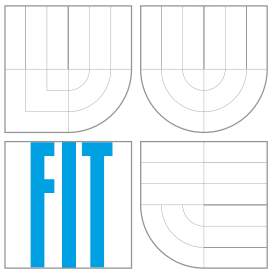
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

DIGITÁLNÍ KNIHOVNA PRO MOBILNÍ ZAŘÍZENÍ

DIGITAL LIBRARY FOR MOBILE DEVICES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — VOJTĚCH ROBOTKA
AUTHOR

VEDOUCÍ PRÁCE — doc. RNDr. SMRŽ PAVEL, Ph.D.
SUPERVISOR

BRNO 2011

Abstrakt

V posledních letech začíná s mírou využívání digitálních knihoven a obsahu v nich narůstat i potřeba zjednodušit přístup k těmto datům a umožnit uživatelům vyhledávat napříč co největším množstvím knihoven a uspokojit tak jejich potřeby po tak důležitých informacích. Obsahem této práce bylo seznámit se se situací a nedostatky přístupu k obsahu v digitálních knihovnách a na základě těchto informací pak navrhnout a implementovat aplikace pro mobilní platformy Android a iOS (iPad a iPhone), které by využily potenciál mobilních zařízení s těmito systémy.

Abstract

With exponential growing amount of the content stored in digital libraries, there is a need to make this content easy accessible and allow the users to search the documents across many repositories to satisfy their needs for information. The aim of this work is to investigate the state-of-the-art of accessing digital content and identify the gap in the currently used solutions. Then design a digital library system and implement applications for the mobile platforms Android and iOS (iPad and iPhone), that will utilize the potential of the mobile devices with these systems.

Klíčová slova

digitální, knihovna, mobilní, zařízení, Android, iOS, iPad, iPhone

Keywords

digital, libraries, mobile, device, Android, iOS, iPad, iPhone

Citace

Vojtěch Robotka: Digital library for mobile devices, bakalářská práce, Brno, FIT VUT v Brně, 2011

Digital library for mobile devices

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana doc. RNDr. Pavla Smrže, Ph.D. Další informace a zkušenosti mi poskytl Ing. Petr Knoth. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

. . . . . .
Vojtěch Robotka
May 18, 2011

Poděkování

Rád bych poděkoval Ing. Petru Knothovi za pomoc při vypracování této práce. Během práce na projektu mi předal mnoho cenných rad a zkušenosti, které mi umožnily bakalářskou práci dokončit a ovlivnily její podobu.

Contents

List of Figures

Chapter 1

Introduction

In the world of digital libraries, many important steps have been done in the last years. Many important approaches for manipulating with the metadata and content stored in digital libraries have been realised and set by the Open Archives Initiative[11], including standard protocol for Metadata harvesting (OAI-PMH[9]) and standards for the description and exchange of aggregations of Web resources[10]. It is known that digital libraries can be used for much more than simple documents or research papers. New approaches focus on developing ways how to handle in digital libraries with audio and video data, images, theses, maps or any other primary data. Nowadays, the content is growing every year and there is a need to continue in improving the accessibility of digital content. The content stored in digital libraries is essential for scholars to stay in touch with their research field. Researchers should know work of other researchers related to their topic as much as they can. Digital libraries were developed for these purposes and the content in digital libraries is growing almost exponentially(Figure 1.1). But how to search and navigate in more than 30 million records of content in digital libraries? This is a crucial task of this field for many companies and academic groups, like for example Google (Google Scholar) or Biefield University (BASE[16]). The production of the content is as important as easy way of accessing this content. People searching for a research article, do not want to search from more web pages or learn how to use all these tools for accessing digital content. If they already find some article they are interested in, they will want to read it or even download it. If the search engine does not allow them to perform these operations, their needs will not be satisfied and they will have to try another search engine. These situations reduce accessing and using digital libraries, since researchers do not want to spend hours by searching articles on the web.

Now, imagine a life of a researcher or academic teacher. They probably spend much less time at one place or in their office with PC than any other employee. They travel to conferences, present their work at universities or just work on a research requiring working on different places with different people. For these situations they need to make their resources mobile. New market with many different mobile devices offers a significant potential for improving the way in which they access information. They could read, present and work with their resources anywhere. However this could very improve productivity of some researchers or make their lives a bit easier, nowadays, they have no suitable mobile application for these purposes. Usually, the only way how to access the content in digital libraries is by using a web browser. This solution does not usually allow them to save the content to device for reading without need of broadband connection or it could be also difficult to operate with these search engine through internet browser in mobile devices, since the websites are

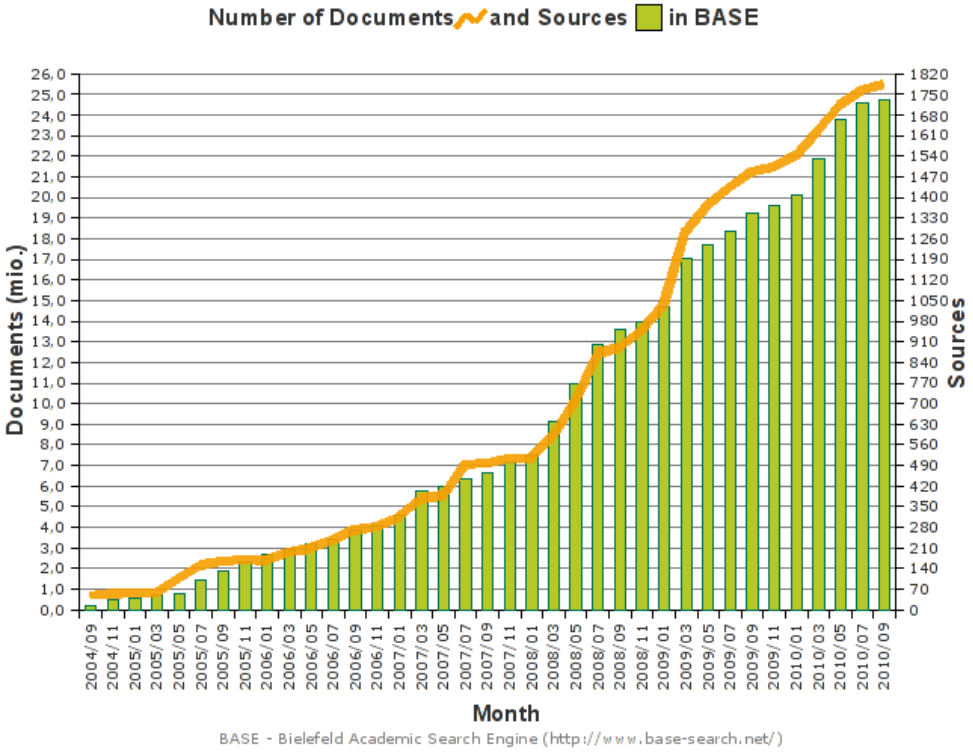


Figure 1.1: BASE content grow statistics

usually not optimized for mobile access.

The goal of this work is to fill this communication gap between the world of digital content stored in digital libraries and the world of mobile devices that are getting more popular than ever before. This should help help many researchers that are often out of their office or traveling abroad without broadband connection to get better access to information. With connection to the CORE Server we want to offer standalone tool for different associations or universities as well as one online system connected to many open access repositories for accessing huge amount of digital content at one time.

In this section we present the motivation for this work and description of often used key terms related to this work. Connection between this work and the CORE Project will be explained afterwards continued with the main goals delimited for this project. In Section 2, other work related to our project will be discussed and compared with our approaches. Then process of designing our application with reasons about chosen solutions will be explained in Section 3. In Section 4, the results of the work will be described further. Finally, the evaluation of our goals compared to results will be presented in Section 5.

1.1 Key terms

1.1.1 Digital libraries

Digital library (DL) is a web based system for storing any kind of texts, documents or articles. They have become an important aspect of most (if not all) academic projects and have consequently been widely utilised in the academic environment. Nowadays, every

research paper, article or learning document, appears online, stored usually in some digital library. Since there are many digital libraries on the world wide web, it can be very difficult to find the desired document.

1.1.2 Open Access

The term *Open Access* describes the way of accessing content in a digital library, however many repositories are covered by subscription. This means only the people that pay subscription fee can access the content in these libraries. Fortunately, there are more than 1,800 quality checked Open Access repositories worldwide [7] storing more than 30 million records. Search engines, such as Google Scholar or Vascoda, do not differentiate between the Open Access and subscription-based content. Therefore, the users information needs are only satisfied if the resulting links from scientific searches lead to full-text versions of articles that are covered by a subscription or follow the Open Access policies.

1.1.3 Related content

Users, searching only for Open Access content, have to submit their queries to a number of relevant Open Access repositories or to use systems that harvest metadata from multiple sources, such as BASE [16] or OAIster [6]. However, these metasearch systems do not currently provide well-grounded information about semantically related content and do not provide services to digital repositories that would allow them to improve their browsing capabilities. To improve the navigation across the different articles or repositories, semantically related content should be described by its similarities or dependencies in standard format (usually RDF). For example if the author of an article is described in RDF format, another service can easily recognize this information and find some more information about the author.

1.1.4 Mobile access

Digital libraries store the huge amount of the content and contain the most of important information about the world researches. On the other hand, the growing market with new mobile devices offers great opportunities for accessing digital repositories. People would not be dependent on their computers when they need to find or read articles from a digital library. Although this is a very simple goal, there is no solution, which allows owners of mobile devices to access digital libraries. Nowadays the only possibility to access the content from digital libraries on mobile devices is accessing the articles from built-in internet browser. In this case, it is usually difficult to download a full-text document and users are also limited to a few digital repositories. This means, they sometimes have to search from many different web pages till they find what they want.

1.2 The CORE Project

This work is associated to a project named CORE. The project presents a method for improving the access to content and navigation between semantically related items across Open Access repositories. Our approach is based on the use of automatic link generation algorithms that are applied to discover relations between full-text content. The relationships are represented and published as Linked Data and can be queried using a set of web services. Making information about related resources publicly available in an interoperable

format is an essential step in order to allow its exploitation by various third-party applications. In addition, we demonstrate the usability of the CORE service on two results. First, a demonstration client for Open Access repositories is developed to be used by institutions that administer repositories. The client is implemented as a widget that complements standard search services of Open Access repositories by providing dynamic links to semantically similar content stored in other repositories. As the second result, we support ubiquitous access to information stored across Open Access repositories using mobile technology. With the current quick spread of smart phones and tablet devices, this application area has a lot of potential for digital library technology. The following work describes the second part of the project - developing applications to provide access to digital repositories for users with mobile devices.

1.3 Project goals

At the beginning of the project there was an idea to create an environment for accessing digital texts on the internet of any focus as simply as possible. There was also a need to provide the interface for mobile devices, that are getting more exploit than ever before. This can make the access to digital libraries much easier since people are not obliged to use their PCs or laptops to search articles on the internet.

The main goals of the project:

1. Improve the way of searching articles in digital libraries
2. Provide a simple interface for most widely used mobile platforms (Android, iPhone, iPad)
3. Access the documents directly from the application
4. Personalization
   (a) Save documents (also download pdf document)
   (b) Save search results
   (c) Recommend new articles (based on search history and saved documents)

1.3.1 Realtime search

Search documents in realtime is one of the axiomatic features of our application. To achieve very fast searching, it is necessary to use indexing technique. This technique have some requirements to work properly. The indexed structure of the harvested data should be placed on the same machine where the search engine is running. The harvested content requires also large storage. The best way, how to solve this problem, is to run the search engine and the indexer on the separate server and access it remotely from mobile device.

1.3.2 Mobile interfaces

Mobile clients are part of the CORE system and the main topic of this work. While the CORE Server is on the background of the system, hidden for users and doing all the hard work like harvesting, downloading content, indexing etc., the mobile clients are the foreground applications, that provide interface to the CORE Server for students and researchers.

Interfaces should be very easy to use, but provide all the functionality useful for users. They should respect common forms and the ways of using mobile applications, to allow users work with them easily without a need of reading user's manual.

1.3.3 Access full-text documents

Information, stored in digital libraries are usually based on meta-data only and exclude the full text of a document. They can hold information about the full-text document location on the internet. Access to the full-text document can be also limited for subscribed users only. This situation can be very discouraging for users, that spend their time searching many articles, but ultimately are not successful. People searching for the articles on the internet do not want to access the meta-data, instead they want to read the whole article. That is the reason why we want to give them the access to the full-text documents directly from the CORE Mobile applications.

1.3.4 Personalization

Mobile devices are usually very personal. This gives the potential for mobile applications to also be well-personalized. Like everybody wants to have their favorite music in their device, they might want to save the documents and the articles of their interests. There are many options to improve user's experience using a mobile application. We chose these features to achieve user-friendly personalized application:

Saving documents

This feature may look like very common functionality, which is available in many applications, like internet browsers. Actually, for example iPad users are currently unable to download the content from web browser and store it in their device. This is one of the reasons why we can not develop the CORE client like web-based application. This feature is important, since people working in research or universities have to travel a lot and might want to access their articles without need of internet connection.

Saving search results

Users sometimes make a search, which can lead to very relative results, according to their interests. In this case, we want to allow them to save these results to read them later. Users could also want to save all their results (without need to save them manually), so we can allow them to enable this feature in settings, if they choose this option.

Recommendations

Users usually search for similar topics about their research focus or studies. For experts in some research field, it is important to stay up-to-date. They will be interested if some new similar article about their topic appears in the library. Unfortunately, in this case, they will not find this article until they make proper search leading to it. But this can do their application for them, since it can remember his searching and determine their topics of interest. Then, when the user opens the application, it can do a search to find new articles of his interests for him and serve the results him like recommendations. This can help him to have up to date overview about the research.

Chapter 2

Related work

After setting the main goals of the project, we tried to find out all related solutions that focus on at least some of our goals. We found many ready standalone tools offering interesting features for private use as well as web-based services searching in huge amount of content for public use. Most of these solutions are open-source or provide free web interface for accessing their content. We selected most interesting solutions, according to our goals and depicted a few of their characteristic features and functionalities in Figure 2.1. In the following sections most relevant features will be discussed further.

2.1 Library federations and harvesting

Federated Digital Libraries (DLs) can act as (a) standalone systems, working on a locally collected set of harvested data (typically acquired using the OAI-PMH [9] protocol), or (b) as real-time federations by distributing query messages to external digital libraries [3]. While standalone systems have full control over their structures (and typically also higher data storage and computational requirements), it is evident that distributed DLs lack significant part of the full picture, be it either the control over the information retrieval facilities, such as ranking algorithms, full-text content for indexing and other internal knowledge of the host. Since the first specification of the OAI-PMH protocol [9] in 2001 a number of tools for OAI-PMH metadata harvesting or tools relying on OAI-PMH harvesting have been developed. These include (a) Open Source tools for metadata harvesting, (b) web-portals providing cross-repository metasearch services and (c) publishing platforms capable of integrating and exposing information in an OAI-PMH compliant way.

2.1.1 Standalone tools

There are many solutions available for federating multiple digital libraries to use as a standalone tools in universities or other academic environments. Most of them are Open Source, so they are easily customizable and extendable. Open Source harvesting tools, such as MOAI [5] and D-NET [13] aim to deliver a software solution for metadata harvesting that can be easily integrated into complex and specialised library systems. MOAI [5] can also transform federated metadata to new collections and serve them through OAI-PMH protocol, so it can be used as another source or separate module to use in combination with some other applications. Another system, OJAX [19] is concerned with the development of a user interface for a federation of OAI-PMH repositories.

	MOAI[5]	OJAX[19]	BASE[16]	OAIster[6]	DNet[13]	CORE
Metadata aggregation (oai-pmh)	✓	✓	✓	✓	✓	✓
Full-text harvesting					✓	✓
Standalone tool	✓	✓			✓	
Web-based search/metasearch		✓	✓	✓		✓
Web-service access	✓		✓		✓	✓
Mobile access						✓
Linked data						✓

Table 2.1: Characteristic features/functionalities of systems using the OAI-PMH protocol and the features/functionalities offered by the CORE system

2.1.2 Web-based applications

On the other hand, web-portals, such as BASE [16] and WorldCat (the OAIster database [6]) provide metasearch over more than 25 million records large collections directly to end-users. In addition, BASE allows the use of a programmable API for the searching of their collection. The last group of systems consist of complex publishing platforms that, among other functionalities, are capable of exposing the library content using OAI-PMH.

2.2 Full-text downloading

Though there is a number of tools for the metadata harvesting only a few perform full-text downloading.[1] Thus most systems, including the mentioned large metasearch engines BASE and OAIster, are completely reliant on the provided metadata and their information retrieval functionalities are limited. Furthermore, it is difficult to build on top of these services applications that can automatically generate richer metadata using natural language processing methods to help better organize library information.

2.3 Personalized search

Described solutions focus on the federated search and consolidating data through the OAI-PMH protocol. In the case of OJAX using data indexing technique, which helps to provide results very quickly. But none of these solutions care about the relevancy of found results, you can get many irrelevant results and pass some articles you might be interested in. To provide better results, there is a technique called personalized search. This technique is based on users profile, which consists of their focus and recent topics of interest. There are many different approaches how to determine users profile. Many authors focus on users browse history, actions, browsed documents and past search queries, mostly made in internet browser [12]. This approach meets many benefits for application in an e-commerce Web site [15]. But in the case of reading articles from digital libraries, the interests are usually based

[1] To the best of our knowledge the only system that offers the full-text download functionality is DNet [13]

on studies or work focus, which can differ from users interests of shopping or other internet browsing.

2.4 Mobile access

Full-text information could also be better utilized to support access from mobile devices. Smart phones and tablet devices offer high potential for improving accessibility of digital media[18]. The current attempts to create mobile device applications for digital libraries [1] are mostly concerned with the development and the optimization of user interfaces for mobile internet browsers [19, 14]. However, native mobile applications can currently offer a more user-friendly solution with better and more natural functionality. For example, web browsers on iPhone and iPad devices (at the time of writing) do not support the downloading of full-text documents. Providing a service capable of harvesting both metadata and full-text content plus delivering it to mobile applications through a native application could enable mobile users to access their documents from anywhere more easily.

Chapter 3

Design

3.1 Architecture

This section describes the architecture of the CORE system (Figure 3.1) and describes how information is collected, processed and exposed for further use. There are three main stages in which the system handles data. In the **first stage**, the metadata records and full-text documents are harvested from available Open Access repositories (represented by the Metadata Harvester and Full-text Downloader components of the system - Figure 3.1). In the **second stage**, the available content is indexed for better searching and processed in order to discover meaningful relations between papers using automatic link generation methods (represented by the Semantic Relation Analyser component - Figure 3.1). In the **third stage**, the extracted relations are exposed as Linked Data and are represented in the CORE Triple store and through XML-RPC protocol provided to mobile clients. Semantic relations were not the main focus of my work, so the second and the third stage will only be partly described further.

3.2 Content harvesting

3.2.1 Metadata harvesting

The first component of the CORE system is responsible for acquiring (1) metadata records and (2) the associated full-text content from Open Access repositories. The harvesting of the metadata is performed using standard OAI-PMH requests to the repositories. Successful requests return an XML document which contains information about the papers stored in a repository. Although the OAI-PMH protocol itself is not directly concerned with the downloading of full-text content, as it focuses on the transfer of metadata, a good practise in repositories (which is unfortunately not consistently applied) is to provide as a part of the metadata the URLs to the full-text documents. Document URLs can be thus extracted and used to automatically download full-texts from repositories over the HTTP protocol. The CORE system provides this functionality and is optimized for regular metadata harvesting and full-text downloading of large amounts of content. The fact that CORE caches the actual full-text content in order to process the documents and to discover additional metadata distinguishes this approach from a number of other Open Access federated search systems, such as BASE or OAISTER, that rely only on the metadata accessible through OAI-PMH.

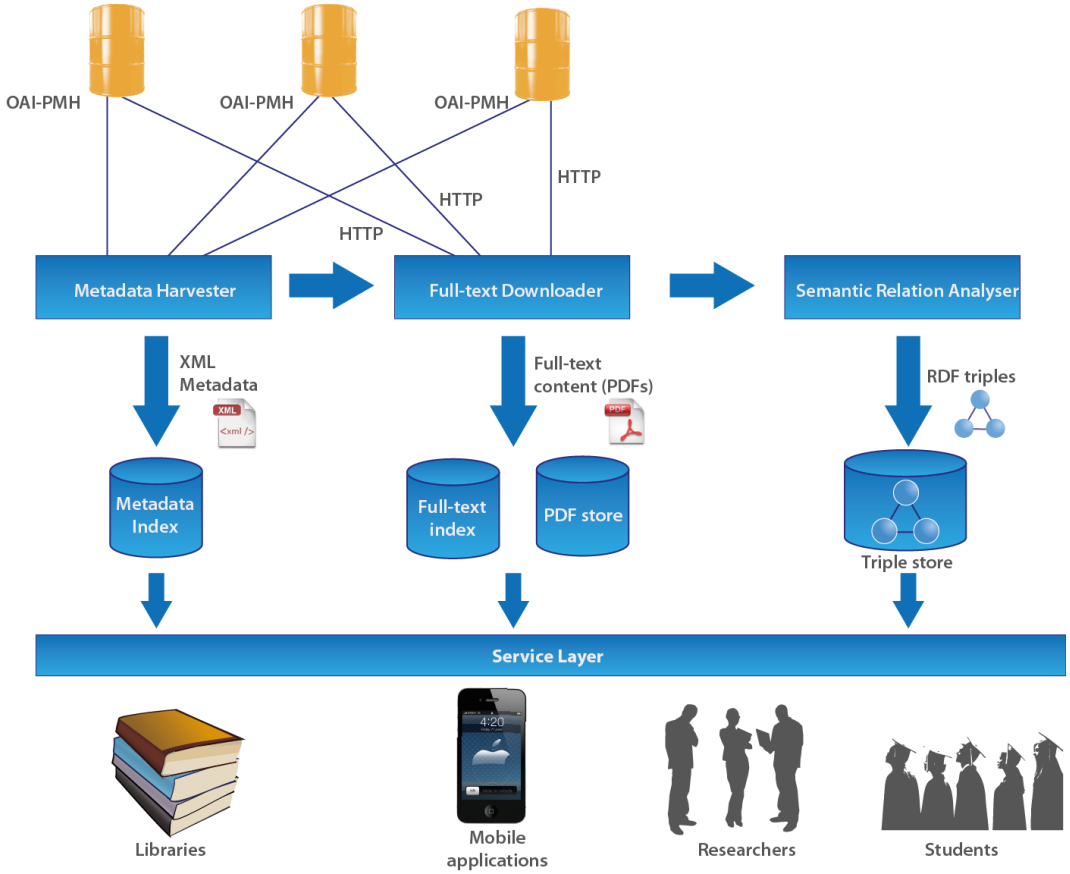


Figure 3.1: CORE architecture

3.2.2 Fulltext downloading

At the time of writing, the CORE harvesting system has been tested on 142 Open Access repositories from the UK. Table 4.1 provides preliminary statistics on the amount of content harvested from UK repositories. We expect to extend the number of repositories in the future to all available repositories listed in OpenDOAR. A larger number of repositories will increase the demand on the storage space as in our case not only metadata, but also full-text has to be stored as opposed to services, such as OAIster or BASE. To provide a rough estimate, the Open University's repository Open Research Online (ORO) contains more than 16,000 records out of which about 5,000 records are accompanied by full-texts. Storing the full-text records requires about $5GB$ of storage space. There are currently over 30 million metadata records stored across Open Access repositories worldwide. It is expected that about 10% of the records contain links to downloadable pdfs, which would account for about 3 million full-text papers stored across Open Access repositories worldwide. By generalizing from the ORO data we can estimate that about $3TB$ of space are required to cache all these documents. Though the real space needed will be larger due to the index and metadata, this amount can still be relatively easily managed and the size does not prevent us from extending the system to the remaining repositories listed in OpenDOAR (more than $1,600$ non-UK repositories).

3.3 Search engine and indexing

When the data and metadata are downloaded from remote repositories, another module starts indexing them. This technique rapidly speeds the searching, since the data is sorted in a structure optimized for fast manipulation. For indexing we use the Apache Lucene module, which is probably the most used indexing module. Manipulation with the data in the index is similar to databases. We can specify the fields that should be stored to the index and we can use easy queries to read the data from the index. The main difference is in the way, that the data is stored. In addition to metadata, the full-text pdfs are also indexed. The Lucene module selects only a few keywords from the full-text document, so it does not store all the common words, that do not say anything about the document.

3.4 Mobile access

According to our goals, there was a need to make applications for different mobile platforms. As a result, we had to implement similar functionally, but programmatically very different applications. Platform iOS for iPad and iPhone devices is based on the Objective C language, while the Android applications are usually based on Java. Therefore our effort was to move the most of functionality on the CORE Server and make the mobile applications as easy as possible. This can also make implementation much easier. The applications should also be very versatile for different input data and for API changes. If we decide to make some changes in the future, we would not change the mobile applications, but only the CORE Server. Also users would not be obliged to update their applications after some minor changes.

3.4.1 XML-RPC server

XML-RPC server (in Java terminology *XML-RPC servlet*) is a part of the CORE server, exposing the data and providing the search functionality through the XML-RPC protocol. It is basically extension of the HTTP protocol with standard syntax and data formatting. The CORE Server is based on the Apache server, which supports deploying different modules (Servlets). XML-RPC servlet is connected to the metadata and the full-text index and uses the Apache Lucene module for searching the articles.

3.4.2 Fulltext server access

Another module in the CORE Server is a Fulltext download Servlet (based on simple HTTP Servlet). It processes the requests from the mobile (or any other) devices for the fulltext documents that are stored in the CORE Server. Proper document is determined by the identifier of the selected document and its original repository (For example: `{id: 7452; repository: 'oro.open.ac.uk'}`. For the possibility of the bigger pdf files, it is essential to send the content using buffer, to stabilize the downloading process of the document.

3.4.3 Data flow

Since the CORE System consists of more application modules running on different platforms, it is important to define standards and select protocols for exchanging data between these modules. Figure 3.2 describes the processes with data transforming between the modules of the CORE System. We can split this data flows into the follpwomg two groups:

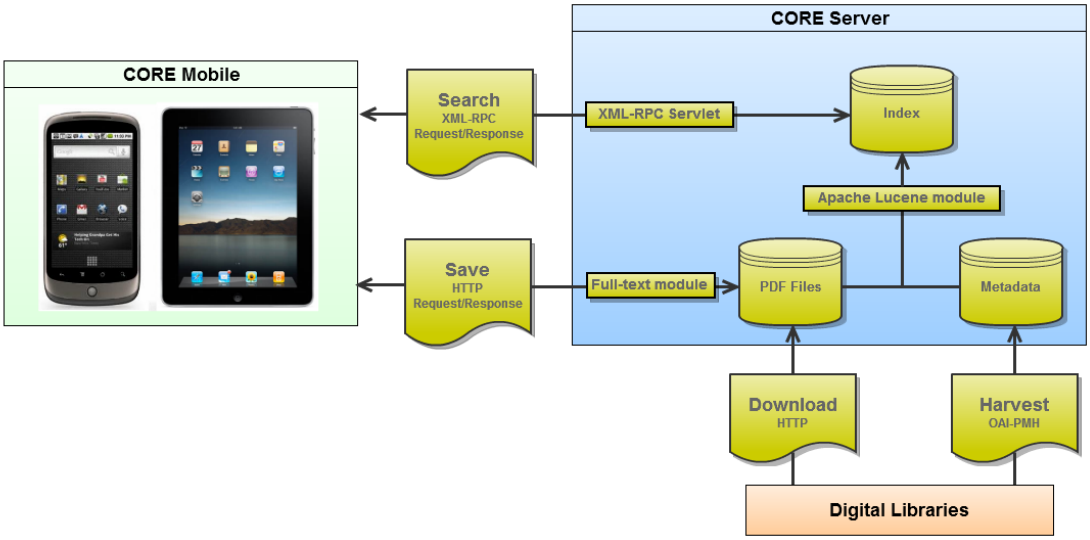


Figure 3.2: Data flow model with protocol description.

External data flows

The CORE Server retrieves data from many digital libraries across the World Wide Web. It uses OAI-PMH standard for metadata harvesting, which is supported by most digital libraries on the internet[7]. The OAI-PMH protocol is built on HTTP protocol. The request for retrieving metadata is usually based on HTTP Get method, but it HTTP Post method can be used also. If the request is successful, the library returns the HTTP Response with XML file encoded by OAI-PMH standards. The metadata are usually represented in Dublin Core format [8]. Example of OAI-PMH response shows the Figure 3.3.

Internal data flows

The CORE System consists of 3 main applications. The CORE Server and the CORE Mobile for Android and iOS platforms. Main communication is realized by the XML-RPC protocol, also built on the HTTP protocol. When the user submits a search, the CORE Mobile application makes the XML-RPC request and send it on the CORE Server using the HTTP Post method. The server performs the search on the index and builds the XML-RPC response message. This protocol uses standardized format for the standard data types like the basic types (number, string, boolean) and also advanced data types (structure, hash, array). In figures 3.4 and 3.5 you can see the examples of the XML-RPC messages (method call and method response), transfering between the CORE Server and CORE Mobile application.

```
<OAI-PMH xmlns="http://www.openarchives.org/OAI/2.0/"
     xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/OAI-PMH.xsd">
  <responseDate>2011-05-06T09:57:02Z</responseDate>
  <request verb="ListMetadataFormats">http://oro.open.ac.uk/cgi/oai2</request>
  <ListMetadataFormats>
    <metadataFormat>
      <metadataPrefix>context_object</metadataPrefix>
      <schema>http://www.openurl.info/registry/docs/info:ofi/fmt:xml:xsd:ctx</schema>
      <metadataNamespace>info:ofi/fmt:xml:xsd:ctx</metadataNamespace>
    </metadataFormat>
    <metadataFormat>
      <metadataPrefix>oai_dc</metadataPrefix>
      <schema>http://www.openarchives.org/OAI/2.0/oai_dc.xsd</schema>
      <metadataNamespace>http://www.openarchives.org/OAI/2.0/oai_dc/
      </metadataNamespace>
    </metadataFormat>
    <metadataFormat>
      <metadataPrefix>rdf</metadataPrefix>
      <schema>http://www.openarchives.org/OAI/2.0/rdf.xsd</schema>
      <metadataNamespace>http://www.w3.org/1999/02/22-rdf-syntax-ns#
      </metadataNamespace>
    </metadataFormat>
    <metadataFormat>
      <metadataPrefix>rem_atom</metadataPrefix>
      <schema>http://exyus.com/xcs/tasklist/source/?f=put_atom.xsd</schema>
      <metadataNamespace>http://www.w3.org/2005/Atom</metadataNamespace>
    </metadataFormat>
    <metadataFormat>
      <metadataPrefix>uketd_dc</metadataPrefix>
      <schema>http://naca.central.cranfield.ac.uk/ethos-oai/2.0/uketd_dc.xsd</schema>
      <metadataNamespace>http://naca.central.cranfield.ac.uk/ethos-oai/2.0/
      </metadataNamespace>
    </metadataFormat>
  </ListMetadataFormats>
</OAI-PMH>
```

Figure 3.3: Example of OAI-PMH response

3.5 Mobile applications

If we want to help people accessing digital content, the design of applications and their interfaces is very important as well as the whole process of searching and getting the document from the library. The application has to put all the functions together to make the interface user-friendly and intuitive. The user should be able to use the application without the need to read a user-manual or documentation.

3.5.1 Use cases

Searching articles

The main use case of CORE Mobile applications is searching articles. Users should specify the words that they want to search and also they can determine, where the words can appear, whether in the full-text document, title, author etc. When the user specifies the search keywords and their appearance, after submitting the query to server, the CORE system processes the query and returns the results as quickly as possible.

```
<?xml version='1.0' ?>
<methodCall>
    <methodName>LuceneSearcher.getHitsArray</methodName>
    <params>
        <param>
            <value>
                <string>title</string>
            </value>
        </param>
        <param>
            <value>
                <string>modelling</string>
            </value>
        </param>
    </params>
</methodCall>
```

Figure 3.4: Example of XML-RPC method call

Reading and saving articles

When the CORE Server returns the search results, the mobile client parse the results to a list. The results should be very clear to read, but they should also show all the information that could help users to decide which record they might be interested in. When the user selects any record, the record details open and the user can read all the information about the article. They also have the options to save the document and to find similar articles.

Settings

Every application needs some settings to adapt different conditions and environment. When user install the application, the settings is preset by default values. These default values should meet the most used settings, so many users do not have to change the settings at all.

3.5.2 The CORE Mobile GUI

Very crucial task in designing user-friendly application is user interface design (GUI - Graphical User Interface). Main use cases should be accessible as easy as possible. Actions should be intuitive and should meet common user habits (this means common activities like searching or changing settings should be similar to other applications).

Since the screen on mobile devices, in particular on cell phones, is much smaller than common PC screen, we had to respect this fact in designing CORE Mobile interface. Users with specific screen sizes, that were not tested in application development process, usually can be disappointed with the interface, since some components could not display properly and it could be difficult for the users to perform some actions. To avoid these situations, we designed universal interfaces and tested them on multiple screen sizes. The Android system offers very good framework, when designer can make multiple different interfaces for specific screen sizes and the Android system chooses the interface which suits best to users screen. Since this application should be used both tablet devices and cell phones, it is very useful feature. These approach is presented in figure 3.9 (interface for Android tablet device) and figure 3.8 (interface for Android phones).

```
<?xml version="1.0" encoding="UTF-8"?>
<methodResponse>
    <params>
        <param>
            <value>
                <array>
                    <data>
                        <value>101</value>
                        <value>Symbolic modelling</value>
                        <value>2005-02-10</value>
                        <value></value>
                        <value>empty</value>
                        <value>Oxford University Press</value>
                        <value>10</value>
                        <value>empty</value>
                        <value>empty</value>
                        <value>oro.open.ac.uk</value>
                        <value>empty</value>
                        <value>Modelling the dynamics of industry populations</value>
                        <value>2001-07</value>
                        <value></value>
                        <value>2164</value>
                        <value>empty</value>
                        <value>10</value>
                        <value>empty</value>
                        <value>empty</value>
                        <value>oro.open.ac.uk</value>
                        ...
                        ...
                        ...
                    </data>
                </array>
            </value>
        </param>
    </params>
</methodResponse>
```

Figure 3.5: Example of XML-RPC method response

3.5.3 Accessing CORE Server

As mentioned before, the CORE Mobile clients are mobile interfaces that use functionality of the CORE Server. The CORE Server implements all the functionalities for harvesting the content from digital libraries and searching articles. Communication between the CORE Mobile application and the CORE Server is implemented using remote function calls through the XML-RPC protocol. The mobile application creates a text file containing the information which function should be called and parameters related to a specific function. The text is structured into an XML file and send to the CORE Server. The server application calls the function with given parameters and sends the result structured to the XML files back. The XML-RPC protocol has standard format for manipulating with data types, so it the CORE Server could communicate with any application sending data in the XML-RPC format.

While the functions for searching articles are implemented through the XML-RPC protocol, downloading the articles is done using the HTTP protocol. Each record in results contains the information about the name of full-text file stored in the CORE Server. When the user selects the record he want to download, the HTTP Request is created and sent to the CORE Server - Downloader module. Then the module tries to find the article in

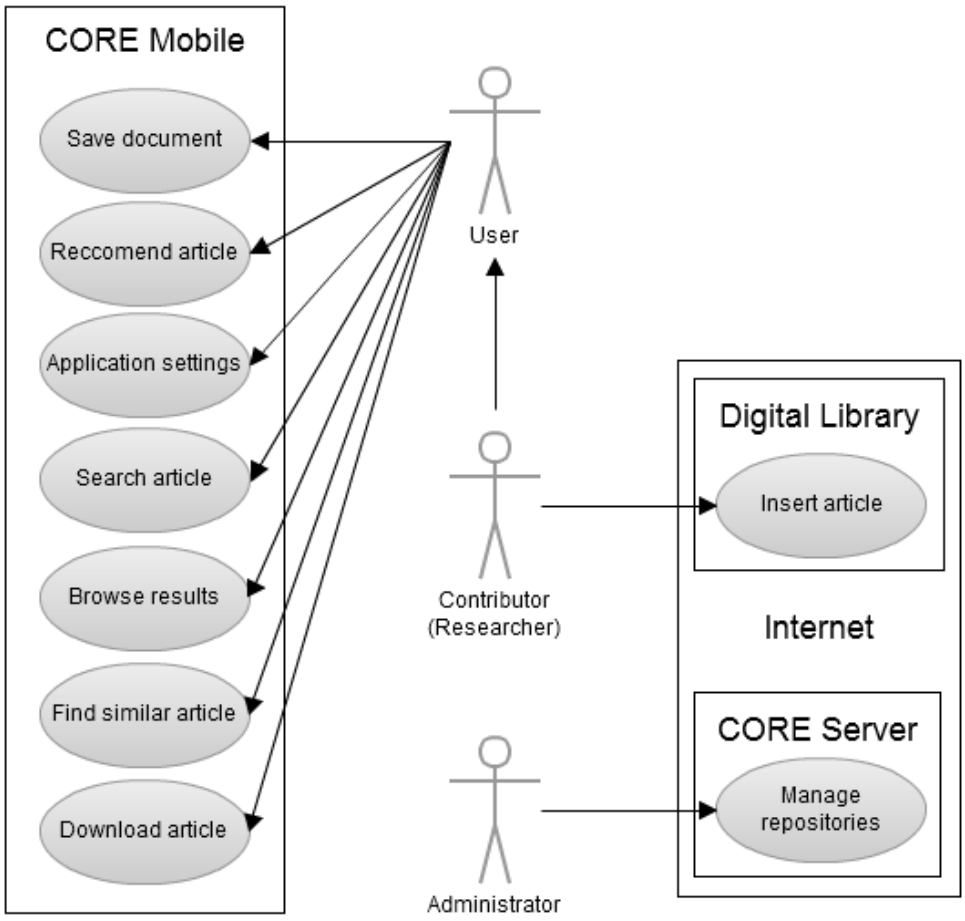


Figure 3.6: Mobile application: **Use case diagram** - describes the use cases and relations between actions in CORE Mobile and CORE Server

its storage and send it as an HTTP Response back to client. The client stores the file to external storage and saves the information about its location.

3.5.4 Storing user data

Since the CORE Mobile application is meant to personalise users needs and interests, it has to store the data to the device. We can divide this process to two groups – first for the information without direct user access and second for the data accessible directly from the device without need of using the CORE Mobile application. First group contains information about users search history, metadata about saved files and application settings. In the second group will mostly appear saved documents (full-text pdf files).

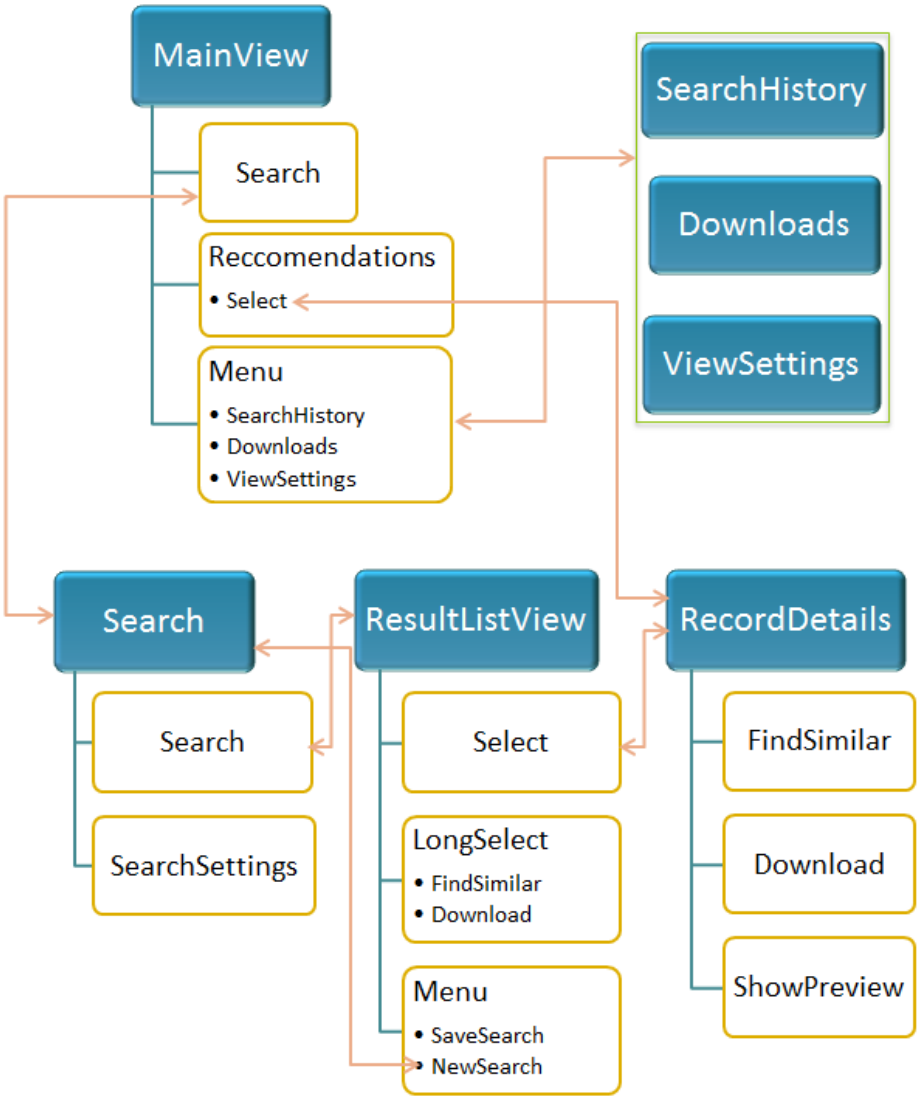


Figure 3.7: Mobile application: **Graphical User Interface design** - describes the *Views* (blue boxes) and *Actions* (yellow boxes) with their relations

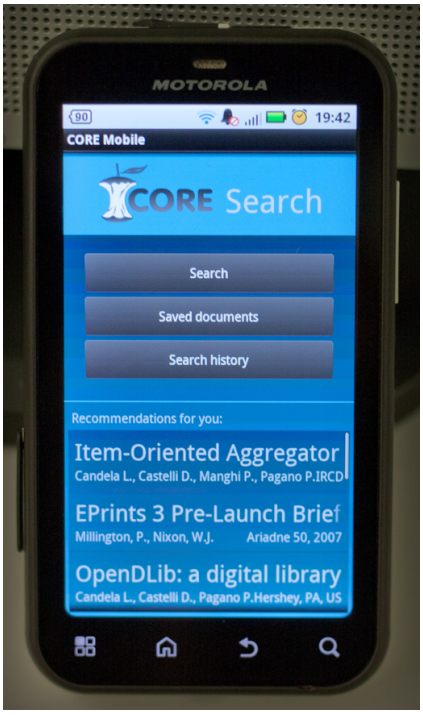


Figure 3.8: Mobile application: **Interface for Android phones** - the interface for small screens on the phones has to be clear and simple. The main contains less components.

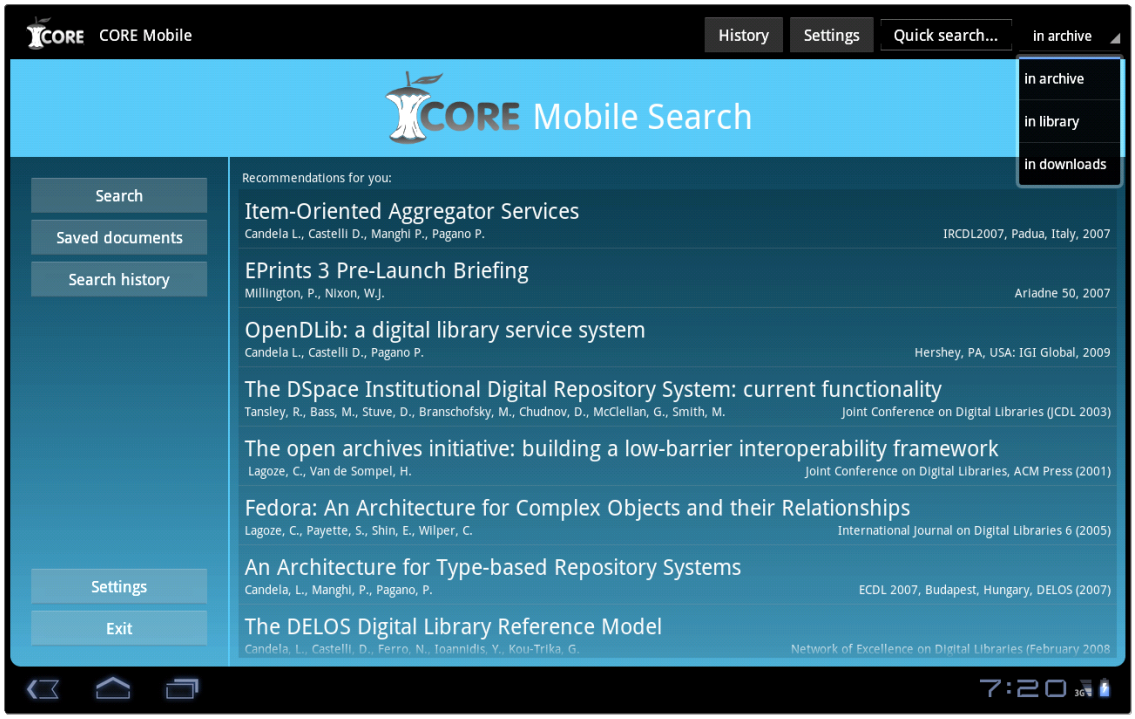


Figure 3.9: Mobile application: **Interface for Android tablets** - due to bigger screen can offer more functionality and components on a screen.

Chapter 4

Results

In this chapter final results of the work and mobile applications will be presented. All parts of the system will be further described as well as used methods and solutions with explanation of reasons of their usage.

Target platforms

The main goal of this work was to develop applications for easy accessing digital library system from mobile devices. Due to reasons discussed in section 3 the server application maintaining the background processes had to be implemented first. For this purpose, we decided to use desktop server with Apache Tomcat Server. This platform is very modular and flexible and fits for our objectives very well.

In case of mobile applications, we had to choose from many various mobile platforms that are being used. Despite the statistics of the market with mobile devices, we decided to focus on two most used mobile platforms for smart devices. The first, open-source platform with huge community support, deep software development kit and very well-documented APIs is the Android platform. On the other hand, the second choice, iOS platform, offers also great potential in spread of users amount. The iPad was also the first common-used tablet device on the market and nowadays it is still a leader on the tablet market. However there are many other platforms, these two have about 75% of recent acquired smartphones in USA (Figure 4.1).

4.1 CORE Server

The CORE Server (part of the CORE Server related to mobile applications) is an application meant to run on dedicated computer. The base of the application is Apache Tomcat Server. It is standalone Java application that allows deploying a number of Java modules (Servlets). The Apache Tomcat Server takes care about the access from the internet, communication between modules and selects which module should be run, according to defined mappings. The CORE Server runs many modules for harvesting, downloading full-text documents, accessing XML-RPC functions, providing downloaded documents, etc. In this section, modules related to the mobile applications (XML-RPC module and Full-text access module) will be discussed further.

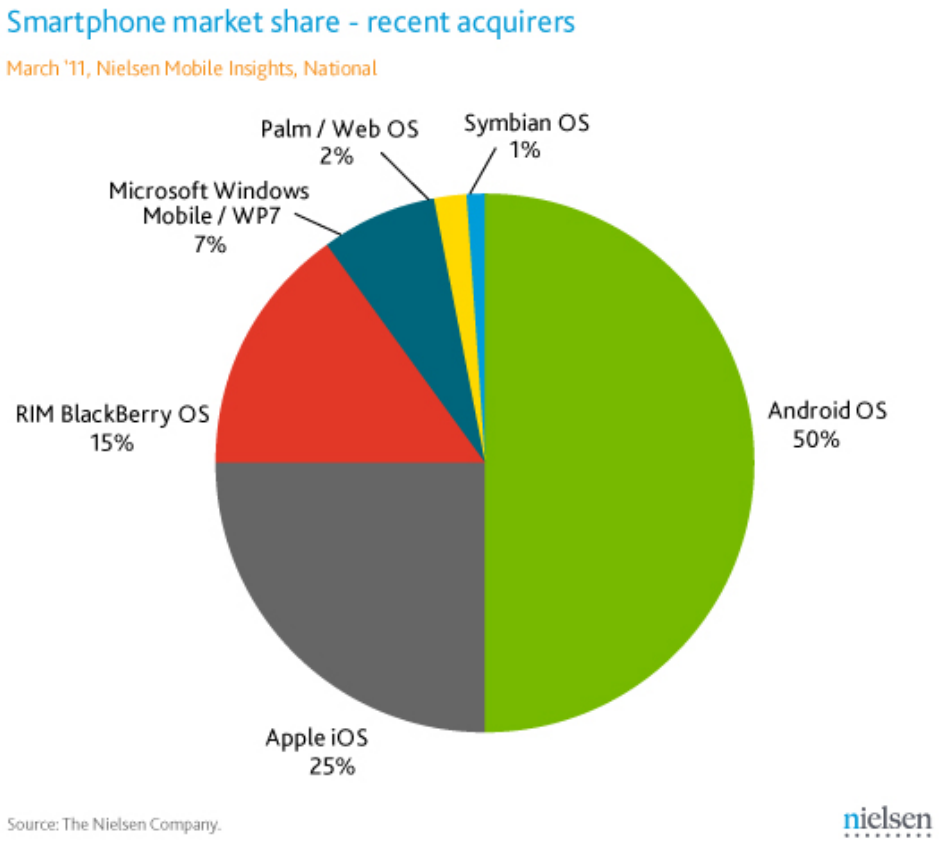


Figure 4.1: Smartphone OS market share from March 2011. According to Nielsen [4] statistics, iOS with Android have about 75% of the market.

4.1.1 XML-RPC module

is mapped on URL address `http://core-server-address/XmlRpcServlet` (by default) and receives remote calls from mobile applications. The used module is `XmlRpcServlet` from Java package `org.apache.xmlrpc.webserver`. In properties of the module, we can define, which functions will be accepted and how to process them. In the case of the CORE Server, we implemented class named `LuceneSearcher` which contains main functions for searching articles and digital content. As mentioned before, we use Lucene Indexer module for indexing aggregated content. When the XML-RPC module gets the search request from the mobile device, it calls the function `getHitsArray()` of module `LuceneSearcher` and the module accesses this index for searching aggregated articles. The module contains some more functions for example for different result structures (`getHitsMultiArray, getHitsStruct`) or for searching similar articles.

4.1.2 Full-text access module

is part of the CORE Server, deployed as a module of Apache Tomcat Server. The module is implemented as a Servlet derived from `javax.servlet.http.HttpServlet`. It uses map-

ping with parameters, when some parts of the URL are used to determine variables of the following process. The module has the following mappings:

```
http://<core-server-address>/Downloader
http://<core-server-address>/Downloader/*
```

The first mapping can be used for the `HTTP POST` method when the parameters are send in the `POST` message. The second mapping says that every request beginning with this path and continuing other characters will be handled by this module. The following URL can be then used to access a specific PDF on the server:

```
http://<core-server-address>/Downloader/oro.open.ac.uk/2134
```

The request will be handled by the `Downloader` module. The last two parts of the URL specify the repository and the ID of the document we want to download.

4.2 Android client

Although developing an Android application may seem to be similar to developing a desktop application, Android framework is quite different to other platforms, even those are are also based on Java language. Android uses *activities* to split single parts of the application. Every activity consists of a class, which is defined by a simple Activity class or it can be diferentiated for its special purposes, like browsing list (`ListActivity`), setting preferences (`PreferenceActivity`), watching images (`ImageActivity`), and many others. These special activities help the developer with programming, since they have many common tasks already implemented. The behaviour of the activity is to be implemented by overloading function `onCreate()`, which is called imediatelly after creating activity. There are also other methods for setting the behaviour usually handling the activity life cycle (switching to other activity, resuming activity, changing screen rotation, etc.).

4.2.1 User interface

In Android platform, developers define user interface in external XML file. Android framework allows them to define multiple resource files for different mobile devices. Developer can create filesystem with folder naming according to specific needs of every device and environment. Android then chooses the resource that fits the best to the device. For example, developer can define two different user interfaces for one activity for different screen rotations. The first definitions will put to folder `/res/layout` and the second file with the same name will put into `/red/layout-land`. Android then chooses the proper interface according to screen rotation. This technique was used to handle more screen sizes, screen orientations and for tablet specific interface.

4.2.2 Activities

In CORE Mobile, the Android application consists of these activities:

- StartMain

- MainSearch
- SearchResults
- RecordDetails
- SavedDocuments
- BrowseHistory

`StartMain`

It is the main activity of the application. This activity is run first when the application is started. From the interface of the activity user can navigate to search activity, application preferences and saved or browsed documents. In this activity the recommendations are presented to the user and he can navigate directly to these recommended documents and download them.

`MainSearch`

`MainSearch` activity has interface for searching articles from the CORE Server. A user can search simply by any keyword, or he can select additional filters like Author, Title, Publisher, etc. If he does not specify additional filter, the keyword is searched in all fields and in fulltext document also (this behaviour is set in the CORE Server).

`SearchResults`

When the user submits a search, the request is sent to the CORE Server and when the application gets the response, it launches new activity `SearchResults`, which processes the response message and show the results in a list. Each record in the list contains the title of the document, publisher (source repository) and date. When the user selects the desired document, it is sent to another activity `RecordDetails`.

`RecordDetails`

The activity class with document details, where the user can see all the metadata of the document, can save the document and try to find similar document. Immediately when the activity is started, the opened document is saved to the database, to the `BrowsedDocuments` table (only metadata). When the user selects to save the document, the record is saved to the `SavedDocuments` table and then application tries to download the fulltext pdf from the CORE Server. If the download is successful, the `Save document` button disappears and instead of it the `Open document` button is shown. If the user clicks on it, the application sends the request for opening the file to the operating system with the information about the filepath and document type. If the application for viewing pdf files is installed, it will launch for opening the document. This means that the CORE Mobile application is not suitable for viewing pdf files, but it uses external application for this purpose.

`SavedDocuments` **and** `BrowseHistory`

These activities are run from the main activity and shows the documents stored in database (in tables `SavedDocuments` and `BrowseHistory`. The activities use the same layout as the

`SearchResult` activity. The user can select the record and view the metadata and open (for the `SavedDocuments` activity) or download (for the `BrowseHistory` activity) the documents.

4.2.3 `Utils` class

All additional functions for other operations used in more activities are situated in the `Utils` class. For example, the method for downloading the content from the internet, in the case of the CORE Mobile application downloading pdf documents from the CORE Server. Downloading the documents is based on the `HTTP Request` and the `HTTP Response` methods. For better stability of the downloading process, the transfer is realised using a buffer. If the document is bigger than 1kB, the content is divided into more messages and received sequentially.

4.2.4 XML-RPC module

For implementing XML-RPC communication there is an open-source project `android-xmlrpc`[17] hosted on *Google Code* service. This project provides libraries for handling XML-RPC communication, like creating XML-RPC Request messages or parsing the XML-RPC Response messages. Other important part is serialization, when the structured data are transformed to serial data that can be written it text form to XML structure. The module can do the serialization for many standard data types. For the representing data in the CORE modules, specific data structures are defined. Because of this, we had to implement the serialization methods for these specific data types and structures (transforming complex structures to simple array or list of strings).

4.2.5 Structure classes

Data structures are important to define abstract objects of the documents and search results. Since the applications are sending this data through network, they must use strictly same data types. The best way how to achieve this to use one module for both the CORE Mobile and CORE Server applications. This is not a problem for the CORE Server and Android client, since they are based on Java. The iOS client has to copy these structures to its own class. There was an idea to make this data universal, when the applications will not have to understand exactly the incoming message and parse it with strict rules. Unfortunately it was realized when the most applications was almost finished and it was not implemented yet. There are two classes `SearchResult` which define the structure of the document with its metadata. The second class `SearchResults` contains an array of found documents (array of class `SearchResult`), other information about searching and also methods for manipulating the data (converting data from array of structures to simple array of strings and backwards).

4.2.6 Database and `DataHelper` class

Since the CORE Mobile application needs to store data, there are few options how to do it. Different operations need different manipulation. For example for downloaded documents the best way is to store them in separate files, not in the database. For metadata and other information about saved and browsed documents it is better to use a database. Android platform provides a framework for using the `SQLite` database engine. This framework consists of APIs and classes that make using the database much easier. To create and maintain the database, there is a prototype class called `SQLiteOpenHelper`. To implement own

database, developer has to create new class overriding `SQLiteOpenHelper` class. For the even better manipulation with database we created the class `DataHelper`, which contains the subclass `OpenHelper` that overrides the prototype class `SQLiteOpenHelper`. The `DataHelper` class contains also other objects for manipulation with database, like table names constants, prepared statements, object abstracting the database (class `SQLiteDatabase`, database constructor and methods for data manipulation: `insert, selectAll, deleteAll,...`.

4.3 iOS client

The following issues must be considered for developing an iOS application. At first, we had to choose a form of application, since there are few different frameworks and approaches, like *Nimble kit framework, View-based, Window-based or Navigation-based application*, and others. Since we were developing two clients for different platforms, we wanted to make these applications similar as much as possible. The most similar approach to Android platform is View-based application (or special split-view based). The CORE Mobile clients are in the most functionality very similar. In this section we will discuss only difference between this applications to not repeat all the methods and approaches from section 4.2.

4.3.1 Graphic interface

iOS platform in view-based application mode uses defined views for representing graphical interface. The views are defined in *Interface Builder* module provided in standard iOS Development Kit. While the components are defined globally in Android platform and developer can access them by calling proper functions, in iOS platform developer has to connect the components with proper modules in Interface Builder. He can define the actions and receivers of the actions.

4.3.2 Data storing

Downloading the full-text documents is realized through the HTTP Protocol, in the similar way like in Android client. The difference is in the way of storing the documents. When user saves the document, it is downloaded and stored to iBooks storage - standard place for storing digital books and pdf files on iPad and iPhone devices. For the storing application data there are also provided libraries for handling with SQL database. The iOS platforms use `SQLite3` database engine.

4.3.3 XML-RPC communication

For the XML-RPC communication in the CORE Mobile for iOS we use the XML-RPC module from open source project WordPress for iOS[2]. It is the most suitable project for our purposes, since the implementation is also tested and the functionality is versatile. The WordPress for iOS project is distributed under the GNU General Public License. The module is integrated to the CORE Mobile for iOS application and remote calls are the same as remote calls used in the CORE Mobile for Android application.

Repository	NrRecords	NrPdfs	PercentagePdfs
eprints.soton.ac.uk	57232	10975	19,18%
www.leodis.net	56066	0	0,00%
eprints.gla.ac.uk	39160	2961	7,56%
epubs.cclrc.ac.uk	29567	265	0,90%
eprints.lse.ac.uk	24436	4533	18,55%
eprints.lancs.ac.uk	23725	4596	19,37%
oro.open.ac.uk	17167	5245	30,55%
eprints.ecs.soton.ac.uk	15921	7196	45,20%
eprints.cdlr.strath.ac.uk	15534	3397	21,87%
irep.ntu.ac.uk	14183	103	0,73%
...			
Average	**4888**	**866**	**29.14%**
Total (142 repositories)	**562075**	**99614**	**17.72%**

Table 4.1: The number of collected metadata and full-text content from 10 largest UK repositories (in terms of metadata records) and the overall ratio of full-text occurrence to the number of metadata records from 142 UK repositories

4.4 Evaluation

At the time of writing the CORE harvesting system has been tested on 142 Open Access repositories from the UK. Table 4.1 provides preliminary statistics on the amount of content harvested from UK repositories using our tool. We expect to extend the number of repositories in the future to all available repositories listed in OpenDOAR. A larger number of repositories will increase the demand on the storage space as in our case not only metadata, but also full-text has to be stored as opposed to services, such as OAIster or BASE. To provide a rough estimate, the Open University's repository Open Research Online (ORO) contains more than 16,000 records out of which about 5,000 records are accompanied by full-texts. Storing the full-text records requires about $5GB$ of storage space. There are currently over 30 million metadata records stored across Open Access repositories worldwide. It is expected that about 10% of the records contain links to downloadable pdfs, which would account for about 3 million full-text papers stored across Open Access repositories worldwide. by generalizing from the ORO data we can estimate that about $3TB$ of space are required to cache all these documents. Though the real space needed will be due to the index and metadata larger, this amount can still be relatively easily managed and the size does not prevent us from extending the system to the remaining repositories listed in OpenDOAR (more than $1,600$ non-UK repositories).

In final period of developing the CORE Mobile applications, different people tested their functionality and the user interface. These tests were focusing on the main goals of the mobile applications we set in the section 3. As a result, we discovered, that people used to specific platform (iOS or Android), had no problem using the CORE Mobile applications for these platforms, since the CORE Mobile applications use the standard methods and components for viewing and navigating in the application.

Chapter 5

Conclusion

This project aims to improve the accessibility of digital content stored in the World Wide Web. We wanted to make universal platform, that could be used like a standalone tool for the specific needs as well as the centralized application consolidating all the open digital libraries in the internet. We tested the capability for both these purposes and the CORE Server was successful in these tests. At the time of writing these work, the CORE Server was tested on 142 UK repositories and downloaded about 560000 records with almost 100000 full-text PDFs. This content is accessible from the CORE Mobile applications for Android and iOS devices. These applications focus on easy and intuitive use. The CORE Mobile users can search the article, save and download it to their device and read it without broadband connection. This system can be used in different environments or universities. The CORE Server and the CORE Mobile applications have versatile configurations. The CORE Server can set different source repositories for the content harvesting and the CORE Mobile applications can set different CORE Servers for searching articles. According to our goals, we developed mobile applications that personalise user's needs and interests. When the user browses or saves the articles, application stores their keywords to the database. Then, when the user starts the application again, it tries to search similar articles and present the results to the user.
This work presented our approach for improving the accessibility of digital content. However, there are still many opportunities and approaches for the future researches in this field. I believe this topic is important for researchers from any field of interests.

Bibliography

[1] Francisco Alvarez-Cavazos, Roberto Garcia-Sanchez, David Garza-Salazar, Juan C. Lavariega, Lorena G. Gomez, and Martha Sordia. Universal access architecture for digital libraries. In *Proceedings of the 2005 conference of the Centre for Advanced Studies on Collaborative research*, CASCON '05, pages 12–28. IBM Press, 2005.

[2] Inc. Automattic and Wordpress community. Wordpress for ios, 2010. http://ios.wordpress.org/.

[3] DeWitt Clinton. Opensearch 1.1 (draft 4). *Opensearch.org*, 2011.

[4] The Nielsen Company. U.s. smartphone market: Who's the most wanted?, April 2011.

[5] MOAI Developers. Moai, an open access server platform for institutional repositories, 2008. http://moai.infrae.com/.

[6] Kat Hagedorn. Oaister: a „no dead ends“ oai service provider. *Library Hi Tech*, 21(2):170–181, 2003.

[7] Bill Hubbard. Opendoar : the directory of open access repositories.

[8] Dublin Core Metadata Initiative. Expressing dublin core description sets using xml (dc-ds-xml). 2000.

[9] Open Archives Initiative. Protocol for metadata harvesting, v2.0, 2002.

[10] Open Archives Initiative. Object reuse and exchange, v1.0, 2008.

[11] Carl Lagoze and Herbert Van De Sompel. The open archives initiative: building a low-barrier interoperability framework. In *In JCDL 01: Proceedings of the 1st ACM/IEEE-CS joint conference on Digital libraries*, pages 54–62, 2001.

[12] Fang Liu, Clement Yu, and Weiyi Meng. Personalized web search for improving retrieval effectiveness. *IEEE Trans. on Knowl. and Data Eng.*, 16:28–40, January 2004.

[13] Paolo Manghi, Marko Mikulicic, Leonardo Candela, Michele Artini, and Alessia Bardi. General-purpose digital library content laboratory systems. In Mounia Lalmas, Joemon Jose, Andreas Rauber, Fabrizio Sebastiani, and Ingo Frommholz, editors, *Research and Advanced Technology for Digital Libraries*, Lecture Notes in Computer Science, pages 14–21. Springer Berlin / Heidelberg, 2010.

[14] Johanna McEntyre and David Lipman. Pubmed: bridging the information gap. *CMAJ*, 164(9):1317–1319, 2001.

[15] Alessandro Micarelli, Fabio Gasparetti, Filippo Sciarrone, and Susan Gauch. The adaptive web. chapter Personalized search on the world wide web, pages 195–230. Springer-Verlag, Berlin, Heidelberg, 2007.

[16] Dirk Pieper and Friedrich Summann. Bielefeld academic search engine (base): an end-user oriented institutional repository search service. *Library Hi Tech*, 24(4):614 – 619, 2006.

[17] pskink@gmail.com. android-xmlrpc: Very thin xmlrpc client library for android platform, 2008. http://code.google.com/p/android-xmlrpc/.

[18] Graham Walton, Susan Childs, and Elizabeth Blenkinsopp. Using mobile technologies to give health students access to learning resources in the uk community setting. *Health Information, Libraries Journal*, 22:51–65, 2005.

[19] Judith Wusteman. Ojax: a case study in agile web 2.0 open source development. *ASLIB Proceedings*, 61(3):212–231, 2009.