Bibliographic Metadata Harvesting to Support the Management of an Institutional Repository

Ricardo Miguel Loureiro da Costa

Dissertation submitted to obtain the Master Degree in Information Systems and Computer Engineering

Jury

President: Prof. Doctor José Carlos Alves Pereira Monteiro
Supervisor: Prof. Doctor José Luís Brinquete Borbinha
Member: Prof. Doctor Bruno Emanuel da Graça Martins

November 2012
Acknowledgements

As any other Masters Thesis, this work has been a challenge from the beginning. A few people helped and guided me along the road and I would like to thank them for that.

First, I would like to thank my advisor, Professor José Borbinha, for all the guidance and patience throughout this work. I would also like to thank him for the challenges he put in front of me, as I truly believe they have contributed for a better final work.

Secondly, to all my colleagues at Instituto Superior Técnico that somehow have contributed to this work, not only by being there throughout the years, but also for the discussions about how to make a better work out of this thesis.

Next, to Patricia, for being at my side during the toughest hours, providing me the strength to carry on. Thank you for all your support and patience.

But more importantly, I would like to thank my parents, Carlos and Manuela, for their efforts throughout the years that led to this point. It has been a long road and I cannot thank them enough for all the sacrifices, support, strength and love they have given me. From the bottom of my heart, thank you.
Resumo

Esta tese aborda o problema da recolha automática de metadados bibliográficos a partir de diversos serviços de indexação, no contexto da população de repositórios institucionais. Sendo a inserção manual de registos uma tarefa fastidiosa e propensa a erros, a automatização do processo visa facilitar a gestão de um repositório. No entanto, a recolha automatizada de registos tem de lidar com o problema de identificação autores e com a necessidade de consolidação de registos duplicados recolhidos a partir de diferentes serviços. Numa abordagem à automatização da referida tarefa, introduzimos um sistema que se propõe a recolher metadados bibliográficos a partir de fontes de informação disponíveis publicamente, identificar e consolidar os registos recolhidos que sejam considerados duplicados e disponibilizar os resultados dessa consolidação a partes externas interessadas na informação, como é o caso de um repositório institucional.

O sistema proposto foi testado com metadados bibliográficos reais correspondentes a publicações científicas de um subconjunto de professores do Instituto Superior Técnico. Os resultados da avaliação mostram que, apesar do tempo necessário para a operação de identificação e consolidação, os registos consolidados contêm uma agregação válida de toda a informação disponível no sistema e podem ser acedidos eficientemente por entidades externas através de interface máquina-para-máquina.

Palavras-Chave: Metadados Bibliográficos, Recolha automática, Repositórios Institucionais, Consolidação de Duplicados
Abstract

This thesis approaches the problem of automatic harvesting of bibliographic metadata records from several indexing services, in the context of the population of institutional repositories. Since the manual insertion of records is a tedious and error-prone task, the automation of the process intends to facilitate the management of a repository. However, the automated harvesting of records has to deal with the problem of identifying authors and with the need to consolidate duplicate records retrieved from different services. In an approach to the automation of the aforementioned task, we introduce a system that proposes to harvest bibliographic metadata records from different information sources publicly available, identify and consolidate the retrieved records that are considered duplicates and make available the results of such consolidation to external parties that are interested in the information, such as an institutional repository.

The proposed system was tested with real bibliographic metadata corresponding to scientific publications of a subset of faculty members at Instituto Superior Técnico. The results of the evaluation show that, despite the required time to identify and consolidate, the merged records contain a valid aggregation of all available information in the system and can be efficiently accessed by external entities through a machine-to-machine interface.

Keywords: Bibliographic Metadata, Automated Harvesting, Institutional Repositories, Duplicate Consolidation
# Contents

Acknowledgements ................................................. i  
Resumo .................................................................. iii  
Abstract .................................................................. v  

1 Introduction ......................................................... 1  
1.1 Motivation ...................................................... 1  
1.2 Problem ......................................................... 2  
1.3 Contributions .................................................. 3  
1.4 Structure of this Document ................................. 3  

2 Related Work ..................................................... 5  
2.1 Digital Libraries .............................................. 5  
2.1.1 Institutional Repositories ............................... 6  
2.1.2 Interoperability ........................................... 6  
2.1.3 Content Management ................................. 7  
2.1.4 Software .................................................. 8  
2.1.4.1 DSpace ............................................... 8  
2.1.4.2 Fedora ............................................... 8  
2.1.4.3 EPrints ............................................... 8  
2.2 Scholarly Communication ................................. 9  
2.2.1 Bibliographic References .............................. 9  
2.2.2 Bibliometrics .............................................. 9
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.3</td>
<td>Identifiers</td>
<td>12</td>
</tr>
<tr>
<td>2.3.1</td>
<td>Resource Identifiers</td>
<td>12</td>
</tr>
<tr>
<td>2.3.1.1</td>
<td>ISBN and ISSN</td>
<td>12</td>
</tr>
<tr>
<td>2.3.1.2</td>
<td>URI and URL</td>
<td>13</td>
</tr>
<tr>
<td>2.3.1.3</td>
<td>DOI</td>
<td>13</td>
</tr>
<tr>
<td>2.3.2</td>
<td>Author Identifiers</td>
<td>14</td>
</tr>
<tr>
<td>2.4</td>
<td>Metadata</td>
<td>15</td>
</tr>
<tr>
<td>2.4.1</td>
<td>MARC21 and UNIMARC</td>
<td>16</td>
</tr>
<tr>
<td>2.4.2</td>
<td>Metadata Object Description Schema (MODS)</td>
<td>16</td>
</tr>
<tr>
<td>2.4.3</td>
<td>BibTeX</td>
<td>17</td>
</tr>
<tr>
<td>2.4.4</td>
<td>Dublin Core Metadata Element Set (DCMES)</td>
<td>17</td>
</tr>
<tr>
<td>2.4.5</td>
<td>Metadata Encoding and Transmission Standard (METS)</td>
<td>18</td>
</tr>
<tr>
<td>2.5</td>
<td>Functional Interfaces</td>
<td>19</td>
</tr>
<tr>
<td>2.5.1</td>
<td>Z39.50</td>
<td>19</td>
</tr>
<tr>
<td>2.5.2</td>
<td>OAI-PMH</td>
<td>19</td>
</tr>
<tr>
<td>2.5.2.1</td>
<td>REST</td>
<td>20</td>
</tr>
<tr>
<td>2.5.3</td>
<td>SWORD</td>
<td>21</td>
</tr>
<tr>
<td>2.6</td>
<td>Data and Transformations</td>
<td>22</td>
</tr>
<tr>
<td>2.6.1</td>
<td>Extract-Transform-Load</td>
<td>22</td>
</tr>
<tr>
<td>2.6.2</td>
<td>Data Quality</td>
<td>22</td>
</tr>
<tr>
<td>2.6.3</td>
<td>Duplicate Detection</td>
<td>23</td>
</tr>
<tr>
<td>3</td>
<td>Problem Analysis</td>
<td>25</td>
</tr>
<tr>
<td>3.1</td>
<td>Description of the Problem</td>
<td>25</td>
</tr>
<tr>
<td>3.2</td>
<td>Information Sources</td>
<td>26</td>
</tr>
<tr>
<td>3.2.1</td>
<td>Definition</td>
<td>26</td>
</tr>
<tr>
<td>3.2.2</td>
<td>Importable</td>
<td>28</td>
</tr>
<tr>
<td>3.2.3</td>
<td>Analysis</td>
<td>28</td>
</tr>
<tr>
<td>3.3</td>
<td>Metadata Schema</td>
<td>29</td>
</tr>
<tr>
<td>3.4</td>
<td>Functional Interfaces</td>
<td>30</td>
</tr>
</tbody>
</table>

viii
### 4 Proposed Solution

#### 4.1 Use Cases

- **4.1.1 Register unit**
- **4.1.2 Register information source**
- **4.1.3 Register author**
- **4.1.4 Set aliases of author**
- **4.1.5 Register identifier**
- **4.1.6 Get list of records**
- **4.1.7 Configure system**
- **4.1.8 Order records harvest**
- **4.1.9 Import records**

#### 4.2 Domain

- **4.2.1 Information Source**
- **4.2.2 Identifier**
- **4.2.3 Record**
- **4.2.4 Author**
- **4.2.5 Unit**

#### 4.3 Architecture

#### 4.4 Used Technology

#### 4.5 System Configuration

#### 4.6 Harvest

- **4.6.1 Scheduling**

#### 4.7 Duplicate Detection

#### 4.8 Interfaces

- **4.8.1 REST**
- **4.8.2 Graphical User Interface**

### 5 Validation
List of Tables

2-1 Bibliographic references in different styles. ............................................. 10
2-2 Comparison of resource identifiers. .............................................................. 14
2-3 Element set in Dublin Core ................................................................. 18
2-4 Mapping between metadata schemas and types ........................................ 18
3-5 Homepage URLs of the analysed services. .............................................. 29
3-6 Comparison of different potential information sources. .......................... 30
List of Figures

2-1 An informal overview of an environment of services based on OAI-PMH . . . . . 19
3-2 Concept map of the problem and its context. ................................. 27
3-3 Results of searching "Tribolet" in IEEE Xplore .............................. 28
4-4 Use cases diagram. .......................................................... 34
4-5 UML Class diagram of the domain model. .................................. 36
4-6 Dependency view of the architecture of the system. ...................... 40
4-7 Harvest process flowchart. ...................................................... 47
4-8 UML Class diagram of importation module. .................................. 48
4-9 Homepage of the system ....................................................... 53
4-10 Authors page of the system .................................................. 53
4-11 Add author window ......................................................... 54
4-12 Import publications windows ............................................. 54
4-13 Author details window ...................................................... 54
4-14 BibTeX download window ................................................... 55
4-15 Units window ............................................................... 55
4-16 Add unit window ........................................................... 56
4-17 Information Sources window ............................................. 56
4-18 Add information source windows ...................................... 57
4-19 REST Services window .................................................... 57
5-20 Time to insert information sources. .................................... 61
5-21 Time to insert units. ....................................................... 62
5-22 Time to insert authors. .................................................... 63
5-23 Time to harvest records, per information source. .......................... 64
5-24 Time to consolidate records. ....................................................... 64
5-25 Time to access all consolidated records. .................................... 66
5-26 Comparison between time to consolidate all records and time to access all records. 66
Listings

4-1 Excerpt of information sources configuration file .................................................. 42
4-2 Example of organization units configuration file ...................................................... 42
4-3 Example of duplicate fields configuration file .......................................................... 44
4-4 Excerpt of BibtexCompliance configuration file ..................................................... 44
5-1 Record retrieved from Google Scholar ................................................................. 65
5-2 Record imported from file ...................................................................................... 65
5-3 Result of consolidating duplicate records ............................................................... 65
A-1 Example of retrieved metadata records from Google Scholar .................................. 78
A-2 Example of retrieved metadata records from Microsoft Academic Search ............ 79
A-3 Example of retrieved metadata records from ACM Digital Library ......................... 80
A-4 Example of retrieved metadata records listing from DBLP ................................ 81
A-5 Example of retrieved metadata record from DBLP of a publication ....................... 82
A-6 Example of retrieved metadata records from ResearcherID ................................ 83
A-7 Example of retrieved metadata records from CiteSeer ......................................... 85
A-8 Example of retrieved metadata record from CiteSeer of a publication .................... 86
B-1 Example of body for Add Units service ................................................................. 90
B-2 Example of body for Update Unit service ............................................................. 90
B-3 Example of body for Add Information Sources service ....................................... 92
B-4 Example of body for Update Information Source service ................................... 93
B-5 Example of body for Add Authors service ............................................................ 95
B-6 Example of body for Update Author service ........................................................ 96
B-7 Example of body for Add Aliases service .............................................................. 96
Chapter 1

Introduction

Scholarly communication has been made through a traditional model run by publishers where several stakeholders interact in order to try and disseminate knowledge among the scientific community. That model induces a large scattering of articles among several journals (Johnson, 2002) and if we intend to look at the scientific production inside an institution, those works are not easily accessible.

The introduction of institutional repositories (IR) (see Section 2.1.1), a special type of a digital library present at institutions that collect the intellectual work produced by the community members, has been contributing to concentrate it in a structured environment, providing a way of better accessing and evaluating that work.

However, given the multitude of existent versions and formats for representing information contained in different sources, and the problem of name management of authors (Cals & Kotz, 2008), the automatic population of an IR becomes a challenge.

1.1 Motivation

The main motivation for this work is the SOTIS system (Coxo, 2009). SOTIS is the institutional repository implemented at Instituto Superior Técnico (IST) and intends to surpass the problem of low adherence by faculty members by requiring a minimal human intervention. Instead of expecting researchers to deposit their works into the system, SOTIS acts proactively to recover metadata records and full-text content from several external sources when new publications from the research community are announced. Human intervention is therefore limited to the cases of correction of automatic activities, and registration of works not found. Naturally, manual deposit is also possible.
CHAPTER 1. INTRODUCTION

Given the problem of name matching (Christen, 2006), SOTIS allows for an author to register several names that may be present in his publications. This way, authorship of a given publication can be automatically determined and, in case the system does not find a direct match, authors are inquired whether they are in fact the person identified as the publication’s author.

Since SOTIS relies heavily on a mechanism of automatic population based on external information sources, one of the main challenges to overcome is to establish different strategies to retrieve the authors’ publications records. Another challenge we face is how to provide a full listing of one’s publications to other stakeholders. This is a challenge because having different sources of records leads to possible duplicates, and we don’t want to fill SOTIS with repeated entries.

1.2 Problem

Having an institutional repository in place is not enough for it to be a success. It is only part of a process requiring the collaboration of faculty or specialized staff as the contents need to be inserted into the repository.

Another issue relates to the way an institutional repository fits into its environment. It can be isolated, as a single entity in the scope of an organization, or it can be aware of and interoperate with other systems in order to share (retrieve or provide) information. This scenario raises functional and non-functional interoperability issues, related respectively to the effectiveness of the processes of publishing and harvesting the contents for that purpose, and to the semantic mapping of the metadata schemas, which eventually can be different from system to system.

The objective of this dissertation is, according to the scenarios previously referred, to define, implement and validate a set of fundamental services that can be proved each digital library in general should have as capability nowadays, giving also a special attention to the specific case of the institutional repositories. The main motivation of this work, is to contribute for the development of the SOTIS system. The purpose was to provide a service to automatize the harvesting and consolidation of metadata from external sources, and make the results available to SOTIS for convenient reuse. Therefore, it was also in the scope of this work to develop services allowing for outwards interoperability with other systems interested in the present records.

The problem can be summarized in three main research questions:

- **Research Question 1:** Assuming that a digital library might replicate contents already existing in other digital libraries or services, how can we update the contents of our system from those external ones with the minimal human effort? In this work this challenge was addressed only at the level of metadata, leaving in open the issue of synchronizing the
information objects described by the metadata.

• **Research Question 2:** Assuming that a digital library is required to share information (metadata records) with other digital libraries, what are the most relevant scenarios where that can occur nowadays, and what are the technical requirements and best reference implementations to make that possible in the most efficient and effective way?

• **Research Question 3:** Assuming that a digital library might harvest records from multiple external services, how can we identify possible duplicate records and produce a consolidated set of records available to outside parties?

We will further analyse the problem in more detail in chapter 3.

### 1.3 Contributions

The main contributions of this thesis are:

• A survey of possible information sources from where to retrieve authors’ publications;

• A proposed architecture of a solution for:
  
  – Manage author’s identities in different information sources;
  – Automatically import publication’s metadata from different information sources;
  – Make available to interested parties a consolidated list of publications of each author;
  – Compute basic statistics over the stored information;
  – Allow for expandability of functionalities, whether in information sources, data formats or computed statistics.

• The proposed architecture was implemented in a real system, and the results validated with a set of real metadata sources for a sample of authors of the IST.

### 1.4 Structure of this Document

This document is structured as follows:

• **Chapter 2** presents the work related to the context of this thesis. We approach several topics, from concepts of a higher level such as digital libraries and scholarly communication, to more concrete and implementation level subjects, such as functional interfaces and data manipulation;
• **Chapter 3** provides a detailed analysis of the problem approached by this thesis.

• **Chapter 4** describes the designed and implemented system;

• **Chapter 5** reports the evaluation of such system, with several metrics and in different scenarios;

• **Chapter 6** presents some conclusions and points directions for future work and possible expandability.
Chapter 2

Related Work

2.1 Digital Libraries

With the proliferation of digitally available documents, the traditional concept of library evolved into a more technology-aware context in the form of digital libraries (DL). However, defining that concept has not been a consensual task, as several definitions arise to describe what digital libraries are and what is their purpose (Cleveland, 1998). Two major points of view exist regarding a definition for digital libraries: one from the librarian community and another from the research community (Borgman, 1999). Librarians see digital libraries as institutions or services, whereas researchers tend to define the concept with focus on technology, such as databases or information retrieval techniques. Despite the differences in these perspectives, they both tend to converge in the Digital Library Federation¹ (DLF) definition:

Digital libraries are organizations that provide the resources, including the specialized staff, to select, structure, offer intellectual access to, interpret, distribute, preserve the integrity of, and ensure the persistence over time of collections of digital works so that they are readily and economically available for use by a defined community or set of communities. (Cleveland, 1998)

This definition shows some of the challenges related to digital libraries, such as digital resources management, referencing of those resources and sharing. Several of these concerns related with DL will be addressed in the following sections.

¹http://www.diglib.org
2.1.1 Institutional Repositories

Institutional repositories (IR) can be seen as a type of digital library. They intend to store, select, offer access to, preserve and ensure persistence over time of a set of resources. However, the scope of institutional repositories can be restricted to an institution’s intellectual capital. (Lynch, 2003) identifies the following as factors to the arise of institutional repositories: the dropping costs of online storage, the existence of standards for harvesting and resource description, and the recognition and definition of the needs for digital preservation, as well as the first approaches to those needs. Also, the first initiatives of publicly available journal articles showed the way of how scholarly communication could change.

Having a central repository, it becomes easier to demonstrate the scientific, social and financial value of a university. But, the implementation of an institutional repository faces challenges that must be addressed for the initiative to be successful. Faculty collaboration is vital and therefore the advantages for a researcher of having their publications in the repository should be well understood. Authors do not receive direct compensation for their publications; they do it to achieve professional recognition and career advancement (Johnson, 2002). A key factor in that recognition is the awareness and visibility of the author’s publications. Therefore, and given that open access publications have higher impact (Antelman, 2004), it becomes of the interest of faculty to deposit them into the repository.

2.1.2 Interoperability

Given the multitude of information that may be transmitted among digital libraries, problems arise related to how can different organizations communicate their information in a way that the other can understand. This is known as the interoperability problem and aims at surpass the heterogeneities that impede such understandable communications. According to (Haslhofer & Klas, 2010), two types of heterogeneities exist:

1. **Structural Heterogeneities**, related to different structures (schemas) to represent the information;

2. **Semantic Heterogeneities**, referring to the different meanings that can be associated to a specific element of information.

In order to tackle this heterogeneities, three strategies are defined (Haslhofer & Klas, 2010):

- **Standardization**: Standardization is a powerful method of ensuring an agreement. Besides being established by accredited institutions, standards can provide means for interoperability at language level or schema level.
2.1. DIGITAL LIBRARIES

- Agreement on a common metamodel: Many times it is not possible for organizations to agree on a standard model because different models are already implemented. In these situations, interoperability can be achieved through the existence of a metamodel to which are established relations from the several in place models.

- Reconciliation of heterogeneities: When no agreement is possible, it becomes necessary to somehow surpass the divergences on the models. Establishing mapping among the different models, i.e. the elements with similar meaning in different schemas is a possible solution. These mappings are known as metadata crosswalks and represent the relationships among elements, semantics and syntax from one schema to another.

2.1.3 Content Management

One of the main purposes of institutional repositories is to collect and preserve the intellectual products of faculty and students within an institution. But, even though an institutional repository serves as a storing platform, the storing process cannot follow an ad-hoc approach, as that will lead to a huge amount of resources with small or no structure. Given the possibility of storage of large quantities of information, it becomes necessary to decide which resources should be kept in the repository. Following the traditional scholarly communication model, digital versions of the published articles should be stored, as they represent the intellectual work of researchers. However, that intellectual work is based upon experimental or observational data, and that too should be preserved (Lynch, 2003), as it provides a basis for future work on the same research area and result comparison.

Another decision that must be made is related to the possible existence of several versions of the same object. For instance, images may be replicated for different purposes, but they still represent the same digital object. On another level, stewardship over the resources is vital. Librarians play a large part in this process, maintaining consistent records and overseeing the storage process executed by faculty members. Having a centralized repository, curated by experts in the area of digital preservation, avoids the problem that many researchers face: curating their own digital collections, that become larger and therefore more difficult to maintain by people that do not necessarily feel comfortable with technology and should be focused on research, not on document stewardship (Lynch, 2003).
2.1.4 Software

2.1.4.1 DSpace

DSpace\(^2\) is an open-source system, written in Java programming language, that serves as repository for all kinds of digital content produced by members of an university or organization. Developed by the MIT Libraries in collaboration with HP Labs, DSpace intends to be a simple system that at the same time supports all the needs of a research organization. The information model is focused on the concept of “communities”, sub-units within an institution, such as schools, departments, labs or research centres, that can adapt the system to their own needs, and manage their own submission process (Smith \textit{et al.}, 2003). DSpace uses a qualified Dublin Core metadata schema for describing items present in collections and later allow browsing and search. That information about the items is then exposed through OAI-PMH. DSpace also supports interoperability via SWORD (both as a server, receiving content, or as a client, submitting content to other repositories) and OpenSearch. According to (Markey \textit{et al.}, 2007), DSpace is the most used institutional repository system in the United States as of 2007.

2.1.4.2 Fedora

Fedora\(^3\) (Flexible Extensible Digital Object Repository Architecture) is a repository software written in Java programming language and developed at Cornell University. It does not provide a full solution for a repository but instead a set of web services for storing, managing and accessing digital content. Fedora provides the concept of “complex object” that aggregates multiple objects of different kinds (Lagoze \textit{et al.}, 2006). Similarly to DSpace, Fedora also supports dissemination of information via OAI-PMH.

2.1.4.3 EPrints

EPrints\(^4\), written in Perl programming language, is an open-source software package for building digital repositories. It provides a web interface for managing, publishing and searching among the repository’s contents and metadata. EPrints also supports several interoperability features, such as OAI-PMH, SWORD protocol, and through usage of plug-ins, \LaTeX, Dublin Core, METS, MODS, among others.

\(^2\)http://www.dspace.org  
\(^3\)http://www.fedora-commons.org  
\(^4\)http://www.eprints.org
2.2 Scholarly Communication

2.2.1 Bibliographic References

Investigators in research areas often come to conclusions regarding their scientific work. The established model of knowledge transmission is based on written articles, describing the investigator’s contributions for the scientific evolution. Those articles are later made available, following some model of publishing. Viewing scientific evolution as a process based on small contributions for a bigger purpose, it is more than natural for the work of some researchers to be based upon previous work of someone else. This process of crediting one’s contribution is called referencing and it contributes for the enrichment of an publication. Authors are, therefore, encouraged to provide a listing of all the bibliographic references used in their work.

Even though references are made along the publication and listed in an appropriate section, it is still necessary to do it in a sufficiently descriptive way, in order for the reader to be aware of who is the author of that contribution. In the current publishing models, publishers often impose a fixed way of describing references.

Several styles exist for references, such as Chicago\(^5\), Harvard\(^6\), Modern Language Association\(^7\), Nature\(^8\), Science\(^9\) and many others. The used style depends on the discipline, and despite the natural differences among different publishers styles, the provided information contains descriptive metadata (see section 2.4) about the reference, like the author(s), title, year of publication, publisher, volume, number, among others.

Table 2-1 shows different reference styles for the same article.

2.2.2 Bibliometrics

Bibliometrics is defined as a set of measurements over text and information in order to evaluate and establish relationships among documents (Norton, 2000). Analysis of documents is made both with quantitative and qualitative methods that provide ways of assessing the impact of a specific document in the community. One of the main factors that influence this assessment is the number of citations an article receives.

The most relevant bibliometrics methods are:

- Citation Analysis When taking into account the main method of scholar communication

\(^5\)http://www.chicagomanualofstyle.org/tools/citationguide.html
\(^6\)http://libweb.anglia.ac.uk/referencing/harvard.htm
\(^7\)http://www.mla.org/style
\(^8\)http://www.nature.com/nature/authors/gta/
\(^9\)http://www.sciencemag.org/site/feature/contribinfo/prep/res.refs.xhtml
and the paradigm of scientific evolution, citations become a subject to look into as a criteria for research evaluation. Citation counts, that is, the number of times a paper is cited may reveal the importance it has in research community (Moed, 2005). Also, another measure that may be relevant is co-citation. Two papers are said co-cited if both of them are cited by a third paper (Small, 1973). However, a high number of citations might be good or bad. A paper may be highly cited because it is important in the field of investigation or because it has flaws other authors decided to correct (Kostoff, 1998). Another problem with citations is what Kostoff calls the "Pied Piper Effect": A large number of citations defines the path of research, even though it may not be the right path towards a scientific break-through (Kostoff, 1998). Summing up, citation analysis is good as an indicator of an individual’s a department’s, an institution’s, a nation’s work, but should not be regarded as a fool-proof criteria.

• **Citation Indexing** With the growing volume of scientific communication over the Internet, it becomes hard for researchers to keep track of recently published literature. However, information over the web is highly disorganized, making it more difficult to find relevant new articles. Citation indexing proposes to catalogue the citations made by an article, linking it with the cited works. Even though citation indices were intended to facilitate information retrieval, they also allow to establish relationships among articles, and identify other relevant related work (Lawrence et al., 1999). Several services provide citation indexing. CiteSeer[^10] is one of them, with functionalities for automatically download the documents and extract the references in an article, resorting to regular expressions and entities (e.g. journals or

[^10]: http://citeseer.ist.psu.edu

<table>
<thead>
<tr>
<th>Style</th>
<th>Bibliographic Reference</th>
</tr>
</thead>
</table>

Table 2-1: Bibliographic references in different styles.
authors’ names) to identify resources and their sub-parts (Giles et al., 1998). Other services that provide searching and citation indexing include: Institute for Scientific Information (ISI), now known as Thomson ISI\textsuperscript{11}, Google Scholar\textsuperscript{12}, Mendeley\textsuperscript{13}, ACM Digital Library\textsuperscript{14}, IEEE Xplore\textsuperscript{15}, Scopus\textsuperscript{16}, and Microsoft Academic Search\textsuperscript{17}.

- **Impact Factor** Publishers’ evaluation is also relevant in the context of bibliometrics. Impact Factor (IF) is defined as the quotient between the number of citations of any article published in the previous two years and the number of articles published in those two years by the journal. The IF is therefore used as a measure to compare journals (Garfield, 1999). However, the Impact Factor is highly contested as it does not represent an “independent” evaluation criteria. Being based on article citations, IF does not take into account the quality of those articles and neither the self-citing problem; IF depends on the research field, as rapidly evolving fields provide a higher IF value; The time to publication also affects the IF value, as the shorter the publication period, the higher the evaluation (Seglen, 1997).

- **h-index** The $h$-index was introduced by J. E. Hirsch in order to surpass the disadvantages of other evaluation measures like the citation count or the total number of papers. It is said that a scientist has index $h$ if $h$ of his or her $N_p$ papers have at least $h$ citations each and the other ($N_p - h$) papers have $\leq h$ citations each. This way, the $h$-index measures the impact of one’s work and allows for comparison of two individuals even if they have a different number of papers or citations (Hirsch, 2005).

- **Webometrics** In the context of digital scholarly communication, new measures arise related to the Internet presence of institutions and authors. Webometrics is defined as “the study of the quantitative aspects of the construction and use of information resources, structures and technologies on the web drawing on bibliometric and infometric approaches” (Björneborn & Ingwersen, 2004). Currently, a service is available at http://www.webometrics.info for ranking institutions worldwide according to their presence on the web. However, this ranking only takes into account the availability of the resources, and therefore open-access initiatives are more likely to have a better ranking.

Many of the bibliometric measures are based upon how many times an article is cited. This raises some issues because the number of citations may be partially controlled by the author himself. This is known as author self-citation and is seen as a sensitive subject. When an author cites himself, it may be for good reasons, such as indicating the continuation of the author’s previous developments. However, when an author cites himself excessively, it may raise questions about the quality and uniqueness of the author’s research.
work, or for bad reasons, such as increasing is reputation and citation count (Borgman & Furner, 2002). Even though bibliometrics provide quantitative measures, its adoption is not well accepted by the community. First, because they do not evaluate the quality of the papers, being based on data known to be imprecise. Other issues also arise related to bibliometrics: evaluation of authors and institutions is based on that data and therefore funding is also dependent on those measures. Further discussion on this topic can be found in (Weingart, 2005).

2.3 Identifiers

Identification is crucial when dealing with collections of resources. It is the common way of referring to a specific resource when it is necessary to uniquely name it among a set of resources that share common characteristics. When analysing identifiers, we can classify them according to four different axis (Paskin, 2008):

- **Uniqueness**: One identifier should refer to only one object. The inverse is not necessary, as the same object may have different identifiers.

- **Resolution**: It should be possible to retrieve an object, given its identifier.

- **Interoperability**: An identifier should be usable outside the direct control of its assigner.

- **Persistence**: An identifier is said persistent if it refers to the same object indefinitely.

Several identifier systems exist related to digital libraries. The most common are presented next.

2.3.1 Resource Identifiers

2.3.1.1 ISBN and ISSN

The International Standard Book Number (ISBN) is a number composed of 10 or 13 digits for uniquely identify published books, or specific editions of books. Each identifier is composed of a prefix “978", stating that it is an ISBN, a country identifier, a publisher identifier, a title identifier and a control digit. Like Digital Object Identifier (DOI) names (see Section 2.3.1.3), ISBN numbers are appointed by agencies responsible for assigning identifiers to their countries (Bradley, 1992). The journal counterpart of ISBN is the ISSN - International Standard Serial Number - and it is represented as an 8-digit number, divided in two groups of 4 digits, being the last number the control digit. The identifier is calculated based on the title of the publication; hence, if the title changes significantly, a new ISSN must be assigned. ISSN identifiers are also assigned to
2.3. IDENTIFIERS

Electronic resources, such as CD-ROM’s, DVD’s and websites. In fact, if a journal has a printed and an online version, different ISSN numbers are assigned to each media\textsuperscript{18}.

2.3.1.2 URI and URL

The Uniform Resource Identifier (URI) is the most common identifier used in the internet. It is a string used for uniquely identify and access a resource across the web, composed of two parts: The Uniform Resource Location (URL) and the Uniform Resource Name (URN). The URL, as its name suggests, provides the location of the object (Berners-Lee \textit{et al}., 1994). It is not persistent, as the locations might change, and therefore the object can no longer be found in its previous location. However, it is unique, since a URL only represents an object, and is often used as identifier. URLs follow the syntax \texttt{<scheme>:<scheme-specific-part>} (example: \url{http://example.com}). The URNs are intended to serve persistently as resource identifiers, regardless of their location. A URN is composed of a namespace identifier (NID) and a namespace specific string (NSS) uniquely identifying the resource in the namespace (Moats, 1997). The following is an example of a URN: \texttt{urn:isbn:0451450523}.

2.3.1.3 DOI

The Digital Object Identifier (DOI) - in the sense of a digital identifier of an object - is a name persistently assigned to an object, not necessarily digital, to provide a persistent link to some information about that object. A DOI name is composed of two parts: a prefix, indicating the registrant that assigns the name and a suffix, uniquely identifying the object in the scope of that registrant. The suffix may, in fact, be an already existent identifier in another system. Therefore, DOI names are unique. They are also persistent: once a DOI name is assigned, it will not change, even if the object is no longer accessible. Each object has attached to itself some kind of descriptive metadata (see section 2.4), as well as some other information such as a URL or an email. The DOI system allows for a name to be resolved (\url{http://dx.doi.org}) by providing the identifier. The system will return the saved information about the object, but not necessarily the object itself. This kind of architecture with a resolution service simplifies the object retrieval, even if it changes over time: it would only be necessary to update the information present in the system (Paskin, 2008).

\textsuperscript{18}\url{http://www.issn.org}
## 2.3.2 Author Identifiers

When trying to identify an object, such as a book or article, typically its title is enough. It is not common for different objects of the same category to have the same title. However, that is not the case regarding authors and their publications. When trying to find all publications of an author, how can we be sure that the listed publications really belong to that author?

There are several reasons to not be sure (Cals & Kotz, 2008) (Enserink, 2009):

1. Different people have the same name;
2. Sometimes initials are mixed into the name;
3. Non-Latin names are converted into the same equivalent (e.g. Zhang);
4. People change name over time (e.g., marriage and divorce);
5. Spelling errors;
6. Affiliation and contact information gets outdated.

The proposed solution for this problem is to create an identifier for each author, and therefore associate the publications to the correct author, based on his or her identifier (Cals & Kotz, 2008).

Given the urgent necessity, several identifier systems were created:

- International Standard Name Identifier (ISNI) \(^{19}\)
- Thomson Reuters’ ResearcherID \(^{20}\)
- Elsevier’s Scopus Author Identifier \(^{21}\)
- Open Research & Contributor ID (ORCID) \(^{22}\)

\(^{19}\)http://www.isni.org/
\(^{20}\)http://www.researcherid.com/
\(^{22}\)http://about.orcid.org/
2.4 Metadata

Metadata is commonly referred to as data about data or information about information. On a more formal level, "metadata is structured information that describes, explains, locates or otherwise makes it easier to retrieve, use or manage an information resource" (National Information Standards Organization, 2004). It can be seen from different perspectives and therefore it is commonly divided into three main types (National Information Standards Organization, 2004):

- **Descriptive Metadata** refers to information about the resource itself. For instance, in a published article, descriptive metadata would contain information about the title, author, abstract and so on, of the article. Thus, we can see that this type of metadata is largely related to the content of the resource and therefore provides semantically-based information.

- **Structural Metadata** tells how resources might be related among themselves, possibly as parts of another resource in a higher level scope. For instance, book chapters may be seen as independent resources in a given context. However, there is still a relation among them, as they all become to the same book. Structural metadata is therefore more about the form of the object, in opposite to descriptive metadata.

- **Administrative Metadata** is related to the intrinsic properties of the resource. Date of creation or modification, file type, access privileges, archiving and preservation informations are just some examples of administrative metadata. This type of metadata becomes important in contexts of specific communities where privileges for accessing certain resources must be defined according to the business logic.

The usage of metadata has been growing over the last decades in line with the growth of digital content and the resultant challenges of technological advances (Sen, 2004). Metadata has been
applied in several areas throughout the years, with a growing importance. Those areas include resource discovery, resource organization, and interoperability (Chan & Zeng, 2006). In order to enhance that organization and interoperability, several metadata schemas have been proposed:

2.4.1 MARC21 and UNIMARC

Both MARC21 and UNIMARC are families of MARC\textsuperscript{24} formats for representing bibliographic information in a way understandable by machines, without prescribing the content or structure of data inside a management system. Instead, they provide guidelines for content and structure of information when exchanged among systems.

The MARC21 format was developed by the Library of Congress as a fusion of CANMARC and LCMARC and provides mechanisms for bibliographic information exchange and use by computers. MARC21 records are divided in different types, each one providing different information about an object. The three most used are (Fritz & Fritz, 2003):

- Bibliographic Records, to describe the item and provide access to that description;
- Authority Records, used for authority control; and
- Holding Records, with information about individual items, such as barcode numbers, volumes, year, etc.

UNIMARC specifies tags, indicators and field codes readable by computers. It is divided in three parts: The Registry Tag is composed of 24 characters and contains data related to the structure of the registry as stated by ISO 2709\textsuperscript{25}. That information is used primarily for registry processing and only secondarily for identifying the bibliographic registry. Each Directory is divided in three parts: the field tag, the number of characters in that field, and the position of the initial character of the field. The Fields contain the bibliographic data (Mac Callum et al., 1989).

| Registry Tag | Directory | Fields | Terminator |

2.4.2 Metadata Object Description Schema (MODS)

MODS\textsuperscript{26} is a schema for a bibliographic element set particularly used in library applications. It was developed by the Library of Congress in order to adapt the existent MARC\textsuperscript{27} to an eXtensible Markup Language (XML) representation and at the same time to reduce some of its complexity.

\textsuperscript{24}http://www.loc.gov/marc/
\textsuperscript{25}http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=41319
\textsuperscript{26}http://www.loc.gov/standards/mods/
\textsuperscript{27}http://www.loc.gov/marc/
Therefore, MODS elements inherit MARC’s elements semantics and as it represents a subset of the later (Guenther & McCallum, 2003). Also, MODS presents itself as a way of tackling the broadening approach of unqualified Dublin Core that reveals itself unsuited in some contexts where high detail is necessary (Gartner, 2003). Hence, it provides a larger element set, with 22 top elements, each one with several subelements\textsuperscript{28}. As said before, MODS is a schema specified in XML and it is based in MARC. However, it uses English language instead of numeric field names, reducing the complexity and allowing the usage of a non-specialist in MARC.

\subsection*{2.4.3 Bib\TeX}

Bib\TeX\textsuperscript{29} is a reference management software for formatting lists of references. It is used in conjunction with \LaTeX{} and eliminates the effort of formatting references present in the document. Bib\TeX{} files are structured in entries, each with its type and tags (not all of them are mandatory), describing the reference. In that way, it becomes descriptive metadata over the referenced documents. An example of a Bib\TeX{} entry is presented next:

\begin{verbatim}
@Article {Cleveland_1998,
    title = "Digital libraries: definitions, issues and challenges",
    journal = "Challenges",
    author = "Gary Cleveland",
    year = "1998",
}
\end{verbatim}

\subsection*{2.4.4 Dublin Core Metadata Element Set (DCMES)}

Dublin Core\textsuperscript{30} (DC) is a metadata standard, composed of fifteen different elements (Table 2-3) with established semantic meaning. All of those elements are optional and may be repeated, according to the needs in a given context. Dublin Core also provides support for extensibility by linkage with elements from other metadata sets. Metadata can be embedded in the object or kept apart in some kind of catalogue as the standard does not prescribe a compulsory approach on this matter. Information kept in metadata following Dublin Core standard can be either free content (such as information regarding the creator) or controlled vocabulary (if such approach is better suited for, in instance, provide a description of the object). Since elements may repeat, Dublin Core provides qualifiers\textsuperscript{31} to further refine their meaning (consider the case when a date of creation and a date of modification are necessary) (Hillmann, 2001).

\textsuperscript{28}A full listing is available at \url{http://www.loc.gov/standards/mods/mods-outline.html}
\textsuperscript{29}\url{http://www.bibtex.org/}
\textsuperscript{30}\url{http://dublincore.org/}
\textsuperscript{31}\url{http://dublincore.org/documents/usageguide/qualifiers.shtml}
2.4.5 Metadata Encoding and Transmission Standard (METS)

METS\textsuperscript{32} is an open standard built upon XML design to convey metadata necessary for object management in a repository and sharing of those objects among several repositories. A METS document is structured in seven major sections: Header (metsHdr), describing the document itself; Descriptive Metadata (dmdSec), descriptive metadata regarding the object; Administrative Metadata (amdSec), containing administrative metadata, such as intellectual property rights among other information; File Section (fileSec), which lists the files that comprise the object; Structural Map (structMap), outlining a hierarchical structure of the object and linking the elements of such structure to files and metadata; Structural Links (structLink) stating the existence of hyperlinks between nodes in the hierarchy previously mentioned; Finally, a Behavior section (behaviorSec), to associate executable behaviors with content (Federation, 2007).

METS does not define a specific structure for descriptive metadata, nor that it is mandatory to be part of the METS documented. That means that it can contain pointers to externally located metadata with its own schema, like Dublin Core or MODS (Guenther & McCallum, 2003).

Establishing a correspondence between each metadata schema and a metadata type we have the mapping on Table 2-4.

\footnotesize
\begin{tabular}{|l|l|l|}
\hline
Descriptive Metadata & Structural Metadata \\
\hline
MARC21/UNIMARC & METS \\
MODS & \\
BibTeX & \\
DCMES & \\
ESE & \\
\hline
\end{tabular}

\footnotesize
\textsuperscript{32}http://www.loc.gov/standards/mets/
2.5 Functional Interfaces

2.5.1 Z39.50

Initially defined as the "ANSI/NISO Z39.50"\textsuperscript{33}, United States national norm and later as the international norm "ISO 23950"\textsuperscript{34}, Z39.50 is a client-server protocol which specifies data structures and interchange rules that allow information retrieval over a database of records. Since different databases may have different ways of describing the contained information, a common model is necessary for that description to which each implementation should be mapped (NISO, 2003). A client performs a search operation on the server database, specifying the values to be matched against the access point of that database. The result set, an ordered list of items corresponding to the query, is then produced and returned to the client. The actual items are not transferred until a present request is sent to the server (Lynch, 1997).

2.5.2 OAI-PMH

The Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH) is a protocol for harvesting records stored in repositories that expose their metadata. Components that participate in this framework are classified as Data Providers, that expose metadata, and Service Providers, those who use the harvested metadata. Each item within a repository has a unique identifier that must follow the URI syntax. Identifiers are used when listing records present in the repository or the identifiers themselves, or when a request of a record in a specific metadata format from an item issued.

Communication with repositories is made over HTTP GET or POST, with POST method not having

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure2-1.png}
\caption{An informal overview of an environment of services based on OAI-PMH\textsuperscript{35}}
\end{figure}

\textsuperscript{33}http://www.loc.gov/z3950/
\textsuperscript{34}http://www.iso.org/iso/catalogue_detail.htm?csnumber=27446
\textsuperscript{35}http://wiki.cetis.ac.uk/What_is_the_OAI_Protocol_for_Metadata_Harvesting
the limitation on the length of arguments. The response format must be a well-formed XML document, validated against a provided schema⁹⁶. Records present in a repository may follow multiple metadata schemas; however, all repositories must implement Dublin Core format for purposes of interoperability (Lagoze et al., 2008).

An example of usage of OAI-PMH is the very REPOX framework (Freire et al., 2006). It serves both as a server and a client in a OAI-PMH interaction. In order to be possible to expose records by OAI-PMH, the Dublin Core metadata schema is required, not because the framework requires it, but because of the protocol itself. Data providers that don't have their records in Dublin Core format can register a XSLT transformation within REPOX that specifies how to convert the data in its original format to required format.

2.5.2.1 REST

OAI-PMH relies upon the Representational State Transfer (REST) architectural style introduced by (Fielding, 2000) in his PhD Thesis. This style to transfer a representation of a resource, typically in XML or JSON, that exists within a server and has a state. REST is not a standard, but it does rely on standards such as HTTP, URL, XML, among others.

In order for a service to be considered RESTful it must meet the following constraints (Fielding, 2000) (Costello):

1. **Client-Server**
   A client-server architectural style should be followed. This allows for the separation of concerns principle, and therefore improving portability of user interface and scalability by simplifying the server components.

2. **Stateless**
   The communication between client and server must be stateless and each request must contain all contextual information for it to be understood. Therefore, server context cannot be used by the client. This constraint improves visibility (by not having to maintain information about a client's request), reliability (because it becomes easier to recover from failures) and scalability (since there are more resources available). However, by being stateless, the network traffic increases, because each request has to transmit possibly repetitive information.

3. **Cache**
   Clients may cache responses. For that, responses have to be, implicitly or explicitly, labelled as cacheable or not. Cache allows for the suppression of some interactions between client

---

⁹⁶[http://www.openarchives.org/OAI/2.0/OAI-PMH.xsd](http://www.openarchives.org/OAI/2.0/OAI-PMH.xsd)
and server, and therefore increases scalability and efficiency. However, cache may reduce reliability, as the information present in the client may already be outdated comparing to the information on the server.

4. **Uniform Interface**

Having a uniform interface means that all resources are available through a generic interface (e.g. HTTP GET, POST, PUT, DELETE).

5. **Layered System**

With a layered system, a client does not know if it is connected to the end server or to an intermediate one along the way. This constraint allows for improvement in scalability (load balancing and shared cache) and security (authentication before access to data). On the other hand, a layered system increases latency times of the interactions, reducing performance.

6. **Code on demand**

This optional constraint allows for the transmission of code from the server to the client in order to extend client functionality. The code may be in the form of applets or scrips (e.g. Javascript). This way, it is possible to reduce the functionalities that need to be pre-implemented and delegate some of the work on data to the client.

### 2.5.3 SWORD

Simple Web-service Offering Repository Deposit\(^\text{37}\) (SWORD) is a protocol for depositing content from one location to another and a profile of the Atom Publishing Protocol. Unlike Z39.50 or OAI-PMH that focus on retrieval, SWORD aims at depositing any kind of files into a remote repository in a standardized way. It supports querying a repository for information about collections available for deposit (request of a service document). Afterwards, the user may deposit the object into the repository. Mediated deposit is also possible, that is, a user (e.g. a librarian) depositing a resource in name of another user (e.g. a teacher) (Currier, 2009). After the deposit, the system sends a response indicating the success, or unsuccess due to unacceptable formats or other causes, of the operation. This protocol is useful in several use cases, such as “an author deposits an article into his institution repository” or “a librarian intends to deposit a surrogate of an object in another repository”. Different deposit interfaces have been developed for EPrints, Fedora and DSpace repositories (Allinson et al., 2008).

\(^{37}\text{http://swordapp.org/}\)
CHAPTER 2. RELATED WORK

2.6 Data and Transformations

Dealing with large amounts of information from many disparate sources brings the problem of data management. This section follows an approach from the source of the information until the usage of that information in the context of organizational business processes.

2.6.1 Extract-Transform-Load

The Extract-Transform-Load (ETL) process, as the name suggests, is a sequential process largely used in data warehousing (Kimball et al., 1998) that consists in three main steps:

1. Extracting the source information from its origin, possibly applying some filter in order to discard unwanted data.

2. Transforming the extracted information to a format suitable for loading in the destination database. In this step, many transformations may be applied, such as: transforming to a standard format (and this includes interpretation of different ways of representing the same concept, e.g. "M" for "Male" or "F" for "Female"), correcting misspellings, resolving name conflicts, combining different data sources, removing duplicates or sorting.

3. Loading the data into the receiving database. This phase might encompass different approaches, such as replace existing data or adding new data, maintaining the previous information.

2.6.2 Data Quality

When consolidating information retrieved from several data sources, it becomes necessary to somehow clean the data. Misspellings may happen or wrong values associated with a concept instance. That is described as a data accuracy, the closeness between the current value and the value that is considered to be the correct one. However, other dimensions need to be addressed when talking about that quality (Batini & Scannapieco, 2006):

- **Completeness**, the extent to which data is available for its defined purpose;

- **Consistency**, related to the semantic rules that must be verified in the data;

- **Currency**, the frequency at which data is updated.
2.6.3 Duplicate Detection

When consolidating information, often occurs that information ends up being duplicated. In that way, it is necessary to eliminate the duplicate information. Since most information is represented as strings, string similarity algorithms are used for duplicate detection, such as Edit Distance, Bag Distance, Smith-Waterman, Longest Common Sub-string, q-grams, Jaro or Winkler (Christen, 2006). A common problem related to bibliographic references is to find whether two different strings represent the same real name of a person, e.g., "John Smith" and "Smith, J.". In addition to the already pointed pattern matching algorithms, others exist that take into account the sound of the words. Those are known as phonetic algorithms and some examples are: Soundex, Phonex, Phonix and Double-Metaphone (Christen, 2006).
Chapter 3

Problem Analysis

This chapter presents an in-depth analysis of the problem stated in section 1.2. It starts by recalling the research questions that guide this work and explain them in a more detailed fashion. It also presents the conclusions of a survey regarding services that may be of use as metadata sources, and the reasons to the choices made in the implementation regarding a metadata descriptive schema, a functional interface for external interoperability, and a business-aware way of grouping authors.

3.1 Description of the Problem

Recalling the research questions defined in section 1.2, this section describes them further and integrates them in the context of the problem.

As seen in section 2.1.4 there are several implementations of institutional repositories systems. Therefore, besides being able to exist as an isolated entity, an institutional repository might be part of a larger environment where it works in collaboration with other external entities to retrieve information. Since an institutional repository does not need to be created from scratch, relying on the environment to populate the repository is seen as a good strategy. Therefore, an automatic process of updating the contents of a repository might prove very valuable, since it prevents the tedious and error-prone task of manually inserting records in the system.

Given the presented context, this work will try to define and explore a process of automatically take advantage of other available services to populate an institutional repository with metadata records, reducing human effort to the minimum required task, such as stewardship and maintenance of the retrieved records. This means that the problem of synchronization (whether by replication or reference) of the digital objects described by the retrieved metadata records will not
be approached by this work.

As stated earlier, this work assumes the operation of an institutional repository in a heterogeneous environment. This approach to the problem implies a need for information sharing processes. To be able to define such processes, it is necessary to investigate the best ways of providing access to the information within the repository. Section 2.5 already presented an analysis of the different functional interfaces that might be suitable to share metadata records, and section 3.4 discusses which of the analysed interfaces is better suited in the context of this work, taking into account the technologies currently used to share information with external parties, as well as technical motives for not using a specific functional interface.

Finally, having a repository that harvests metadata records from multiple services, it is expected that such repository will contain multiple records describing the same information object. These are considered duplicate records, and each one might provide different and valuable information about the described object. This situation brings the problem of consolidating records without loosing the potentially valuable information contained in each individual record. However, a record might have several attributes and the decision regarding which ones are to be used when detecting duplicates might have an high impact in the success of consolidation process. This detail will need to be considered when configuring a system with a duplicated records detection strategy.

In this section it was provided a more detailed explanation of the research questions, fitting them into the expected context of operation of a repository, and giving special attention to the major challenges associated to each research question. Figure 3-2 summarizes the problem and its context in a concept map. Next sections will describe some topics related to the first implementation decisions as a consequence of the analysis previously presented.

### 3.2 Information Sources

This section introduces and defines the concept of information source that will be used throughout this work, based on the analysis of available services that collect and provide metadata records of scholarly communication.

#### 3.2.1 Definition

We define an information source as a service or entity that registers an identifier for an author and, optionally, offers the possibility of harvesting the author’s publications metadata records.

From this definition we can establish that a valid information source in our domain always has, at
least, one publicly available identifier for each author, even if that identifier only has a meaning within the scope of that service.

Therefore, and given the problem of author identification discussed in section 2.3.2, we exclude services that provide publications listings only through queries with the name of the author or some other criteria, such as affiliation. An example of such system is IEEE Xplore, where to acquire the publication’s metadata records a query like the following is necessary:

http://ieeexplore.ieee.org/search/searchresult.jsp?newsearch=true&queryText=Tribolet&x=0&y=0

Figure 3-3 shows the results of searching “Tribolet” in IEEE Xplore. The figure clearly shows a mix of results: a publication with "Tribolet" in its title, a publication authored by Tribolet, J., and in the left sidebar, it is possible to see that another author name is registered: Tribolet, J.M.. This example clearly shows that without a unique identifier, it is not possible to be sure if it is the desired author.

As mentioned before, the capability to provide metadata records is optional: an information source may simply represent some entity that has a unique identifier for each author. This situation is considered because some stakeholders may want to establish a match between an author in the system and that same author in the stakeholders’ system. For instance, SOTIS system may want to map an author through his/her istID and use that identifier to retrieve author’s publications.
3.2.2 Importable

Given that some information sources might be used just for identifier matching among systems, we considered that an information source has an "importable" property that establishes the possibility of retrieval of metadata records through an automatic process. The harvesting procedure will be later described in section 4.6.

3.2.3 Analysis

An analysis was conducted prior to the system implementation in order to assess which existent services could play the role of an importable information source as previously described. That analysis was based on the following criteria:

- Existence of an unique identifier for each author;
- Covered domain;
- Access to the system information;
- Data format of retrieved information;
• Required response document manipulation.

The analysed services are indicated in table 3-5.

<table>
<thead>
<tr>
<th>Information Source</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google Scholar</td>
<td><a href="http://scholar.google.com/">http://scholar.google.com/</a></td>
</tr>
<tr>
<td>Microsoft Academic Search</td>
<td><a href="http://academic.research.microsoft.com/">http://academic.research.microsoft.com/</a></td>
</tr>
<tr>
<td>ACM Digital Library</td>
<td><a href="http://dl.acm.org/">http://dl.acm.org/</a></td>
</tr>
<tr>
<td>DBLP</td>
<td><a href="http://dblp.uni-trier.de/">http://dblp.uni-trier.de/</a></td>
</tr>
<tr>
<td>IEEE Xplore</td>
<td><a href="http://ieeexplore.ieee.org">http://ieeexplore.ieee.org</a></td>
</tr>
<tr>
<td>ResearcherID</td>
<td><a href="http://www.researcherid.com/">http://www.researcherid.com/</a></td>
</tr>
<tr>
<td>ScienceDirect</td>
<td><a href="http://www.sciencedirect.com/">http://www.sciencedirect.com/</a></td>
</tr>
<tr>
<td>Nature</td>
<td><a href="http://www.nature.com/nature/index.html">http://www.nature.com/nature/index.html</a></td>
</tr>
<tr>
<td>CiteSeer</td>
<td><a href="http://citeseer.ist.psu.edu/index">http://citeseer.ist.psu.edu/index</a></td>
</tr>
<tr>
<td>Mendeley</td>
<td><a href="http://www.mendeley.com/">http://www.mendeley.com/</a></td>
</tr>
</tbody>
</table>

Table 3-5: Homepage URLs of the analysed services.

From table 3-6, that summarizes the results of the analysis, it is possible to see that of all analysed services that may act as information sources, 7 of them provide a unique identifier for each Author. As this a mandatory requirement for an information source, services such as IEEE and ScienceDirect are excluded.

For the other ones, a deeper descriptive analysis is presented in Appendix A. As the result of that analysis, it is possible to conclude that the most suitable services to act as information sources are: Google Scholar, Microsoft Academic Search, ACM Digital Library and DBLP. Despite meeting all requirements, ResearcherID requires a high manipulation of the retrieved document, as well as user interaction simulation to allow harvesting of all records of an author. Because of that, it was not included in the set of information sources considered for this work. Mendeley provides an API for retrieving information but requires authentication for each author to harvest his publications. Both ResearcherID and Mendeley will be approached in section 6.1 dedicated to future work.

3.3 Metadata Schema

From the conclusions of the analysis of information sources in the previous section, and taking into account the analysed metadata schemas in section 2.4, the logical choice for the metadata descriptive schema was \texttt{BibTeX}, since it is the prominent one. Other reasons for this choice are that \texttt{BibTeX} is a \textit{de facto} standard to manage bibliographic references and allows to convey
3.4 Functional Interfaces

In order to fetch records from information sources and provide them to other stakeholders, it became necessary to choose a functional interface for machine-to-machine communication.

The choice regarding retrieval from information sources was limited to the available interfaces. In the cases of the analysed services (see section 3.2.3), the requests needed to be done via HTTP GET since it was not possible to identify other interfaces, such as OAI-PMH.

For the interaction with the system as the source, the choice fell on RESTful web services for the following reasons:

1. Simplicity and availability of tools to easily integrate REST web services into the system;
2. Easiness of access from outside, since only HTTP requests are necessary.

Z39.50 was left out because it is no longer used in recent implementations of metadata records interchange. OAI-PMH was not chosen due to the overhead of setting a Service Provider that

<table>
<thead>
<tr>
<th>Information Source</th>
<th>Unique Identifier</th>
<th>Domain</th>
<th>Access</th>
<th>Data Format</th>
<th>Document Manipulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google Scholar</td>
<td>YES</td>
<td>Multiple</td>
<td>Single</td>
<td>B\LaTeX{}</td>
<td>None</td>
</tr>
<tr>
<td>Microsoft Academic Search</td>
<td>YES</td>
<td>Multiple</td>
<td>Single</td>
<td>B\LaTeX{}</td>
<td>Minimal</td>
</tr>
<tr>
<td>ACM Digital Library</td>
<td>YES</td>
<td>Single</td>
<td>Single</td>
<td>B\LaTeX{}</td>
<td>Minimal</td>
</tr>
<tr>
<td>DBLP</td>
<td>YES</td>
<td>Single</td>
<td>Multiple</td>
<td>XML</td>
<td>Major</td>
</tr>
<tr>
<td>IEEE Xplore</td>
<td>NO</td>
<td>Single</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ResearcherID</td>
<td>YES</td>
<td>Multiple</td>
<td>Multiple</td>
<td>HTML</td>
<td>High</td>
</tr>
<tr>
<td>ScienceDirect</td>
<td>NO</td>
<td>Multiple</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Nature</td>
<td>NO</td>
<td>Multiple</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CiteSeer</td>
<td>YES</td>
<td>Multiple</td>
<td>Multiple</td>
<td>HTML</td>
<td>High</td>
</tr>
<tr>
<td>Mendeley</td>
<td>YES</td>
<td>Multiple</td>
<td>API</td>
<td>JSON</td>
<td>High</td>
</tr>
</tbody>
</table>

Table 3-6: Comparison of different potential information sources.

additional information and not just the standard fields.
would not be used given the constraints of the information sources, and a Data Provider that 
would increase the complexity of the proposed solution. Finally, SWORD was obviously discarded 
because the scope of this work is the harvesting of metadata records and not the deposit of digital 
objects described by those records into a repository.

Summing-up, it was chosen a functional interface taking into account the ease of implementation 
as well as the context in which the system will exist, namely the machine-to-machine access 
using HTTP functionalities.

3.5 Units

Almost all organizations are structured in such way that there is an organizational division of its 
resources according to the context where the organization is inserted. Several examples exist: 
departments in enterprises, age-divided teams in a sports clubs, platoons in armies, ... The 
conclusion is that, in the context of an organization, it makes all sense to split resources according 
to some criteria to allow a better management of those resources.

In a teaching and research organization, such as a University, faculty members and researchers 
are also divided into organizational divisions: Academic Units. Then, it is expectable to organize 
the authors registered in the system the same way they are organized in real life context. There-
fore, this work considers the concept of Unit: An organizational division of authors according to 
same criteria used in business context (such as expertise or research interests). Each Unit con-
tains a subset of all authors registered on the system, in such way that each author is associated 
with only unit at the same time.
Chapter 4

Proposed Solution

Chapter 4 describes the proposed solution for the problem described earlier. It exposes the main use cases expected for the system and then approaches a description of the domain considered for the implemented system, and the architecture of such system. Afterwards it explains the mechanisms of metadata records importation and duplicate records detection. The functional interfaces for interaction with the system are detailed in the end of the chapter.

4.1 Use Cases

Taking into account the problem this work proposes to approach, stated in the section 3.1, it is possible to define a number of use cases within the system. This section identifies and details such expected use cases represented in figure 4-4 and presents, when necessary, the several expected scenarios for each use case.

4.1.1 Register unit

Rationale: Register a unit in the system so that it is possible to associate an author to it.

Scenario 1: Register a unit using the graphical user interface.

Scenario 2: Register a unit using a REST service.

Actors: System administrator; librarian.

4.1.2 Register information source

Rationale: Register an information source in the system, being importable or not, so that it is possible to associate an author to it through an identifier.
4.1.3 Register author

**Rationale:** Register an Author in the system by providing, at least, the authoritative name and the unit to which the author belongs to.

**Scenario 1:** Register an author using the graphical user interface.
**Scenario 2:** Register an author using a REST service.
**Actors:** System administrator, librarian.
4.1.4 Set aliases of author

**Rationale:** Set the aliases list of an author so that it is possible to find more information about that author.

**Scenario 1:** Set aliases list of author using the graphical user interface.

**Scenario 2:** Set aliases list of author using a REST service.

**Actors:** System administrator, librarian.

4.1.5 Register identifier

**Rationale:** Associate an identifier to a pair (author, information source) allowing the importation of records of such author from the given information source.

**Scenario 1:** Register an identifier using the graphical user interface.

**Scenario 2:** Register an identifier author using a REST service.

**Actors:** System administrator, librarian.

4.1.6 Get list of records

**Rationale:** Get a list of records according to some criteria, such as the format, author, or information source.

**Scenario 1:** Get list of an author's records imported from a given information source.

**Scenario 2:** Get consolidated list of records.

**Scenario 3:** Get list of imported records in BibTeX format.

**Actors:** Librarian, client.

4.1.7 Configure system

**Rationale:** Configure different options that affect the behaviour of the system.

**Scenario 1:** Configure duplicate detection criteria.

**Scenario 2:** Configure BibTeX compliance criteria.

**Actors:** System administrator, librarian.

4.1.8 Order records harvest

**Rationale:** Order a harvest process.

**Scenario 1:** Order records harvesting from information source using the graphical user interface.
Scenario 2: Order records harvesting from information source using a REST service.

Actors: Librarian, client.

4.1.9 Import records

Rationale: Import metadata records.

Scenario 1: Import records of an author in a bibliographic source.

Actors: Bibliographic source.

4.2 Domain

The domain of the system is based upon five different core entities: Author, Unit, Identifier, Information Source, and Publication. The class diagram in figure 4-5 shows the relationships among these entities.

![UML class diagram](image)

Figure 4-5: UML class diagram describing the domain entities of the system and the relationships among them.

The following subsections describe in further detail each one of the core entities of the implemented system, focusing in the relationships among entities.

4.2.1 Information Source

The definition of information source was stated in section 3.2.1. For the purpose of explaining the domain model, further details regarding the implementation of the defined concept will be approached.
4.2. DOMAIN

An information source contains the following attributes:

- **Name**
  The name of the information source.

- **Homepage URL**
  The URL that takes to the homepage of the information source.

- **User profile URL**
  The URL of the profile page of an author in the information source.

- **Importable**
  Importable is a property that defines the possibility or not to import metadata records from the information source as explained in section 3.2.2. The next attributes are relevant only when an information source is considered importable.

- **Harvest URL**
  The URL used to harvest the document with the metadata records of an author's publications.

- **Data format**
  The format of the retrieved data, such as XML or BibTeX.

- **Import class**
  The fully qualified name of the Java class responsible for the manipulation of the retrieved document into the BibTeX format.

- **Max authors**
  The maximum number of authors whose publications will be harvested in a single import operation.

- **Next harvest date**
  The date for the first (or next) harvest operation for an information source.

- **Repeat value**
  The number of units of time to wait until the next periodic harvest operation.

- **Repeat frequency**
  The unit of time to wait. It may be minutes, hours or days.

An information source interacts with other entities in the domain, namely the identifier entity and the publication entity. Each information source keeps track of the identifiers that belong to its scope. For instance, a "Google Scholar" information source contains references for all identifiers of authors that have a profile page in Google Scholar. On the other hand, an information source...
also keeps track of the publications that were harvested from it. This allows filtering publications by their information source.

### 4.2.2 Identifier

The identifier entity represents the unique identifier each information source has for an author. It is responsible for holding the vital information necessary to publications harvesting, since it is the way of identifying an author in an external system. Also, the identifier is necessary in order to provide a link to the profile page of an author in the information source system.

The identifier entity encompasses the following attributes:

- **Identifier**
  The identifier value, the one that will be used to harvest publications and link to the profile page.

- **Last Harvest Date**
  The date when the identifier was last used in a harvesting process. This allows to define a priority mechanism for the automatic harvesting process.

As mentioned before, an identifier only exists in the scope of an information source. Also, an identifier always belongs to an author. Therefore, there is a one-to-many association from author to identifier and also from information source to identifier.

### 4.2.3 Record

The record entity represents a vital part of the system. It contains all information regarding a harvested record from an information source and it has the following attributes:

- **Type**
  The type of the publication. This allows to distinguish among different kinds of records, such as articles, books, inproceedings, etc.

- **Full Record**
  The full harvested record from a publication in \LaTeX format. Since the record depends on the harvested information, there is no limitation for the contained information, being possible to exist fields that are not standard in \LaTeX and only have a meaning in the originating scope.

- **First Harvest Date**
The date when the record was first harvested, that is, when it was inserted in the system and no other version of it was detected.

- **Last Modification Date**
  Since multiple harvest process will take place across time, it may happen that the same record is imported more than once. In that case, that record may be a different version of the already existent record. This attribute represents the date when a harvested record was detected to be a modified version of an existing record.

As mentioned before, the record entity has a many-to-one association to an information source. This allows to know exactly from where has a publication been harvested from, and therefore identify different versions of the same object. The detailed harvesting process will be described in section 4.6.

The record entity also has a many-to-one association to an author, since it is expected for an author to have several records in the system.

### 4.2.4 Author

An author entity represents any faculty member or researcher that has authored scientific publications. It contains the following attributes:

- **Authoritative Name**
  The authoritative name of the author, that is, the name frequently used by the author to sign his publications.

- **Full Name**
  The full name of the author.

- **Aliases**
  A list of aliases of the author. Aliases are other names by which the author is also known, such as variations with middle names or initials.

As it can be seen in the class diagram in figure 4-5, the author entity is the central entity in the system. It contains an association to the identifier entity as described in 4.2.2 and an association to the publication entity that was explained in 4.2.3. Besides those associations, an author entity also has a one-to-one association with the unit entity. This means that an author always belongs to one, and only one, unit.
4.2.5 Unit

The unit entity represents an organizational unit in the scope of the organization where the system is installed, as it was approached in section 3.5. It contains two attributes:

- **Name**
  - The name of the unit.

- **Acronym**
  - The acronym of the unit.

As mentioned before, a unit aggregates authors in a one-to-one relationship. This means an author can only belong to single a unit at the same time.

This section described the domain model of the system and the rationale behind each entity and its attributes. The relationships among entities were also explained in detail. Next section will approach the architecture of system.

4.3 Architecture

This section will describe the architecture of the system, focusing the description according to the dependency view. For that, the main modules of the system are also identified and explained.

![Dependency View of the Architecture](image)

*Figure 4-6: Dependency view of the architecture of the system. Each arrow establishes a "uses" relationship from the start to the end of the arrow. For instance, the RESTful Services module uses the Data Access module.*

As it can be seen in figure 4-6, the implemented solution is composed of 5 different modules:
4.4. **USED TECHNOLOGY**

- **Data Access**
  The data access module is responsible for accessing the data storage system to perform read and write operations regarding domain entities. When necessary, it is also responsible for making more elaborated queries in order to retrieve collections of domain entities according to some criteria.

- **Duplicate Detection**
  The duplicate detection module is responsible for the process of finding and consolidating into one the records that are duplicate descriptions of the same publication. The details of the duplicate search process will be described in section 4.7.

- **Import**
  The import module is responsible for the harvesting of metadata records from the different information sources. The specifics of the import process will be approached in section 4.6.

- **User Interface**
  The user interface module implements the GUI (Graphical User Interface) for the human to machine interaction with system. The system’s GUI will be approached in section 4.8.2.

- **REST Services**
  The REST Services module is responsible for the implementation of the machine-to-machine interface of the system. A full explanation of how it works and how it was implemented is provided in section 4.8.1.

### 4.4 Used Technology

This work uses the following technologies as part of the implemented system:

- **Hibernate**\(^1\): Hibernate is used as an object/relational mapping tool for persisting the domain entities into a MySQL database engine.

- **Java-bibtex**\(^2\): Java-bibtex is a \LaTeX\ parser library that simplifies the task of parsing records in this format. It also provides tools for formatting information into a indent version. A modified version of this library was used, as the available version at the time of implementation of the system contained a bug that prevented Masters Thesis records from being parsed.

---

\(^1\) [http://www.hibernate.org/](http://www.hibernate.org/)

\(^2\) [http://code.google.com/p/java-bibtex/](http://code.google.com/p/java-bibtex/)
• XOM\textsuperscript{3}:  
XOM is a library that simplifies XML documents manipulation.

• Vaadin\textsuperscript{4}:  
Vaadin is a framework that allows the creation of the graphical user interface for web applications.

• Resteasy\textsuperscript{5}:  
RESTEasy is a framework that allows the quick creation of REST web services by just specifying the endpoints and parameters of each service.

4.5 System Configuration

The implemented solution is designed as a system to be running continuously. However, at setup time, the system is empty and the task of loading information sources and units may become tedious and lead to errors. To avoid that situation, the system has two configuration files from which it reads the necessary data for loading the defined information sources and organizational units. Listings 4-1 and 4-2 show, respectively, examples of those configuration files.

Listing 4-1: Excerpt of information sources configuration file

```xml
<?xml version="1.0" encoding="UTF-8"?>
<informationsources>
  <informationsource>
    <dataFormat>BibTeX</dataFormat>
    <nextImportDate>2012-09-30T20:30:00+01:00</nextImportDate>
    <homepageURL>http://scholar.google.com</homepageURL>
    <importable>true</importable>
    <importClass>pt.ist.meic.tese.importer.ImportGoogleScholar</importClass>
    <maxAuthors>1</maxAuthors>
    <name>Google Scholar</name>
    <repeatFrequency>Minutes</repeatFrequency>
    <repeatValue>1</repeatValue>
    <retrievalURL>http://scholar.google.pt/citations?view_op=export_citations&amp;user=&lt;userID&gt;</retrievalURL>
    <userpageURL>http://scholar.google.com/citations?user=lt;userID&gt;</userpageURL>
  </informationsource>
</informationsources>
```

Listing 4-2: Excerpt of organizational units configuration file

```xml
<units>
  <unit>
```

\textsuperscript{3}http://www.xom.nu/
\textsuperscript{4}https://vaadin.com/home
\textsuperscript{5}https://resteasy.org
It is important to mention that both information sources and organizational units are only saved in the system if they are not already registered. This way the system prevents the possibility of having duplicated data that would induce an wrong behavior.

In order to provide flexibility to the system, there are other two configuration files. The first one, `duplicate_fields.cfg`, defines which fields in the metadata records will be used as criteria in the duplicate detection module (section 4.7). An example of this configuration file is shown in Listing 4-2.
CHAPTER 4. PROPOSED SOLUTION

listing 4-3.

```plaintext
# File to indicate which fields will be taken into account when consolidating records.
# Lines started by '#' are considered comments and ignored.

title
```

**Listing 4-3: Example of duplicate fields configuration file**

The other configuration file, `compliance.cfg`, defines the fields that must be present in a record for it to be considered BibTeX compliant. The file follows a specific format to specify the entry types that are supported, as well as the fields that are required even in the situation when only one of two fields are required. An excerpt of this configuration file is presented in listing 4-4.

```plaintext
# File to indicate which fields will be taken into account when checking for BibTeX compliance.
# Lines starting with [T] represent the entry type.
# Lines starting with [F] represent a mandatory field for that type.
# A [F] that contains a "|" separator means that only one of the fields is mandatory.
# Lines started by '#' are considered comments and ignored.
# Each entry type should be separated by a line with '−'.

[T] article
[F] title
[F] journal
[F] year

−

[T] book
[F] title
[F] publisher
[F] year

−

[T] booklet
[F] title
```

**Listing 4-4: Excerpt of BibTeX-compliance configuration file**

4.6 Harvest

As mentioned in chapter 3, the envisioned system would have the functionality of automatically harvest descriptive metadata records from external services. This section provides an in-depth description of the harvest process, including the technical justification for some of the attributes described in different entities in section 4.2.
In order to harvest metadata records describing the publications of an author from an information source, two things are necessary:

1. The format of the URL from where to retrieve the document with the metadata records;
2. The identifier of the author in the desired information source.

Recalling the domain description in section 4.2, this information is available from, respectively, the information source entity and the identifier entity that is part of the association between an author and an information source.

After acquiring that information, the harvesting processing is as follows:

- Fetch the document using the specific URL for the author’s publications;
- Transform the document into a BibTeX format;
- Parse the BibTeX document;
- For each parsed record, check if it is new or a modified version of an already existent record and save it.

As seen in section 3.2.3 and appendix A, the retrieved documents from each information source are not homogeneous. All of them require some transformation in order to be parsed by the BibTeX library.

Even with document transformations, sometimes it is not possible to parse a metadata record. This work follows a best effort approach: it parses one record at a time, discarding just the ones it cannot parse, instead of all the document.

After a successful parsing of a record, the system tries to insert it into the database. Two things can happen: 1. it is a new metadata record harvested from the given information source; 2. it is a metadata record that was harvested in a previous import process for the same information source.

In order to make this distinction the system tries to find, from the set of all publications of the given author in the given information source and with the same publication type, another record that is considered a duplicate of the added one. If such record does not exist, it is considered a new record and saved into the database with the current date/time as the first import date (see section 4.2.3). On the other hand, if the system finds an existent record matching the criteria, the harvested record is considered another version of the existent one. In this situation, the system checks if there is any modification comparing to the existent record, and if it exists, replaces the record and sets the last modification date (see section 4.2.3). Figure 4-7 summarizes the harvesting process of an author’s metadata records in a given information source.
In terms of implementation, the importation module is organized as represented in figure 4-8.

Having the depicted organization, with one class responsible for all manipulations required on the retrieved documents from each information source, the importation module acquires high modularity, ideal if one should want to extend the system to support harvesting from more information sources. It is important to mention that the value of the import class attribute described in section 4.2.2 is the fully qualified name of the implemented class in the described hierarchy. For instance, the information source "Google Scholar" has the fully qualified name of the class pt.ist.meic.tese.importer.ImportGoogleScholar as its import class attribute. This design enhances abstraction, as it allows for one class to be responsible for the manipulation of documents retrieved from different information sources, if it is found that all of them require the same transformations.

After being transformed into BibTeX format, the document is passed to the class ImportFromBibTeX which is responsible for the parsing and creation of the domain entities. As mentioned before, during the creation process, the system verifies if such publication was already present in data storage.

Since the system provides a graphical user interface, it recurs to a enumerate to generate the representation of the class name to the user. Therefore, when adding a new information source, it is necessary to add it to ImportStrategies enumerate for it to appear as an option in the user interface.
Figure 4-7: Flowchart summarizing the harvesting process of the metadata records regarding the publications of a given author in a given information source.
Figure 4-8: Domain model of the Importation module
4.6.1 Scheduling

So far, the description of the system showed that it is possible to automatically harvest metadata records from different information sources and in different formats. However, such process requires action from the user to start. Research question 1 (see section 1.2) sets as an objective the minimal human effort in the harvesting process.

The system tries to achieve this goal by fully automatizing the harvesting the following way: as an information source is added to the system, a periodic timer associated to that source is activated. When the defined period ends, the harvesting process is automatically initiated.

However, giving the possible constraints when requiring documents from information sources (for instance, an external system may block requests with the same origin if they happen in a large number over a small period of time), it becomes necessary to parametrize the frequency of the harvesting processes from information sources. The used parameters are part of the information source entity (section 4.2.1):

- Max authors;
- Next harvest date;
- Repeat value;
- Repeat frequency.

By using this information, it is possible to define that the harvesting process for a given information source will start at a given date and time (next harvest date), and repeat as defined (repeat value, e.g., 7, repeat frequency, e.g. days). In each harvest, only records from at most max authors are retrieved. Following this approach allows the system administrator to define the parameters that better suit each information source.

One important aspect to take into account is how to choose the authors whose metadata records will be harvested. If done carelessly, the choice might provoke a starvation situation, where records from some authors will not be retrieved. The implemented system deals with this situation by saving the date and time when an identifier was last used. This way, the chosen authors for the harvesting process are the ones with the oldest records.

4.7 Duplicate Detection

Research question 3 (section 1.2) states as a need the identification of possible duplicates among the harvested records. This section exposes how the duplicated detection module contributes for
answering the question. It will start by describing the algorithm that is used for detect duplicate records and when it is invoked. Afterwards, it explains the records consolidation mechanism. Finally, it explores the potential performance issues and how they are solved by the system.

The duplicate detection algorithm receives two publication records and returns true if they are considered duplicate, and false otherwise. It starts by checking if the two publications are equal, that is, if they cumulatively meet the following requirements:

- Both records are of the same type (e.g., article);
- Both records have the same amount of fields;
- Both records have the same fields with the same value.

If both records are not considered equal, the system uses the configuration file that contains which fields are to be considered when making comparisons (section 4.5) to evaluate the records, and compares the values of those fields. It may happen that a record does not contain a field and therefore, in this situation, it is not possible to draw conclusions regarding the duplicate status, because the missing value could be the one establishing the difference. Summarizing, two records are considered duplicates if and only if they both have the same value for all the specified fields.

The duplicate detection module is invoked in two different situations:

1. When a new record is being added to the system, in order to know if that record is another version of an existing one. In this situation it is not enough to check for equality, since the new version might have new fields.

2. At the end of each harvesting process, when the record consolidation mechanism is invoked. This mechanism will be described next.

With the previously described algorithm it is possible to identify the duplicate records that exist in the system. However, the goal is to generate a consolidated list of records. In order to achieve that, while creating the list of consolidated records, the system checks if it is adding a duplicated record. If so, the existing fields are merged in the record that was already in the list. This way, it is possible to enrich to records with all available information and generate the most informative record. This algorithm has the drawback of being of quadratic growth. To minimize the consequences of this fact, an approach is presented next.

With an elevated number of records in the system, the consolidation process is expected to take longer to complete. In that case, generating the consolidated list every time it is requested would render the system unusable due to long response times. To solve that problem the system generates cache files with the consolidated records lists. The key aspect to take into account is when
to update the cache. The implemented solution follows this rationale: the state of the system, that is, the records that are stored can only change when an harvesting process occurs. In between importations, the records are the same and do not change. Taking advantage of that fact, there is the guarantee that consolidated lists generated between importations will always contain the same results. Therefore, the system will generate cache files right after each harvesting and use them when a client requests a consolidated list. The refresh of the cache is also necessary when an entity is deleted, since the records are not in the system anymore but the cache files still consider them.

With this approach, the system is expected to be much faster when answering requests for consolidated data.

This section described the functionalities of the duplicate detection module, including the detection algorithm and when it is invoked. It also described the way how lists of consolidated records are generated and how the problems of performance when such lists are requested are solved.

### 4.8 Interfaces

The implemented solution tries to provide different interfaces for communication with outside parties, as research question 2 (section 1.2) interrogates. This section describes the implemented interfaces and their goal.

#### 4.8.1 REST

In order to facilitate the access to the stored information, the system provides a machine-to-machine functional interface using RESTful services (section 2.5.2.1).

The defined services are organized in an hierarchical structure of endpoints, with clean URLs that provide a meaning for other developers that build systems that interact with the defined services.

The implemented services can be divided into three categories:

1. Access services, that correspond to HTTP GET requests, and intend to retrieve information contained in the system, such as a list of authors or a list of consolidated records. Given that some of those services require additional information, such as an author ID, those parameters are passed through URL. Since these are services that ask for information, there is no risk in repeated requests changing the state of the system directly. However, if it is a request to force an import process, the records in the system will change accordingly.

2. Addition services, corresponding to HTTP POST requests, that intend to add information to
the system. This includes services for adding a new unit or a new author into the system, for instance. The information regarding the added entity is passed through the body of the request. This body is verified before any change is committed, so that only valid requests are accepted.

3. Deletion services, dedicated to delete some entity in the system. These services correspond to HTTP DELETE requests and permanently delete the designated entity, which may involve cascading deletions. For instance, deleting an author implies that all his identifiers and records are also deleted, and deleting a unit implies deleting all the authors that belonged to the specified unit are deleted.

To provide better technical comprehension regarding the specified RESTful services, a full description of the defined endpoints and correspondent purpose is available in appendix B.

4.8.2 Graphical User Interface

Interacting with the system only through RESTful services may become a tedious and error-prone task if the goal is related to stewardship over the contained information, usually performed by librarians or system administrators. Because of that, a simplistic graphical user interface was developed that provides access to all functionalities of the system so that librarians and administrators might have a more general view of the system, such as the total of registered authors, units, information sources and publications. It is also possible to have an idea regarding the quality of an information source based on the compliance with standards of the harvested records.

The following figures show some of the main screens available in the graphical user interface of the system.
Figure 4-9: Homepage of the system showing overall statistics over the data.

Figure 4-10: Page showing all authors and respective records counts, grouped by unit.
CHAPTER 4. PROPOSED SOLUTION

Figure 4-11: Window to add a new author.

(a) From information Source  
(b) From file

Figure 4-12: Windows to import publications.

Figure 4-13: Window showing all information about an author: aliases, identifiers, unit, harvested records.
4.8. INTERFACES

Figure 4-14: Window to download a BibTeX file with the records harvested from the selected source.

Figure 4-15: Window listing all registered units.
CHAPTER 4. PROPOSED SOLUTION

Figure 4-16: Window to add a new unit.

Figure 4-17: Window listing all registered information sources.
4.8. INTERFACES

(a) Non importable

(b) Importable

Figure 4-18: Windows to add a new information source.

Figure 4-19: Window listing all available REST services, allowing their invocation.
Chapter 5

Validation

The validation chapter of this document aims at describing the procedures taken to validate the results that will sustain the conclusions. It starts by presenting the setup used for the executed tests, and the scenarios and metrics that were recorded. It then states the obtained results for each of the mentioned scenarios and finishes with a discussion of the results.

5.1 Validation Setup

All tests were executed on a localhost server, therefore network latency times are minimal. Also, all presented execution times are recorded within the system using logs to keep track of executed actions so that they do not depend on external factors.

The machine on which the tests were run has the following specification:

- Intel Core i7 - 2670QM @2.2GHz CPU
- 8GB of RAM memory
- 64bit operating system

All tests were performed using Postman - REST Client\(^1\), a plug-in for Google Chrome browser that allows interaction using RESTful services.

\(^1\)https://chrome.google.com/webstore/detail/postman-rest-client/fdmmgilgmpjigdojojojoooidkmomcm?utm_source=chrome-ntp-icon
5.2 Scenarios and Metrics

Since the system is expected to interact with external parties using the machine-to-machine technologies, the graphical user interface, expected to be used mostly by librarians and system administrators, was not subjected to evaluation.

The following scenarios, with respective metrics, will be evaluated using, when appropriate, a subset of authors from IST, consisting of 10 randomly chosen faculty members of Departamento de Engenharia Informática.

Scenario: Add information sources to the system.
Metric: Time to add different cardinalities of information sources.

Scenario: Add units to the system.
Metric: Time to add different cardinalities of units.

Scenario: Add authors to the system.
Metric: Time to add different cardinalities of authors.

Scenario: Harvest records of authors from different information sources.
Metric: Time to harvest records, grouped by information source.

Scenario: Consolidation of records.
Metric: Result of consolidate two records.
Metric: Time to consolidate records after harvest vs number of records to consolidate.

Scenario: Access to records.
Metric: Time to access all consolidated records in the system.

5.3 Results

This section presents the obtained results when executing the previously described scenarios, featuring a detailed description of what is being evaluated and why.
5.3. RESULTS

One important aspect of the system is its scalability. The first tests intend to evaluate how the system performs in situations of inserting an increasing number of instances of entities.

5.3.1 Add information sources

Figure 5-20 shows the results of adding information sources to the system. The x-axis represents the number of information sources inserted using the same HTTP request, and the y-axis the time (in milliseconds) required to add all specified information sources. The presented table contains the raw data of the results and the average time to insert an information source in each request.

<table>
<thead>
<tr>
<th>Total of instances</th>
<th>Total Time (ms)</th>
<th>Time per instance (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>13</td>
<td>6.5</td>
</tr>
<tr>
<td>3</td>
<td>18</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>34</td>
<td>8.5</td>
</tr>
<tr>
<td>5</td>
<td>26</td>
<td>5.2</td>
</tr>
<tr>
<td>10</td>
<td>49</td>
<td>4.9</td>
</tr>
<tr>
<td>25</td>
<td>118</td>
<td>4.72</td>
</tr>
<tr>
<td>50</td>
<td>251</td>
<td>5.02</td>
</tr>
<tr>
<td>100</td>
<td>423</td>
<td>4.23</td>
</tr>
</tbody>
</table>

Overall Average 5.79 (ms)

Figure 5-20: Time to insert information sources.

A linear tendency is observed when the number of instances increases, which means the time to add a single information source is approximately constant. That can be confirmed by looking at the average times of insertion. This behaviour will be further analysed in section 5.4.

5.3.2 Add units

Figure 5-21 shows the results of adding units to the system. The x-axis represents the number of units inserted using the same HTTP request, and the y-axis the time (in milliseconds) required to add all specified units. The presented table contains the raw data of the results and the average time to insert a unit in each request.

Similarly to the behaviour when adding information sources (section 5.3.1), a linear tendency is observed when the number of instances increases. The same conclusions may be drawn, as the
average time to add each unit tends to be constant, as the average values show. This behaviour will be further analysed in section 5.4.

5.3.3 Add authors

The last of the tests related purely with scalability features the evaluation of adding authors to the system. In figure 5-22, the x-axis represents the number of authors inserted using a single HTTP request, and the y-axis the time (in milliseconds) required to add all specified authors. The presented table contains the raw data of the results and the average time to insert an author in each request.

As it was observed in the other scalability tests, a linear tendency is observed. When the number of added authors increases, the average time to add each author is approximately constant, as it can be seen from the raw data presented in the table. This behaviour will be discussed in section 5.4.

The next evaluation tests will focus on the harvest and record consolidation functionalities of the system.
5.3. RESULTS

<table>
<thead>
<tr>
<th>Total instances</th>
<th>Total Time (ms)</th>
<th>Time per instance (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>5</td>
<td>81</td>
<td>16.2</td>
</tr>
<tr>
<td>10</td>
<td>106</td>
<td>10.6</td>
</tr>
<tr>
<td>50</td>
<td>471</td>
<td>9.42</td>
</tr>
<tr>
<td>100</td>
<td>915</td>
<td>9.15</td>
</tr>
<tr>
<td>250</td>
<td>2615</td>
<td>10.46</td>
</tr>
<tr>
<td>500</td>
<td>4655</td>
<td>9.30</td>
</tr>
<tr>
<td>1000</td>
<td>8156</td>
<td>8.156</td>
</tr>
<tr>
<td>2500</td>
<td>18083</td>
<td>7.2332</td>
</tr>
<tr>
<td>5000</td>
<td>33381</td>
<td>6.6762</td>
</tr>
<tr>
<td>10000</td>
<td>80715</td>
<td>8.0715</td>
</tr>
<tr>
<td>Overall Average</td>
<td>10.03 (ms)</td>
<td></td>
</tr>
</tbody>
</table>

Figure 5-22: Time to insert authors.

5.3.4 Harvest records

This test intends to analyse the process of harvesting records from different information sources for a set of 10 randomly chosen authors. All authors have an identifier in each of the information sources, so that a real comparison is possible. In figure 5-23, the x-axis represents the number of harvested records and the y-axis the time required to harvest the records. Each information source is represented by a different plot as described in the chart legend.

From the results of the tests, three things may be concluded:

1. For the same author, different information sources have a different number of records associated to that author;

2. As the number of harvested records increases, the time of the process also increases;

3. The time to harvest also depends on the information source, not only on the number of records.

Further discussion about these conclusions is presented in section 5.4.
5.3.5 Consolidate records

The purpose of the record consolidation test is to analyse the efficiency of the consolidation algorithm. In figure 5-24, the x-axis represents the total number of records in the system, that is, the number of records to consolidate, and the y-axis the time required to perform the task, in seconds.

From the presented chart, it is possible to conclude that the growth in time is largely affected by the number of records to consolidate. This conclusion will be focus of a deeper analysis in section 5.4.

The following listings show the result of the consolidation process for one publication. Having
5.3. RESULTS

harvested one record from an information source (listing 5-1), another record (listing 5-2) was added to the system through file import. As it can be seen, both records represent the same publication. Listing 5-3 shows the result of the consolidation, where the record contains the set of all fields, ignoring the repeated ones.

```latex
@inproceedings{martins2008extracting,
title = {Extracting and exploring the geo–temporal semantics of textual resources},
author = {Martins, B. and Manguinhas, H. and Borbinha, J.},
booktitle = {Semantic Computing, 2008 IEEE International Conference on},
pages = {1--9},
year = {2008},
organization = {IEEE}
}
```

Listing 5-1: Record retrieved from Google Scholar

```latex
@inproceedings{martins2008extracting,
title = {Extracting and exploring the geo–temporal semantics of textual resources},
author = {Martins, B. and Manguinhas, H. and Borbinha, J.},
pages = {1--9},
organization = {IEEE},
isbn = {978--0--7695--3279--0},
doi = {10.1109/ICSC.2008.86},
publisher = {IEEE Computer Society},
}
```

Listing 5-2: Record imported from file

```latex
@inproceedings{Martins:Manguinhas:Borbinha:2008,
author = {Martins, B. and Manguinhas, H. and Borbinha, J.},
booktitle = {Semantic Computing, 2008 IEEE International Conference on},
doi = {10.1109/ICSC.2008.86},
isbn = {978--0--7695--3279--0},
organization = {IEEE},
pages = {1--9},
publisher = {IEEE Computer Society},
title = {Extracting and exploring the geo–temporal semantics of textual resources},
year = {2008}
}
```

Listing 5-3: Result of consolidating duplicate records

5.3.6 Get records

Figure 5-25 relates the total number of records in the system (x-axis) with the time in milliseconds required to generate and send the response to the client (y-axis). Opposite to the consolidation
test, the results of accessing all existent records show a linear trend. This result will be further analysed in section 5.4, within the context of harvesting and providing records.

In figure 5-26, where the results of the two previous tests (sections 5.3.5 and 5.3.6) are compared using the same units to represent time – seconds –, we can see that the access time is nearly zero compared to the consolidation time. This is one of the most important results in our analysis and will be subject of a deep explanation in section 5.4.
5.4 Discussion

This section presents a discussion of the results obtained with the validation tests.

5.4.1 Scalability

Regarding scalability tests, we can see that the time required to add instances of entities grows linearly with the number of created instances. This leads us to assume that the system will have a behaviour within acceptable response times, even when adding a large number of authors.

For instance, inserting 10000 authors (when the full name is optional) takes about 80 seconds, which gives \( \sim 8 \) milliseconds per author, contrasting to the overall average of \( \sim 10 \) milliseconds per author. A curious observation is that adding a small number of entities takes longer per entity. One possible reason for this behaviour may be related to the connection mechanism to the underlying database engine. As the number of instances increase, the system may use already existent connections and therefore no time is lost opening new ones.

Similar results, where the average time of adding an instance while inserting a large number of entities is lower than the overall average, were also obtained in the tests of units and information sources addition.

5.4.2 Harvest

As mentioned in the description of figure 5-23, different information sources have a different number of records associated to the same author. This shows us that the usage of one information source instead of another has a big impact on the quantity of harvested information.

As demonstrated in the figure, Google Scholar and Microsoft Academic Search are the services that provide more records. They are also the services for which it takes the less time to harvest the same amount records. Two factors may influence these results:

- The transformations and processing required on the retrieved document. Google Scholar and Microsoft Academic Search already provide the records in \( \text{BibTeX} \) format, resulting in less operations over the document. On the other hand, DBLP requires a request for each record, which translates into a considerable latency time, and ACM Digital Library provides a document that requires the removal of HTML tags.

- The infrastructure and computational resources of Google and Microsoft should be larger, compared to the other tested information sources. Besides that, it is known that these organizations have powerful search engines that may contribute to a broader reach when it
In line with these observations, we can assert that the most important information sources in our system are Google Scholar and Microsoft Academic Search, due to the number of available records.

### 5.4.3 Consolidation vs Access

The evaluation regarding the consolidation of and access to the records in the system (sections 5.3.5 and 5.3.6) shows us that the consolidation process takes a long time to execute. As mentioned in section 4.7, with more records in the system, it will take longer to consolidate all of them. In fact, the implemented algorithm has a complexity of $O(N \times M)$ where $N$ is the total number of records in the system and $M$ is the number of consolidated records.

Given this complexity of the consolidation algorithm, the observed behaviour was expected and will force a cautious configuration of the system when in production, regarding the frequency of harvests from different information sources, as a harvest process might cause a long de-duplication process. A possible contribution to minimize this issue relies on indexing the harvested data, as mentioned in section 6.1.

In figure 5-25 we can see the time required to fetch all records in the system. This shows that the access to the records follows a linear growth as expected, since the consolidation work has already been done, and represents the consequence of the cache mechanism described in section 4.7.

Comparing both consolidation of and access time to all records in the system (see figure 5-26), we can see that the access times are almost negligible when compared to the consolidation times. From this chart, we can conclude that the effort in consolidation generates a high reward when accessing the data, going in line with the expected behaviour described in section 4.7.

The results of these tests show the consequences of the flexibility introduced by the configurable duplicate detection criteria (see section 4.5). Therefore, we must set for a trade-off between flexibility and execution time. If the duplicate criteria was not configurable, the consolidation process would be much faster, but we would lose the possibility of choosing which fields to consider when de-duplicating.
Chapter 6
Conclusions

In this thesis, it was studied the problem of bibliographic metadata harvesting as a support for institutional repositories management, following a semi-automatic process.

A system was developed, based on the needs identified by the research questions, that aimed at, given the necessary information, automatically harvest the metadata records of an author’s publications from different information sources, identify and consolidate the harvested records that are considered duplicates, generating the richest record possible, and make available the existent information to outside parties that might pretend to use the system’s functionalities. The system aimed at the goal of computing some basic statistics regarding the available information that might be useful for the institutional repository management, by librarians and system administrators. Finally, it was considered the possibility of further development with minimal effort, so that other functionalities, such as more information sources support, may be added.

The implemented system was used in a real context environment, using real metadata records of a sample of authors at Instituto Superior Técnico, harvested from the supported information sources. Based on the results presented in section 5, we can conclude that the implemented system is capable of:

1. Harvest metadata records from external sources and update the contents of the system, without introducing unnecessary duplicated records.

2. Share stored information taking advantage of a technical solution that uses the available technology for such interoperability.

3. Identify and consolidate duplicate records into a smaller collection in which each record is an information-enriched version of all the records identified as duplicates.
CHAPTER 6. CONCLUSIONS

However, as shown in figure 5.3.5 and discussed in section 5.4.3, the consolidation of the harvested records takes a long time. We accept this drawback because we assume that the system will focus on serving information instead of retrieving it. With the implemented solution, even with long times for consolidation, the access to the information by any external stakeholder is almost immediate.

Concluding, with the existence of the presented architecture implemented in a real system, it is possible for an institutional repository to abstract from the multitude of external services that provide metadata records, and use the implemented solution as a proxy that has aggregation capabilities to harvest metadata records. It is also possible to abstract from the problem of identifying and consolidating duplicate records within a large collection.

6.1 Future Work

Even though the implemented system provides an answer to the research questions, there is still room for several improvements. Some possible extensions to the system include:

- Introduce support to other descriptive metadata schemas, such as Dublin Core (section 2.4.4) or RIS\textsuperscript{1}, that allow for a larger set of publication types and more descriptive fields. With the current architecture, this improvement would be relatively easy to implement, by only requiring changes in the methods that access the fields of the record in order to be able to distinguish among the implemented descriptive metadata schemas. For that, a parser for the specified format would need to be included or developed.

- Implement support for more information sources. In this work, only four external services were used to harvest metadata records but, as seen in section 3.2.3, there are more services from where to harvest records, such as ResearcherID and Mendeley. This involves extending the identifier entity to include login details, as Mendeley requires OAuth to provide a response. It would also require the inclusion of a JSON parser to process the harvested records. As for ResearcherID, support for it would require an HTML parser, possibly taking advantage of XPath or algorithms that deduce the structure of the page and find the information, and a way of simulating the change of page.

- Develop a different algorithm to detect duplicate records since that, at the moment, the system checks for equality between field values. Another possibility is to use one of the similarity measures presented in section 2.6.3. This way, even if a record misses a small word such as a pronoun in the title, it would still be considered a duplicate as it is supposed to.

\textsuperscript{1}http://www.refman.com/support/risformat_intro.asp
6.1. FUTURE WORK

- Add an indexing engine such as Hibernate Search\(^2\) or Solr\(^3\), both build on top of Lucene\(^4\), that add search capabilities to the system, as well as other ways of improving the detection and consolidation of duplicate records, and therefore reducing the time required to complete that task.

\(^2\)http://www.hibernate.org/subprojects/search.html
\(^3\)http://lucene.apache.org/solr/
\(^4\)http://lucene.apache.org/
Bibliography


COSTELLO, R.L. Building web services the REST way.


Appendix A

Information Sources

This appendix presents a detailed analysis of each service that was considered as a possible Information Source in the scope of this work. For each one, the analysis contains the following information:

- **Name** - Name of the Information Source;
- **Homepage URL** - URL for the homepage of the Information Source;
- **Domain** - Single or Multiple areas of research;
- **Author Identifier** - If it provides a unique identifier for each Author;
- **Author profile URL format** - The format of the URL for the Author profile page;
- **Records retrieval URL format** - The format of the URL for retrieve the list of metadata records;
- **Data format** - The format in which the records are returned (e.g., BibTeX, XML, HTML, ...);
- **Example of retrieved records** - A small example of the metadata records as returned by the Information Source;
- **Required document manipulation** - A description of the changes that need to be done to the retrieved document to allow its processing;
- Other relevant information, such as examples of subsequent requests.

A.1 Google Scholar

**Name**: Google Scholar
APPENDIX A. INFORMATION SOURCES

Homepage URL: http://scholar.google.com/

Domain: Multiple areas of research

Author Identifier: Yes

Author profile URL format: http://scholar.google.com/citations?user=<userID>

Records retrieval URL format: http://scholar.google.com/citations?view_op=export_citations&user=<userID>

Data format: BibTeX

Example of retrieved records:

```
@inproceedings{martins2008extracting,
    title={Extracting and exploring the geo–temporal semantics of textual resources},
    author={Martins, B. and Manguinhas, H. and Borbinha, J.},
    booktitle={Semantic Computing, 2008 IEEE International Conference on},
    pages={1--9},
    year={2008},
    organization={IEEE}
}

@article{ferreira1997using,
    title={Using LDAP in a Filtering Service for a Digital Library},
    author={Ferreira, J. and Borbinha, J.L. and Delgado, J.},
    journal={5th DELOS Workshop, Budapest},
    volume={5},
    year={1997}
}

@article{manguinhas2009digmap,
    title={The DIGMAP geo–temporal Web gazetteer service},
    author={Manguinhas, H. and Martins, B. and Borbinha, J. and Vaca, S. and Libardo, W.},
    year={2009},
    publisher={e–Perimeton}
}
```

Listing A-1: Example of retrieved metadata records from Google Scholar

Required document manipulation: No manipulation is required.

A.2 Microsoft Academic Search

Name: Microsoft Academic Search

Homepage URL: http://academic.research.microsoft.com/
Domain: Multiple areas of research

Author Identifier: Yes

Author profile URL format: http://academic.research.microsoft.com/Author/<userID>

Records retrieval URL format: http://academic.research.microsoft.com/<userID>.bib?type=10&entitytype=2&format=0&download=1

Data format: BibTeX

Example of retrieved records:

```
@article{
    author = {José Barateiro and Gonçalo Antunes and José Borbinha},
    title = {{Addressing Digital Preservation: Proposals for New Perspectives}},
    year = {2009},
    masid = {4871812}
}

@inproceedings{
    author = {Bruno Martins and Hugo Manguinhas and José Luis Borbinha},
    title = {{Extracting and Exploring the Geo-Temporal Semantics of Textual Resources}},
    booktitle = {International Computer Science Conference},
    year = {2008},
    pages = {1--9},
    doi = {10.1109/ICSC.2008.86},
    masid = {4351166}
}

@book{
    author = {Nuno Freire and Jose Luis Borbinha and Bruno Martins},
    title = {{Consolidation of References to Persons in Bibliographic Databases}},
    booktitle = {International Conference on Asian Digital Libraries},
    year = {2008},
    pages = {256--265},
    doi = {10.1007/978-3-540-89533-6_26},
    masid = {4267888}
}
```

Listing A-2: Example of retrieved metadata records from Microsoft Academic Search

Required document manipulation: Minimal - Insert citation key into entries to allow parsing.
A.3 ACM Digital Library

Name: ACM Digital Library

Homepage URL: http://dl.acm.org

Domain: Single Domain - Computing

Author Identifier: Yes

Author profile URL format: http://dl.acm.org/author_page.cfm?id=<userID>

Records retrieval URL format: http://dl.acm.org/authorBibTex.cfm?query=Author:<userID>

Data format: BibTeX

Example of retrieved records:

<pre>
&lt;pre id="2116261.2117288"&gt;@inproceedings{2117288,
    author = {Barateiro, Jose and Antunes, Goncalo and Borbinha, Jose},
    title = {Manage Risks through the Enterprise Architecture},
    booktitle = {HICSS '12: Proceedings of the 2012 45th Hawaii International Conference on System Sciences},
    year = {2012},
    isbn = {978–0–7695–4525–7},
    pages = {3297--3306},
    doi = {http://dx.doi.org/10.1109/HICSS.2012.419},
    publisher = {IEEE Computer Society},
    address = {Washington, DC, USA},
}
&lt;/pre&gt;

&lt;pre id="2336457.2336459"&gt;@article{2336459,
   author = {Manguinhas, Hugo and Freire, Nuno and Machado, Jorge and Borbinha, Jos' e},
   title = {Supporting multilingual bibliographic resource discovery with functional requirements for bibliographic records},
   journal = {Semant. web},
   volume = {3},
   number = {1},
   year = {2012},
   issn = {1570–0844},
   pages = {3--21},
   doi = {http://dx.doi.org/10.3233/SW-2012-0046},
   publisher = {IOS Press},
   address = {Amsterdam, The Netherlands, The Netherlands},
}
&lt;/pre&gt;
</pre>
A.4 DBLP

Name: DBLP

Homepage URL: http://dblp.uni-trier.de/

Domain: Single Domain - Computing

Author Identifier: Yes

Author profile URL format: http://dblp.uni-trier.de/db/indices/a-tree/<userID>.html

Records retrieval URL format: http://dblp.uni-trier.de/rec/pers/<userID>/xk

Data format: XML

Example of retrieved records:

```xml
<xml version="1.0"?>
<dblperson name="José Luis Borbinha">
<dblpkey type="person_record">homepages/27/71</dblpkey>
<dblpkey>journals/semweb/ManguinhasFMB12</dblpkey>
<dblpkey>conf/esws/FreireBC12</dblpkey>
<dblpkey>conf/hicss/BarateiroAB12</dblpkey>
<dblpkey>conf/jcdl/FreireBC12</dblpkey>
<dblpkey>conf/dgo/BeckerABVB11</dblpkey>
<dblpkey>conf/dgo/VieiraBVV11</dblpkey>
</xml>
Listing A-4: Example of retrieved metadata records listing from DBLP

Required document manipulation: Major - Subsequent requests for each record; transformation of XML to $\text{B\kern-1.5ptE\kern-1.5ptX}$ language.

Listing A-5: Example of retrieved metadata record from DBLP of a publication

A.5 ResearcherID

Name: ResearcherID

Homepage URL: http://www.researcherid.com/

Domain: Multiple areas of research

Author Identifier: Yes

Author profile URL format: http://www.researcherid.com/rid/<userID>

Records retrieval URL format: http://www.researcherid.com/rid/<userID>

Data format: HTML
Example of retrieved records:

<table>
<thead>
<tr>
<th>Author(s)</th>
<th>Title</th>
<th>Volume</th>
<th>Pages</th>
<th>Published</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>FREIRE, N; BORBINHA, J; CALADO, P; et al.</td>
<td>Identification of FRBR works within bibliographic databases: An experiment with UNIMARC and duplicate detection techniques</td>
<td>4822</td>
<td>267–276</td>
<td>2007</td>
<td>ASIAN DIGITAL LIBRARIES: LOOKING BACK 10 YEARS AND FORGING NEW FRONTIERS, PROCEEDINGS</td>
</tr>
</tbody>
</table>

Times Cited: 5
Listing A-6: Example of retrieved metadata records from ResearcherID

Required document manipulation: High - Transformation of HTML to \LaTeX; Subsequent requests for each page of records that requires simulation of user interaction.
### A.6 Mendeley

**Name:** Mendeley  
**Homepage URL:** http://www.mendeley.com/  
**Domain:** Multiple areas of research  
**Author Identifier:** Yes  
**Author profile URL format:** http://www.mendeley.com/profiles/<userID>  
**Records retrieval URL format:** Unknown  
**Data format:** JSON  
**Example of retrieved records:**  
Unknown  
**Required document manipulation:** High - Transformation of JSON to \texttt{BibTeX}; Other manipulations unknown.

### A.7 CiteSeer

**Name:** CiteSeer  
**Homepage URL:** http://citeseer.ist.psu.edu/index  
**Domain:** Multiple areas of research  
**Author Identifier:** Yes  
**Author profile URL format:** http://citeseer.ist.psu.edu/viewauth/summary?aid=<userID>  
**Records retrieval URL format:**  
http://citeseer.ist.psu.edu/viewauth/summary?aid=<userID>&list=full  
**Data format:** JSON  
**Example of retrieved records:**

```xml
<table class="refs" border="0" cellspacing="5" cellpadding="5">
  <tr>
    <td class="title">987</td>
  </tr>
</table>
```
<table>
<thead>
<tr>
<th>Title</th>
<th>Author(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>654</td>
<td>End-to-End Internet Packet Dynamics</td>
</tr>
<tr>
<td>627</td>
<td>Equation-Based Congestion Control for Unicast Applications</td>
</tr>
<tr>
<td>523</td>
<td>End-to-end routing behavior in the Internet</td>
</tr>
<tr>
<td>435</td>
<td>The bsd packet filter: A new architecture for user-level packet capture</td>
</tr>
</tbody>
</table>

Listing A-7: Example of retrieved metadata records from CiteSeer

**Required document manipulation:** High - Transformation of JSON to BibTeX; Subsequent requests for each publication.

```latex
@INPROCEEDINGS{Balakrishnan01resilientoverlay,
    author = {Hari Balakrishnan and Arthur C. Smith and David G. Andersen and David G. Andersen},
    title = {Resilient Overlay Networks},
    booktitle = {},
    year = {2001}
}```
Listing A-8: Example of retrieved metadata record from CiteSeer of a publication

```
8     &nbsp;&nbsp;&nbsp;&nbsp;pages = (131--145)<br/>
9     </p>
10    </div> <!-- End content box -->
11
```

Appendix B

RESTful Web Services

This appendix lists all defined endpoints for RESTful web services, including the URL, parameters explanation, the HTTP method required, the structure of the body when needed and an example of interaction.

B.1 Unit Services

B.1.1 List Units

Endpoint URL: /rest/unit/list
HTTP Method: GET

B.1.2 Get Unit

Endpoint URL: /rest/unit/unitID={unitID}
Parameters:

- {unitID} - internal identifier of the unit.

HTTP Method: GET

B.1.3 Add Units

Endpoint URL: /rest/unit/add
Parameters: None
HTTP Method: POST

Body:

```xml
<units>
  <unit>
    <acronym></acronym>
    <name></name>
  </unit>
<units>
```

HTTP Response: 201 if unit created, 400 otherwise.

Example:

```xml
<units>
  <unit>
    <acronym>DEI</acronym>
    <name>Departamento de Engenharia Informática</name>
  </unit>
<units>
```

Listing B-1: Example of body for Add Units service

### B.1.4 Update Unit

Endpoint URL: `/rest/unit/update/unitID={unitID}`

Parameters:

- `{unitID}` - internal identifier of the unit.

HTTP Method: POST

Body:

```xml
<unit>
  <acronym></acronym>
  <name></name>
</unit>
```

HTTP Response: 200 if unit updated, 400 otherwise.

Example:

```xml
<unit>
  <acronym>DECivil</acronym>
```
B.1.5 Delete Unit

**Endpoint URL:** /rest/unit/delete/unitID={unitID}

**Parameters:**

- `{unitID}` - internal identifier of the unit.

**HTTP Method:** DELETE

**HTTP Response:** 204 if unit deleted, 404 otherwise.

B.2 Information Source Services

B.2.1 List Information Sources

**Endpoint URL:** /rest/informationSource/list

**HTTP Method:** GET

B.2.2 Get Information Source

**Endpoint URL:** /rest/informationSource/informationSourceID={informationSourceID}

**Parameters:**

- `{informationSourceID}` - internal identifier of the information source.

**HTTP Method:** GET

B.2.3 Add Information Sources

**Endpoint URL:** /rest/informationSource/add

**HTTP Method:** POST

**Body:**

<informationsources>
APPENDIX B. RESTFUL WEB SERVICES

HTTP Response: 201 if information source created, 400 otherwise.

Example:

```
<informationsources>
  <informationsource>
    <dataFormat>BibTeX</dataFormat>
    <nextImportDate>2012-10-30T20:30:00+01:00</nextImportDate>
    <homepageURL>http://scholar.google.com/</homepageURL>
    <importable>true</importable>
    <importClass>pt.ist.meic.tece.importer.ImportGoogleScholar</importClass>
    <maxAuthors>1</maxAuthors>
    <name>Google Scholar</name>
    <repeatFrequency>Minutes</repeatFrequency>
    <repeatValue>1</repeatValue>
    <retrievalURL>http://scholar.google.pt/citations?view_op=export_citations&amp;user=&lt;userID&gt;</retrievalURL>
    <userpageURL>http://scholar.google.com/citations?user=&lt;userID&gt;</userpageURL>
  </informationsource>
</informationsources>
```

Listing B-3: Example of body for Add Information Sources service
B.2. INFORMATION SOURCE SERVICES

B.2.4 Update Information Source

**Endpoint URL:** /rest/informationSource/update/informationSourceID={ID}

**Parameters:**

- {ID} - internal identifier of the information source.

**HTTP Method:** POST

**Body:**

```xml
<informationsource>
  <name></name>
  <homepageURL></homepageURL>
  <userpageURL></userpageURL>
  <importable></importable>
  <retrievalURL></retrievalURL>
  <dataFormat></dataFormat>
  <importClass></importClass>
  <maxAuthors></maxAuthors>
  <nextImportDate></nextImportDate>
  <repeatValue></repeatValue>
  <repeatFrequency></repeatFrequency>
</informationsource>
```

**HTTP Response:** 200 if information source updated, 400 otherwise.

**Example:**

```xml
<informationsource>
  <dataFormat>BibTeX</dataFormat>
  <nextImportDate>2012-10-30T20:30:00+01:00</nextImportDate>
  <homepageURL>http://academic.research.microsoft.com/</homepageURL>
  <importable>true</importable>
  <importClass>pt.isi.teic.tese.importer.ImportMicrosoftAcademicSearch</importClass>
  <maxAuthors>1</maxAuthors>
  <name>Microsoft Academic Search</name>
  <repeatFrequency>Minutes</repeatFrequency>
  <repeatValue>2</repeatValue>
  <retrievalURL>http://academic.research.microsoft.com/&lt;userID&gt;.&bib?type=10&amp;entitytype=2&amp;format=0&amp;download=1</retrievalURL>
  <userpageURL>http://academic.research.microsoft.com/Author/&lt;userID&gt;</userpageURL>
</informationsource>
```

Listing B-4: Example of body for Update Information Source service
B.2.5 Delete Information Source

Endpoint URL:
/rest/informationSource/delete/informationSourceID={informationSourceID}

Parameters:

• {informationSourceID} - internal identifier of information source.

HTTP Method: DELETE
HTTP Response: 204 if information source deleted, 404 otherwise.

B.3 Author Services

B.3.1 List Authors

Endpoint URL: /rest/author/list
HTTP Method: GET
Example:

B.3.2 Get Author

Endpoint URL: /rest/author/authorID={authorID}
Parameters:

• {authorID} - internal identifier of the author.

HTTP Method: GET
Example:

B.3.3 Add Authors

Endpoint URL: /rest/author/add
HTTP Method: POST
Body:

<authors>
B.3. AUTHOR SERVICES

HTTP Response: 201 if author created, 400 otherwise.

Example:

Listing B-5: Example of body for Add Authors service

B.3.4 Update Author

Endpoint URL: /rest/author/update/authorID={authorID}

Parameters:

• {authorID} - internal identifier of the author.

HTTP Method: POST

Body:

<author unitID="">
  <authoritativeName></authoritativeName>
  <fullName></fullName>
</author>
</authors>

<author unitID="">
  <authoritativeName>José Tribolet</authoritativeName>
  <fullName>José Manuel Nunes Salvador Tribolet</fullName>
</author>
<author unitID="">
  <authoritativeName>João Pavão Martins</authoritativeName>
  <fullName>João Emílio Segurado Pavão Martins</fullName>
</author>
<author unitID="">
  <authoritativeName>José Alves Marques</authoritativeName>
  <fullName>José Manuel da Costa Alves Marques</fullName>
</author>
<author unitID="">
  <authoritativeName>Arlindo Oliveira</authoritativeName>
</author>
<author unitID="">
  <authoritativeName>Mário Gaspar da Silva</authoritativeName>
</author>
</authors>
HTTP Response: 200 if author updated, 400 otherwise.

Example:

```xml
<author unitID="7">
  <authoritativeName>Arlindo Oliveira</authoritativeName>
  <fullName>Arlindo Manuel Lime de Oliveira</fullName>
</author>
```

Listing B-6: Example of body for Update Author service

### B.3.5 Add Aliases

**Endpoint URL:** `/rest/author/addAliases/authorID={authorID}

**Parameters:**

- `{authorID}` - internal identifier of the author.

**HTTP Method:** POST

**Body:**

```xml
<aliases>
  <alias>Oliveira, A.</alias>
  <alias>Oliveira, A. M.</alias>
</aliases>
```

**HTTP Response:** 201 if aliases added, 404 if author does not exist, 400 otherwise.

**Example:**

```xml
<aliases>
  <alias>Oliveira, A.</alias>
  <alias>Oliveira, A. M.</alias>
</aliases>
```

Listing B-7: Example of body for Add Aliases service

### B.3.6 Set Identifier

**Endpoint URL:** `/rest/author/setIdentifier/authorID={authorID}&informationsourceID={informationsourceID}

**Parameters:**
B.3. AUTHOR SERVICES

• \{authorID\} - internal identifier of the author.

• \{informationSourceID\} - internal identifier of the information source.

**HTTP Method:** POST

**Body:**

```xml
<identifier></identifier>
```

**HTTP Response:** 201 if identifier created, 200 if identifier updated, 404 if author or information source does not exist, 400 otherwise.

**Example:**

```xml
<identifier>dqtEnaoAAAAJ</identifier>
```

Listing B-8: Example of body for Set Identifier service

---

B.3.7 Delete Author

**Endpoint URL:** /rest/author/delete/authorID={authorID}

**Parameters:**

• \{authorID\} - internal identifier of the author.

**HTTP Method:** DELETE

**HTTP Response:** 204 if author deleted, 404 otherwise.

---

B.3.8 Delete Identifier

**Endpoint URL:** /rest/author/delete/identifierID={identifierID}

**Parameters:**

• \{identifierID\} - internal identifier of the author’s identifier.

**HTTP Method:** DELETE

**HTTP Response:** 204 if identifier deleted, 404 otherwise.
Appendix B. Restful Web Services

B.4 Harvest Services

B.4.1 Harvest Records

Endpoint URL: /rest/harvest/authorID={authorID}&informationsourceID={informationsourceID}
Parameters:

- {authorID} - internal identifier of the author.
- {informationsourceID} - internal identifier of the information source.

HTTP Method: GET

B.5 Records Services

B.5.1 List Consolidated Records

Endpoint URL: /rest/record/list
Parameters: None
HTTP Method: GET

B.5.2 List Consolidated Records - BibTeX

Endpoint URL: /rest/record/list/bibtex
Parameters: None
HTTP Method: GET

B.5.3 Get Record

Endpoint URL: /rest/record/recordID={recordID}
Parameters:

- {recordID} - internal identifier of the record.

HTTP Method: GET
B.5.4 Get Record - BibTeX

Endpoint URL: /rest/record/recordID={recordID}/bibtex
Parameters:

• \{recordID\} - internal identifier of the record.

HTTP Method: GET

B.5.5 Get Author Consolidated Records

Endpoint URL: /rest/record/authorID={authorID}
Parameters:

• \{authorID\} - internal identifier of the author.

HTTP Method: GET

B.5.6 Get Author Consolidated Records - BibTeX

Endpoint URL: /rest/record/authorID={authorID}/bibtex
Parameters:

• \{authorID\} - internal identifier of the author.

HTTP Method: GET

B.5.7 Get IST Author Consolidated Records

Endpoint URL: /rest/record/istID={istID}
Parameters:

• \{istID\} - the IST ID of the author.

HTTP Method: GET

B.5.8 Get IST Author Consolidated Records - BibTeX

Endpoint URL: /rest/record/istID={istID}/bibtex
Parameters:
• `{istID}` - the IST ID of the author.

**HTTP Method:** GET

### B.5.9 Get Author Records

**Endpoint URL:** `/rest/record/authorID={authorID}&informationsourceID={informationsourceID}`

**Parameters:**

- `{authorID}` - internal identifier of the author.
- `{informationsourceID}` - internal identifier of the information source.

**HTTP Method:** GET

### B.5.10 Get Author Records - BibTeX

**Endpoint URL:**

`/rest/record/authorID={authorID}&informationsourceID={informationsourceID}/bibtex`

**Parameters:**

- `{authorID}` - internal identifier of the author.
- `{informationsourceID}` - internal identifier of the information source.

**HTTP Method:** GET