helsinki'/>
19:   <dc:creator>John Doe</dc:creator>
20: </rdf:Description>
```

Figure 16. In this excerpt of the metadata repository, we use the namespace dc for Dublin Core vocabulary, and the namespace `loc` for our categorization vocabulary. The category class `transport` (lines 1–6) was created by hand, and its subclass `tranport-Station` (lines 7–11) was automatically created using the ontology editor of the system. Lines 12–20 show an annotation for a photo that was created with the system.

The current implementation handles and stores the ontology and the statements about the photographs all in one place. Support for multiple ontologies could be added in the future. This would allow more flexible usage of third-party ontologies.

6 From the prototype towards the future

In this section I evaluate the presented prototype, its limitations and compare it to other means of mobile information retrieval. In the end of this section, I present future development ideas and a SWOT analysis for the system as a tourist information portal.

6.1 Limitations of the implementation environment and the prototype

The general limitations of the implementation environment, especially the limitations of the J2ME environment, had an effect on the prototype. The memory limitations were not a big problem. The API limitations, instead, forced us to make compromises and workarounds. The prototype system itself has some limitations that are discussed in this section.

6.1.1 Memory limitations

In mobile phones, one critical issue is the size of the applications. Both the runtime memory footprint and the storage size of the midlet are limited. In the Series 60 phones (e.g. the Nokia 7650 phone), these are not that critical, since the applications can take up to 4 megabytes of storage space, and use up to 1,4 megabytes of heap (runtime) memory. In the Series 40 phones (e.g. the Nokia 7210 phone) the size of a midlet suite is limited to 64 kilobytes and the available heap size for an applet is 200 kilobytes.[26]

In the presented system I managed to keep the size of the midlets reasonable. The size of the RDF engine is 6 kilobytes (inside a jar package). The user interface

[26] The memory limitation information were taken from specifications for Java phones in Forum Nokia, http://www.forum.nokia.com/ .

component of the ontology browser / editor takes another 6 kilobytes. The size of the whole client-end midlet package is 41 kilobytes.

6.1.2 API limitations

The Java MIDP (Mobile Information Device Profile) has got only a limited set of Java API packages. Ready-made metadata toolkits (e.g. the Jena Framework) can be used on the server-side, but for the terminal we needed to create a lighter solution.

Two important APIs that are lacking from the MIDP 1.0 specification are 1) an API for the location information and 2) a camera control API. There is a Java Mobile Media API[27] available that allows camera control in Java midlets. It is implemented in the Nokia 3650 phone, which will be available in early 2003. There is also a JSR (Java Specification Request) for Location API for J2ME[28].

In the MIDP 1.0 specification (and even in the MIDP 2.0), there is no support for XML and web services. There is a Java Specification Request for J2ME Web Services Specification (JSR 172), which should include XML support. Despite the fact that there are open source XML parsers for J2ME available, our prototype uses the N-Triple format to keep the size of the application small.

6.1.3 Limitations of the prototype

The search criteria are limited to a single categorization and the geographical distance from the current location to the target. Adding multiple categorizations would be quite easy, but the user would need to browse one categorization at a time. In this case, a search summary page showing the different search criteria

[27] See http://java.sun.com/products/mmapi/

[28] See http://jcp.org/en/jsr/detail?id=179

could be useful. The page could even contain an option of selecting different locations and the distance limit for the location-based search. On the other hand, these changes would increase the complexity of the user interface.

As noted in the Section 5.4, obtaining the location information is simulated by using point-and-click on an on-screen map. In the future, this could be replaced by location services provided by the operators. Another possibility could have been to use the numeric cell ID (CID) and the location area code (LAC) of the mobile network, which are available in the Nokia 7650 API for native applications, but not for Java midlets. This would have allowed displaying items that are inside the current cell or the current area on the network of a single operator. User profile and terminal profile are not yet fully used in the prototype system, but there is a place for them in the architecture.

Context privacy is not yet managed in the prototype. The user is not questioned for willingness to post the location data. According to the current practices and regulations in Finland, the location information should be used only if the user gives permission for using it. In the system, the user should be notified that his current location is stored and published in the service, together with his unique ID, the current time, and the photograph.

The MIDP version of the client implementation lacks the support for location information and maps. This is because there is no API for accessing web services in the current MIDP Java implementations. This could be easily circumvented with creating a servlet interface to the current location and map services.

The presented prototype uses pre-defined URLs for connecting to the tourist portal. If the system was used in many countries at different sites, service discovery mechanisms should be used for connecting a local portal instead of a central one.

6.2 Evaluation and comparison

6.2.1 Speed and cost of data transfer and location services

The speed of data transfer and the connection delays make a significant part of the usability of the mobile services. The cost of the services is another essential issue. The estimations in this section are based on the GSM/GPRS services of the Finnish mobile phone operators in the end of the year 2002.

Most of the current GSM networks with GPRS support have a maximum transfer rate of 13.4 kbit/s per timeslot. A mobile terminal with GPRS multislot class 6 can use combinations up to 2+2 timeslots and 3+1 timeslots. This gives us a maximum transfer rate of 26.8 – 40.2 kbit/s for downloading and 13.4 – 26.8 kbit/s for uploading. At these speeds, uploading a 40-kilobyte JPEG image (with the maximum 640x480 pixel resolution of the Nokia 7650 phone) would take 12 – 24 seconds, and downloading a 5-kilobyte image (160x120 pixels) that fits to the screen would take 1 – 1,5 seconds for the transfer plus the time for initiating the connection.

In Finland, most operators have a monthly fee for GPRS subscription and use also the amount of transmitted data as a basis for invoicing. If the amount of monthly-transmitted data is 5 megabytes, the average price for a megabyte stays between 2,60€ and 4,20€ per megabyte[29]. In this case, uploading one JPEG image of size 640x480 and 40 kilobytes would cost 0,10 – 0,16€. Some of the operators offer GPRS service with a monthly fee only, so increasing the amount of transmitted data would decrease the price per megabyte.

[29] Calculations are based on price information from four Finnish GSM operators in October 2002. The monthly fees for a basic GSM subscription (about 3–5€ per month) are not included in the calculations.

Location-based SMS-services in Finland cost currently (December 2002) about 0,60€ for a query, which is approximately the same as the prices of other commercial SMS-services. It is hard to estimate, how much would the location information cost for a service like the tourist portal of the prototype.

6.2.2 Usability

A field research from the year 2000 shows, that searching information by using WAP on a mobile phone (Nokia 7110e and Ericsson R320s in the research) can be cumbersome, slow, and costly [Nie00]. Some of the reasons for this were 1) delay in establishing connection, 2) the price of circuit-switched GSM data, 3) small screen size in mobile phones 4) slowness of textual input with the mobile phone keypad, and 5) design/implementation failures in WAP services.

Packet-switched GPRS connections have been at least a partial cure for the first two problems. The smart phone category mobile phones have bigger screen sizes (e.g. 176x208 pixels in Nokia 7650) than the phones that were used in the research (96x65 in Nokia 7110e), and the bigger screen size helps navigating. The T9-dictionary-based text input has improved the writing speed. There are still delays in navigating the menus of a WAP service, when the phone needs to fetch new WAP pages from the network.

By using location information and computer-understandable metadata we can possibly overcome some of the problems in the mobile information retrieval. The number of network requests can be made smaller, and the user input can be minimized.

The procedures for finding a fish restaurant in different SMS and WAP services and the prototype are presented in Table 1. The user actions in the table describe a shortest relevant path of actions that I could find for (in some cases partly) accomplishing the task. The price comparisons should be considered suggestive, and the prices can differ from operator to another. The future price of the location information for applications like our prototype is still an open question, and the price of using the prototype system cannot be accurately estimated. Usability tests or expert reviews for the user interfaces were not performed, but they would help in finding problems with the user interface.

The content annotation and sharing on the mobile phone, which is another half of this prototype, is not possible in the other presented services. The other services (except the WWW search engine) have only content that is fed to the system by one of the administrators. For example, the yellow pages services show only advertisements of the services of paying customers.

Table 1. Comparison of different forms of mobile browsing in a "Find a nice fish restaurant" use case with the Nokia 7650 phone. The time and cost estimations are approximate and not tested on real users.

Service	User actions and time estimation	Costs for the user	Other
Location-based SMS (On Radiolinja and Sonera GSM networks in Finland)	Need to know the SMS service number and the keyword in advance 1. Start sending a message 2. Type the number of the SMS service 3. Type SMS message ("FIND RESTAURANT"), 4. Wait for an answer. As response, get the address and name for the two nearest ones. 5. If these are not enough, send another message ("FIND MORE") to get the two of the next closest restaurants ~ 1 min for the nearest two restaurants	Service SMS: **0,59€–0,66€** / two matches	Gives only the two closest alternatives; limited selection of service types. The user needs to know how to get to the specific address. Works on nearly any GSM phone.
GPRS WAP aktivist.fi service	1. Open the WAP browser 2. Browse the operator's service menu to find the service (or open the service from bookmark) 3. Browse through title pages 4. Select "City info" 5. Select "Helsinki" 6. Select "Restaurants" 7. Select "Fish restaurants" 8. Select one of the restaurants 9. If the address is nearby (need to know the address!), ok, otherwise return to step 8. ~ 2–3 min for finding the list of restaurants	Service 0,49€ (only for the first time) Data ~0,07€ (24kB) Overall costs **0,56€**	If there are many results, the user needs to check one by one, which are near, which far away. The user needs to know how to get to the specific address, and know from the address if it is nearby.

Service	User actions and time estimation	Costs for the user	Other
GPRS WAP Finnish Yellow pages service	1. Open the WAP browser 2. Browse the operator's service menu to find the service (or open the service from a bookmark) 3. Wait for title page to disappear 4. Type keywords into field 5. Type city ("Helsinki") 6. Type current street address 7. Select FIND 8. The service lists the names of 20 nearest services and their addresses. Select one of the matches 9. Select "Map" to display a zoomable map that displays the location of the selected restaurant 10. Click "call" to invoke a voice call to the restaurant to make table reservations. ~ 2–3 min for finding the restaurants	Service 0,55€ / search Data ~0,10€ (35 kB) Overall costs: **0,65€**	The user needs to type in the current city and address. Downside: level of categorization – Found no match in the whole Helsinki for "Fish restaurant" ("Kalaravintola"), another search was needed with the keyword "Restaurant". The user can get a map showing the restaurant.
GPRS WWW using the ReqWireless WebViewer and Google	1. Open the web browser 2. Pick Google from the bookmarks 3. Type "Fish restaurant Helsinki" Google returns with a list of hits 4. Select the first hit – link to activist.fi city guide, class "Fish restaurants" in Helsinki. The activist.fi service returns a list of fish restaurants and their addresses. 5. Select a link and get detailed description and images of the selected restaurant. ~ 3–4 min for finding the list of restaurants	Service: free Data: 0,30€–0,60€ (~100–300 kB, with images) Overall costs: **0,30€–0,60€**	The service is designed for desktop use and is more difficult to use on the mobile phone.
GPRS context-aware browser (The prototype presented in this thesis)	1. Launch the tourist browser application 2. Select category using the ontology browser: Restaurants → Fish 3. Select "Results" from the menu. The system returns a list of fish restaurants within the surrounding 30 km. 4. Highlight on one of the items and select "display" to see a photo and the description of the restaurant. 5. Select "Map" from the menu to see a map with the destination and the current location	Service: free? Location information: 0,10€–0,50€ (an estimation) Data: 0,10€–0,30€ (~30–100 kB) Overall cost: **0,20€–0,80€**	Finds only items that are fed into the service. The items can be added using a mobile phone. The overall cost depends on the price of location information and the pricing of the portal service.

6.2.3 Performance

All the server-side components, including location (map) services, location-based queries, and image transformation, require a lot of resources. The server side is neither optimized nor performance-tested. There are solutions for better performance location-based search available.

N-Triple was used in our prototype for transmitting RDF-metadata between the server and the terminal. Using N-Triple saved application space, since we did not need to include an XML parser in the midlets. On the other hand, N-Triple is not the most size-optimized format for transmitting metadata. Using it for transmitting big amounts of metadata (e.g. more than 10 statements) requires more space than using the XML format. In RDF/XML, each subject needs to be transmitted only once. The use of namespaces saves also a lot of bytes. Currently the use of RDF/XML would eat a little of application space, but if an XML-parser was included in the MIDP API, the RDF/XML format would be a better option. Compression would also help in reducing the size of transmitted ontologies.

6.2.4 Other issues

The presented system would help in creating shareable, mobile-accessible services with a repository of resources. Storing the context information raises an issue of privacy. An example of privacy violation could be a service that stores the unique identifiers of the surrounding persons together with the location and time information, and makes all of them publicly available together with the photo. It would be fairly easy for unknown parties to gather person registers that store a location of a certain person at a certain time.

The system should also filter out illegal (e.g. copyrighted) or inappropriate content. The filtering process could include automatic parts and human moderators.

6.3 Towards the future

6.3.1 Future work

There are different ways to continue the development of the system. One user interface issue would be adding a free-text search possibility to selecting the category. Alternative visual representations for ontologies should be considered. We could try out tree-views for ontologies on a mobile terminal, and implement more complex editing of ontologies on a mobile phone.

User interface testing would help in getting more objective results on the pros and cons of the presented search and annotation methods. In the user interface tests, our system could be compared with the traditional WAP or WWW browsing using a mobile phone, and the existing mobile-accessible image repositories.

Automatic image classification, automatic high-level context detection, and different annotation methods, such as speech annotation could be feasible further in the future. These fields of research should be monitored while the computing capabilities in the mobile phones develop further.

The use of multiple facets, multiple ontologies or semantically richer ontologies should be researched further. This way we could perform queries like “find timetables for ferries to a nearby island”. The system should also include a way of expressing multiple photographs or pieces of information about the same target, e.g. a restaurant, its menus and wine lists, and photos from the street and inside. Linking to web pages and different services could be a very usable feature.

Since the server part of the system is not optimized, the server end should be performance tested and optimized on the most crucial parts. The real implementations of the Location API, the Web Services API, and the Camera API should naturally be used, when they become available for the terminals.

6.3.2 Towards the mobile semantic web

Automatic and easy manual semantic annotation of content at the creation time could help in creating content for the semantic web[30], in addition to managing a personal collection of images or creating shareable tourist portals. However, even richer annotation methods could be used. These could include user interface solutions for annotation with multiple ontologies, and easier semantic linking between two resources or between two ontology elements in different ontologies. The prototype shows that semantic annotation can be made on the field during the creation of the content, and it does not even require big efforts from the user. The use of context and semantically annotated services could help in creating user-friendly compositions of mobile services.

6.4 SWOT analysis

If we consider the future enhancement ideas that were described above, we can imagine an enhanced and polished production version of the system, customized to be a tourist information service. A SWOT analysis for using such a system as a tourist information portal is presented in Table 2. Most of the presented weaknesses depend on the current implementation and could be fixed in the future. I think that the success of such a system would depend at least on the real added value it provides to the users, and the technical maturity of the system, and the market trends.

[30] Visions for the semantic web are presented in [BHL01].

Table 2. A SWOT analysis of an imaginary production version of the presented system, if it was customized to be a tourist information service.

Strengths	**Weaknesses**
• Only a small amount of textual user input required in annotation of the images • The use of context information together with classification allows simple and efficient search without textual input • The presented system is feasible in the current smart phone class mobile devices; it is simple and light-weight • The ability of entering user-created content to the service with rich annotation by using only the mobile phone is a major advantage over the existing systems	• Currently no compatibility with the present systems (WWW search engines, WWW pages) • The ontology browser user interface can be too complex / too unfamiliar for some users • Uploading content to the service costs – this can limit the popularity of the service • Different users can classify the items differently • Language barriers were not considered in the current implementation
Opportunities	**Threats**
• Mobile virtual communities could be formed from the users of the system • Other, cheaper forms of positioning could be used • Linking to WWW or WAP resources would be easy to add • Automatic image classification and annotation by speech could be used in the future for minimizing the efforts of annotation • The service could be internationalized by adding translations of the captions of classes for other spoken languages	• The possibility of misuse can require too much effort from the service provider • The users might not want to annotate images manually • The required amount of user input during annotation might be too much • Gaining a critical mass of users and content could be difficult if the location information costs too much

7 Conclusions

The existing tools for managing metadata, vocabularies, and ontologies were designed for the desktop computers. Especially ontology editors and browsers use a lot of space on the computer screen. The programming tools for using metadata languages, e.g. RDF engines, require too much memory to be used in mobile phone applications on the client side.

In this thesis we presented a system that supports ontology- and context-based annotation, sharing, and information retrieval in the mobile environment. The system uses the existing RDF metadata language in Java MIDP applications for mobile phones. The presented prototype includes a complete system for using ontologies in sharing user-created content with mobile phones.

The presented solution suggested that the creation time context, such as location and time, should be automatically stored as metadata for the content. The content and the metadata can be stored on a server to be shared with other users. The location information together with a simple classification can be usable search criteria for many services. The tourism scenario works as an example of how these technologies can be usefully combined.

Automatic image classification, image recognition, and speech annotation could help in even easier creation of the metadata in the future. In a public service, the different pieces of the context that are stored should be carefully considered in order to protect the privacy of the users and the surrounding persons.

The presented system was compared with the current WAP and SMS applications. The concept allows creating portals with content that can be created, published, searched, and used with mobile phones. In the future, the methods of the presented system could be used for combining different services under a single context-aware user interface.

References

AAH97 G. D. Abowd, C. G. Atkeson, J. Hong, S. Long, R. Kooper, M. Pinkerton: Cyberguide: a mobile context-aware tour guide. *Wireless Networks* 3, no. 5 (October 1997), 421–433. `http://portal.acm.org/citation.cfm?doid=272186.272199`

Bec02 D. Beckett (ed), *RDF/XML Syntax Specification (Revised).* W3C Working Draft 8-November-2002. `http://www.w3.org/TR/2002/WD-rdf-syntax-grammar-20021108` [19.1.2002]

BHL01 T. Berners-Lee, J. Hendler, and O. Lassila, The semantic web. *Scientific American* 284(5):34-45 (May 2001). `http://www.sciam.com/2001/0501issue/0501berners-lee.html` [31.10.2002]

BiM01 P.V. Biron, A. Malhotra (eds), *XML Schema Part 2: Datatypes.* W3C Recommendation 02 May 2001. `http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/` [27.2.2003]

BrG00 D. Brickley, R.V. Guha (eds), *Resource Description Framework (RDF) Schema Specification 1.0*, W3C Candidate Recommendation 27-March-2000. `http://www.w3.org/TR/2000/CR-rdf-schema-20000327/` [20.5.2002]

BrG02 D. Brickley, R.V. Guha (eds), *RDF Vocabulary Description Language 1.0: RDF Schema*, W3C Working Draft 12-November-2002. `http://www.w3.org/TR/2002/WD-rdf-schema-20021112/` [19.1.2003]

Bri02 B. McBride, Jena: Implementing the RDF model and syntax specification, `http://www-uk.hpl.hp.com/people/bwm/papers/20001221-paper/` [16.10.2002]

Cra02 L. Cranor, M. Langheinrich, M. Marchiori, M. Presler-Marshall, J. Reagle (eds), *The Platform for Privacy Preferences 1.0 (P3P1.0) Specification*, W3C Recommendation 16-April-2002. `http://www.w3.org/TR/2002/REC-P3P-20020416/` [20.1.2003]

ChK00 G. Chen, D. Kotz, *A Survey of Context-Aware Mobile Computing Research*, Dept. of Computer Science, Dartmouth College, TR2000-381. `http://citeseer.nj.nec.com/390713.html` [2.12.2002]

CIH02 G. Colyer, K.Ishii, J. Hunter, *Mapping between Dublin Core and JPX (JPEG 2000) Metadata*, ISO/IEC JTC1/SC29/WG1 N2736, 15 October, 2002. `http://www.jpeg.org/metadata/wg1n2736.pdf` [26.2.2003]

CHH01 D. Connolly, F. van Harmelen, I. Horrocks, D.L. McGuinness, P.F. Patel-Schneider, L.A. Stein (eds), *DAML+OIL (March 2001) Reference Description*, W3C Note 18-December-2001. `http://www.w3.org/TR/daml+oil-reference` [21.5.2002]

CDM00 K. Cheverst N. Davies K. Mitchell and A. Friday: Experiences of developing and deploying a context-aware tourist guide: the GUIDE project. *Proc. of MobiCom '00*, ACM 2000, pp. 20-31, `http://doi.acm.org/10.1145/345910.345916` [19.1.2003]

DCH02 M. Dean, D. Connolly, F. van Harmelen, J. Hendler, I. Horrocks, D.L. McGuinness, P.F. Patel-Schneider, L.A. Stein, *OWL Web Ontology Language 1.0 Reference*, W3C Working Draft 12-November-2002. http://www.w3.org/TR/2002/WD-owl-ref-20021112/ [16.12.2002]

DIG01 Digital Imaging Group (DIG35). *DIG35 Specification: Metadata for Digital Images*, Version 1.1, June 18, 2001. http://www.i3a.org/i_dig35.html

Haa01 M. Haarala, *Gathering and Managing Information for Centralized User Profiles for Utilisation in Third Generation Mobile Services*, 2001, Master's thesis, University of Helsinki, Dept. of Computer Science.

Hay02 P. Hayes (ed), *RDF Semantics*, W3C Working Draft 12-November-2002. http://www.w3.org/TR/2002/WD-rdf-mt-20021112/ [27.2.2003]

HHV02 E. Hyvönen, P. Harjula, K. Viljanen, Representing metadata about web resources. *Proc. Of Semantic Web Kick-Off in Finland*, HIIT Publications, Helsinki, Finland, 2002, pp. 47–75. http://www.cs.helsinki.fi/u/eahyvone/stes/semanticweb/kick-off/proceedings.html

HIR02 K. Henricksen, J. Indulska, and A. Rakotonirainy, Modeling context information in pervasive computing systems. *Proc. of Pervasive 2002*, Zürich, Switzerland, August 26-28, 2002. http://link.springer.de/link/service/series/0558/tocs/t2414.htm.

HVH02 E. Hyvönen, K. Viljanen, A. J. Hätinen, Yellow pages on the semantic web. *Proc. of XML Finland* 2002, HIIT Publications. http://www.cs.helsinki.fi/u/eahyvone/xmlfinland2002/ProceedingsXML2002-final.pdf [20.1.2003]

HSS02 E. Hyvönen, A. Styrman, S. Saarela, Ontology-based image retrieval. *Proc. Of XML Finland* 2002, HIIT Publications. http://www.cs.helsinki.fi/u/eahyvone/xmlfinland2002/ProceedingsXML2002-final.pdf [20.1.2003]

ISO13250 ISO/IEC 13250:2000 *Information technology – SGML applications – Topic maps*

KAC02 G. Karvounarakis, S. Alexaki, V. Christophides, D. Plexousakis, M. Scholl, RQL: A declarative query language for RDF, in *Proc. of WWW2002,* Honolulu, Hawaii, USA, 2002. http://doi.acm.org/10.1145/511446.511524 [14.1.2003]

Kiv01 J. Kivinen, *Puheesta digitaalikuvan metadataksi* (in Finnish), a report from Helsinki University of Technology, 2001. http://www.media.hut.fi/GTTS/Suomi/dt&raportit/j_kivinen_2001.pdf [2.12.2002]

KRW02 G. Klyne, F. Reynolds, C. Woodrow, H. Ohto, M.H. Butler (eds), *Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies*, W3C Working Draft (8-November-2002). http://www.w3.org/TR/CCPP-struct-vocab/. [18.10.2002]

KCM03 G. Klyne, J. J. Carroll, B. McBride (eds), *Resource Description Framework (RDF):Concepts and Abstract Syntax.* W3C Working Draft 23-January-2003. http://www.w3.org/TR/2003/WD-rdf-concepts-20030123/ [26.2.2003]

LaB02 Y. Lafon, B. Bos, *Describing and Retrieving Photos using RDF and HTTP*. W3C Note (19-April-2002), http://www.w3.org/TR/2002/NOTE-photo-rdf-20020419 [21.5.2002]

LaS99 O. Lassila, R. Swick (eds), *Resource Description Framework (RDF) Model and Syntax Specification*, W3C Recommendation (22-February-1999). http://www.w3.org/TR/1999/REC-rdf-syntax-19990222 [22.1.2003]

MaM02 F. Manola, E. Miller (eds), *RDF Primer*. W3C Working Draft (11-November-2002), http://www.w3.org/TR/2002/WD-rdf-primer-20021111/ [10.1.2003]

MaS00 N. Marmasse, C. Schmandt, Location-aware information delivery with ComMotion. *Proc. Int. Symposium on Handheld and Ubiquitous Computing, HUC* 2000, 157-171, Bristol, UK, September 2000. http://citeseer.nj.nec.com/marmasse00locationaware.html [20.1.2003]

McH03 D. L. McGuinness, F. van Harmelen (eds), *Web Ontology Language (OWL): Overview*, W3C Working Draft 10 February 2003, http://www.w3.org/TR/2003/WD-owl-features-20030210/ [25.2.2003]

MSR02 L. Miller, A. Seaborne, A. Reggiori, Three Implementations of SquishQL, a Simple RDF Query Language. *Proc. Of 1st International Semantic Web Conference (ISWC2002), Sardinia, Italia, June 9-12th, 2002*

Nie00 J. Nielsen, *WAP field study findings*, December 2000. Online: http://www.useit.com/alertbox/20001210.html [15.1.2003]

Nok02 *Mobile Web Services Interfaces*, Nokia white paper, 2002. http://nds1.nokia.com/press/pdfs/Mobile_Web_new_A4.pdf. [16.10.2002]

PaS02 P. Patel-Schneider, J. Siméon, The Yin/Yang Web, XML Syntax and RDF Semantics. *Proc. Of WWW* 2002, Honolulu, Hawaii, USA, 2002. http://doi.acm.org/10.1145/511446.511504 [19.1.2003]

Pep00 S. Pepper, The TAO of Topic Maps. *XML Europe* 2000, Paris, France, June 2000. http://www.gca.org/papers/xmleurope2000/papers/s11-01.html [30.10.2002]

PeM01 S. Pepper, G. Moore, *XML Topic Maps (XTM) 1.0*, TopicMaps.org Specification 06-August-2001. http://www.topicmaps.org/xtm/1.0/xtm1-20010806.html [22.1.2002]

RSH02 V. Raatikka, K. Salminen, E. Hyvönen, XML, RDF(S) and Topic Map Databases, in *Semantic Web Kick-Off in Finland*, HIIT Publications, Helsinki, Finland, 2002, pp. 77–109. http://www.cs.helsinki.fi/u/eahyvone/stes/semanticweb/kick-off/proceedings.html

Rad02 *Metropolis Location Application Interface (LAIF).*A brochure from Radiolinja in Finnish, available from http://www.radiolinja.fi/ [27.9.2002]

RBK98 H.A. Rowley, S. Baluja, T. Kanade, Neural network-based face recognition, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol 20, no.1 , pp. 23-38, Jan. 1998

Saa02 J. Saarela, Semantic Information Router (SIR), in *Semantic Web Kick-Off in Finland*, HIIT Publications, Helsinki, 2002, pp. 257–263. http://www.cs.helsinki.fi/u/eahyvone/stes/semanticweb/kick-off/proceedings.html

SAW99 B. Schilit, N. Adams, R. Want, Context-aware computing applications. In *Proceedings of IEEE Workshop on Mobile Computing Systems and Applications*, pages 85–90, Santa Cruz, California, December 1999. IEEE Computer Society Press.

Sea02 A. Seaborne, An RDF NetAPI, *Proc. of ISWC 2002*, pp. 399–403, Springer 2002. http://link.springer.de/link/service/series/0558/bibs/2342/23420399.htm [2.12.2002]

SiH02 P. Silvonen, E. Hyvönen, Semantic web tools, in *Semantic Web Kick-Off in Finland*, HIIT Publications, Helsinki, Finland, 2002, pp. 137–152. http://www.cs.helsinki.fi/u/eahyvone/stes/semanticweb/kick-off/proceedings.html

SiD02 M. Sintek, S. Decker, TRIPLE — A query, inference, and transformation language for the semantic web, *Proc. Of International Semantic Web Conference (ISWC)*, Sardinia, June 2002. http://triple.semanticweb.org/iswc2002/TripleReport.pdf [14.1.2003]

SMV02 M.K. Smith, D. McGuinness, R. Volz, C. Welty (eds), *Web Ontology Language (OWL) Guide Version 1.0*, W3C Working Draft 4 November 2002, http://www.w3.org/TR/2002/WD-owl-guide-20021104/ [19.1.2003]

Son02 Instructions for using the Sonera SMS "Missä"-service, available in Finnish from http://www.sonera.fi/ [27.9.2002]

StL02 A. Stent, A. Loui, Using event segmentation to improve indexing of consumer Photographs, *Proc. Of ACM SIGIR* 2001. http://doi.acm.org/10.1145/383952.383960. [30.10.2002]

SzP98 M. Szummer, R.W. Picard, Indoor-outdoor image classification, In *IEEE Int. Work. On Content-based Access of Image and Video Databases*, January 1998.

Tho02 J. Thoméré, K. Barker, V. Chaudhri, P. Clark, M. Eriksen, S. Mishra, B. Porter, A. Rodriguez, A web-based ontology browsing and editing system. *Proc. of Artificial Intelligence* 2002, pp. 927-934.

TBM01 H.S. Thompson, D. Beech, M. Maloney, N. Mendelsohn (eds), *XML Schema Part 1: Structure*, W3C Recommendation 2-May-2001. http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/ [27.2.2003]

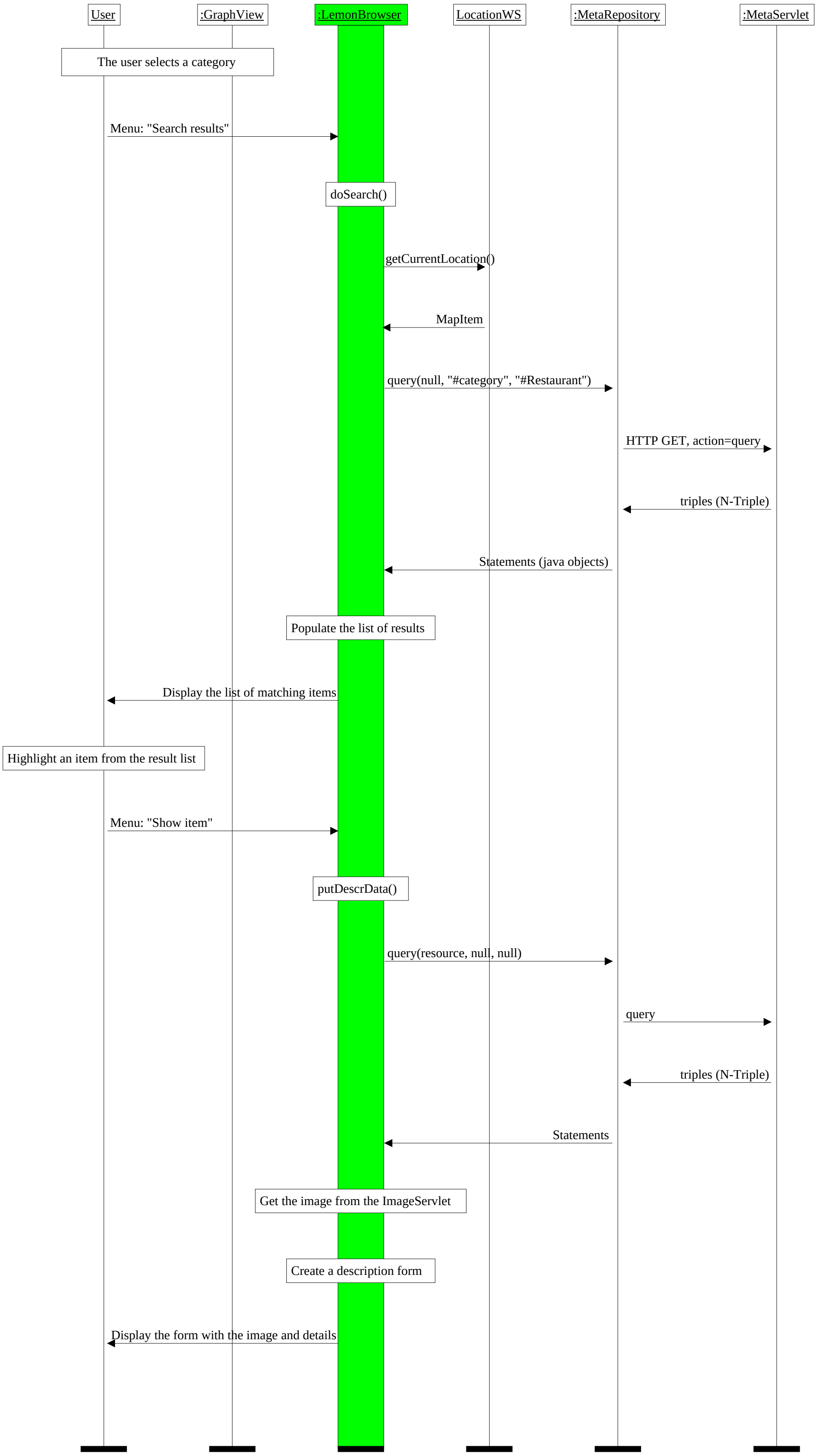
Tuu00 E. Tuulari, *Context aware hand-held devices*. Espoo, VTT Electronics, 2000. VTT Publications; 412 ISBN 951-38-5563-5; 951-38-5564-3. http://www.inf.vtt.fi/pdf/publications/2000/P412.pdf

VFJ01 A. Vailaya, M. A. T. Figueiredo, A.K. Jain, H.-J. Zhang, Image classification for content-based indexing, in *IEEE Transactions on Image Processing*, vol. 10, no. 1, pp. 117—130, Jan. 2001.

Vak02 M. Vakkilainen, *Käytettävyystestaus: Puhe digitaalikuvan metadatan lähteenä* (in Finnish), a report from Helsinki University of Technology, 2002. http://www.media.hut.fi/GTTS/Suomi/dt&raportit/kaytettavyysanalyysi.pdf [20.1.2003]

Vod02 *Vodafone signs Siemens as global location enabling server supplier*, A press release from Vodafone, 12 March 2002. http://www.vodafone.com/ [20.1.2003]

Appendix A. Sequence diagram for searching


User
:GraphView
:LemonBrowser
LocationWS
:MetaRepository
:MetaServlet
The user selects a category
Menu: "Search results"
doSearch()
getCurrentLocation()
MapItem
query(null, "#category", "#Restaurant")
HTTP GET, action=query
triples (N-Triple)
Statements (java objects)
Populate the list of results
Display the list of matching items
Highlight an item from the result list
Menu: "Show item"
putDescrData()
query(resource, null, null)
query
triples (N-Triple)
Statements
Get the image from the ImageServlet
Create a description form
Display the form with the image and details