PhD Dissertation

International Doctorate School in Information and Communication Technologies

DIT - University of Trento

Okkam
Enabling Entity-centric Information Integration in the Semantic Web

Heiko Stoermer

Advisor:

Prof. Paolo Bouquet

Università degli Studi di Trento

January 2008

Abstract

Like to the way the WWW provided a global space for the seamless integration of small hypertexts into a global, open, decentralized and scalable publication space, the Semantic Web has the vision to provide a global space for the seamless integration of knowledge bases into a global, open, decentralized and scalable knowledge space, whose domain is extended beyond the realm of digital objects, and relations are extended beyond hyperlinks. But unlike the WWW, where reference to documents through hyperlinks is provided in an unambiguous and architectural way, the Semantic Web in its current state suffers from an identity and reference problem for the entities it describes, a problem which hinders significantly the integration of Semantic Web knowledge on the data level.

The cause for this problem is that to date there exists no suitable infrastructure which would enable the systematic re-use of global identifiers for entities, which could solve the mentioned issues by making some of the assumptions that the Semantic Web bases on a reality.

The objective of this work is thus to provide such an infrastructure. We will describe the notion of an Entity Name System (ENS) as a global approach for providing re-usable identifiers for entities. A prototype implementation of such an ENS – called OKKAM *– has been developed, is operational, and is part of the novel contributions of this work. Additionally, a set of new applications that make use of the ENS will be illustrated, as well as analyses and experimentation results which base on the ENS.*

The main contribution of this work is the provision of an ENS, an infrastructural component for the Semantic Web which enables a-priori information integration in a novel way, and enables the development of novel applications and information management paradigms.

Keywords
Semantic Web, Information Integration, Web of Entities, Knowledge Management, Identity and Reference

“It is a mistake to try to look too far ahead. The chain of destiny can only be grasped one link at a time.”

Sir Winston Churchill (1874 - 1965)

To my father.

Acknowledgments

The author would like to express his gratitude to the following people:

My supervisor, Prof. Paolo Bouquet, for being a true mentor.

Angela, for our future, wherever it may be.

My family. My mother, for never giving up, in her own way. Alberto and Ankatrin Giussani, for their unlimited support. Silia Giussani, for everything.

Robert Brickman, Anna Kunde, Joachim Fischer, Martina Zidek, Michael Härle, Sarah Kate Selling, Filippo Tosoratti, Amy Fountain, Paolo Marchiori, Tazneem Twells, Michele Porrino, for being my counter balance, and my partners in crime on many adventures.

Carsten Burghardt, for dragging me through my undergrad studies. Anja Paulmichl, for always helping out.

Peter Fiener, for showing me how many balls can be kept in the air.

Marco Braun, for many dark coffees in dark hours.

Claudia Niederée and Rodolfo Stecher for fruitful discussions and their indispensible input in the concept phase of this work.

The UNITN OKKAM group: Barbara Bazzanella for her support with *R*. Stefano Bortoli for his work on FOAF-O-MATIC. Daniel Giacomuzzi for his work on the OKKAM prototypes. Xin Liu and Daniele Zanoni for their work on OKKAM*4P*. Finally, Elisabetta Fauri for her management.

Ekaterini Ioannou, Antonio Maña, Luciano Serafini, Andrei Tamilin and Andreas Wombacher for discussions and pointers to literature in their respective areas of expertise.

Profs. Norbert Gerth, Wolfang Kowarschick, Michael Lutz and Christian Märtin, for recommending me for the PhD.

My Final Exam Committee, Prof. Sonia Bergamaschi, Prof. Frank van Harmelen and Dr. John O'Flaherty, for their numerous suggestions and critical evaluation of my work.

This research was partially funded by the European Commission under the 6^{th} Framework Programme IST Integrated Project VIKEF - Virtual Information and Knowledge Environment Framework (Contract no. 507173, Priority 2.3.1.7 Semantic-based Knowledge Systems; more information at `http://www.vikef.net`).

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 A Normal Day on the Semantic Web

Jane, an advanced internet user with an affinity to online social networks, heard something somewhere about the Friend-Of-A-Friend network. She discovers that there is an application which she can use to create a file that lists all her friends and that she can publish on her website. She finds this to be a fantastic idea, because like this one day she might not have to register with many different social network websites anymore, but instead "the Semantic Web" would simply *know*.

One of Jane's friends is Peter Paul, who is a researcher in Computer Science. Together with some fellows he was successful and got a paper accepted at the prestigious XYZ conference. When he submitted the paper, he entered a set of contact data and the affiliation for himself and all his co-authors into the conference management system of XYZ. (Apart from that, Peter is rather well-known in his area, and one of his students decided to write a Wikipedia page about him some time ago. This page has been imported into *Dbpedia*, a Semantic Web version of Wikipedia.)

Allen is student volunteer working at the XYZ conference. One of her tasks is to export some of the data about accepted papers and their authors into the RDF format, and to publish the resulting file on the internet. As she is not very versed in this new technology, she scripts a small custom tool that outputs RDF statements as strings into a text file and uses the database ID of authors and papers for building their identifiers.

Several people import Allen's RDF data into their own repositories, directly or indirectly. The L3S Research Center in Germany runs a version of the DBLP bibliographic database that uses Semantic Web technology, and they are mostly interested in all articles and their authors. *Ontoworld* imports the data for similar reasons, with a focus on events related to the area of ontologies. Both modify the data to fit their local conventions, also because Allen's RDF output contained some syntactic errors.

Several software agents, so-called "crawlers", make use the hyperlink structure of the (Semantic) Web to find and analyze RDF documents. The crawler of the Semantic Web search engine *Sindice* indexes RDF documents so that the search engine part can answer queries as to where a certain string has been mentioned in RDF statements.

At the end of the day, many things have been annotated in RDF, and to this end – as a necessity in RDF – these things have been given URIs, either by users, or by systems that manage the RDF data.

Take for example Peter. A whole set of URIs have been created for him[1]:

- a blank node in Jane's FOAF file
- `http://xyz2007.org/data/p_793767547`
- `http://en.wikipedia.org/wiki/Peter_Paul`
- `http://dbpedia.org/resource/Peter_Paul`
- `http://dblp.l3s.de/d2r/resource/authors/Peter_Paul`

[1] all of these examples are fictional.

- `http://ontoworld.org/wiki/Peter_Paul`

If now Jane were curious to know who of her friends is an important personality (say, somebody who has written a book, or part of a book, or an article), what would she do? Wasn't this "Semantic Web" supposed to be able to somehow gather distributed RDF data and produce acceptably intelligent answers to such a trivial question?

1.2 The Web of Documents

The WWW as we know it is what we like to call a *web of documents*[2]. Documents, usually encoded in the HTML markup language are stored on Internet hosts and are accessible for viewing in special applications: browsers. It is document-centric by design, and bases on a hypertext structure that establishes links between documents. The *hyperlink* allows users to follow pointers to documents that are located somewhere else on the Internet .

Hypertext systems had been existing for quite a while before the WWW was invented, in fact the Wikipedia Encyclopedia dates "modern" (computer-based) hypertext back to works by Douglas Engelbart and Ted Nelson [19] in the mid-late 1960s. Also these hypertext systems were used to produce webs of documents, and their commercial successors such as *Compuserve* or *AOL* managed to attract paying users to their document collections.

Apart from economic aspects[3], the main difference between the then existing hypertext systems and the WWW was its property of *globality*: so

[2]Of course, during its evolution, the WWW extended quickly with the integration of multimedia objects, which one might object to being called "documents". We accept the criticism that the WWW today is rather hypermedia than hypertext, but for the point we are trying to make we propose to neglect this distinction.

[3]Access to documents – maybe with the exception of data in university research networks – had so far to be paid for by-document, while the WWW even today is still mostly free-of-cost.

far these systems had been *local* webs of documents, based on proprietary technologies that were often hard to use and made it impossible to establish pointers between documents of different systems[4]. The evolution into the WWW was mainly possible through the creation of a uniform hyperlinking and addressing scheme: the Uniform Resource Locator (URL) as a globally unique and dereferencable identifier for documents, and the `HREF` attribute in the HTML language that allows for establishing links inside and between documents by pointing to URLs *anywhere on the WWW*.

This mechanism in reality relies on the existence of a service – the Domain Name System – which plays a crucial role in mapping symbolic hostnames in any resource identifier (URLs) into a physical location on the Internet. This is how one can be sure that, for example, a document published on a Web server will be always and unmistakably located and retrieved through the appropriate URL, and that a `HREF` link to that resource through its URL will be always resolved to the appropriate location on the Internet.

1.3 The Web of Entities

In a note from 1998, Tim Berners-Lee describes the vision of the Semantic Web (cf. [8]):

> Knowledge representation is a field which is currently seems to have the reputation of being initially interesting, but which did not seem to shake the world to the extent that some of its proponents hoped. It made sense but was of limited use on a small scale, but never made it to the large scale. This is exactly the state which the hypertext field was in before the Web [. . .]. The

[4]In fact, the commercial providers of webs of documents were not interested at all to link to information that is stored in the system of a competitor, for obvious reasons.

> Semantic Web is what we will get if we perform the same globalization process to Knowledge Representation that the Web initially did to Hypertext.

We understand this parallel as follows: like the WWW provided a global space for the seamless integration of small hypertexts (or local "webs of documents") into a global, open, decentralized and scalable publication space, so the Semantic Web should provide a global space for the seamless integration of knowledge bases (or local "semantic webs") into a global, open, decentralized and scalable knowledge space. But is this happening?

Today, as a result of many independent research projects and commercial initiatives, relatively large and important knowledge repositories have been made available, and actually are (or can be easily tranformed into) local "semantic webs", namely graphs of resources connected through properties which are defined in some schema or vocabulary. DBpedia[5], GeoNames[6], DBLP[7], MusicBrainz[8] and the network of FOAF profiles are only a few examples of available knowledge bases in semantic web formats (RDF/OWL), and furthermore already interlinked through the LinkedData initiative[9]; but any social network, any digital library metadata collection, any commercial catalog and in principle any relational database could be easily (and mostly syntactically) transformed into a local semantic web by exporting it into the appropriate format. And the suitable languages and tools for building the Semantic Web are mostly available. So why the integration of these local "semantic webs" is not *really* working well?

The argument we put forward is the following. On the one hand, the integration of local "webs of documents" into the WWW was – as previously

[5]http://www.dbpedia.org

[6]http://www.geonames.org

[7]http://dblp.l3s.de

[8]http://www.musicbrainz.org

[9]See also Sect. 3.3.2 for a discussion.

described – largely made possible by the key enabling factor of a global and unique addressing mechanism for locating and retrieving resources. On the other hand, the integration of local "semantic webs" is supposedly based on a very powerful generalization of the notion of resource identifier from information objects (e.g. HTML pages, documents, servers, etc.) to any type of resource, including concrete entities (people, geographical locations, events, artifacts, etc.) and abstract objects (concepts, relations, ontologies, etc.).

The idea can be summarized as follows: whenever a statement is made about an entity e_1, then such a statement is in principle connected with any other statement made about the same entity elsewhere and independently, *provided that the same identifier (URI) is consistently used for it*[10].

And here we get to the core of the problem: to re-use an identifier, we have to *know* it, just as in the WWW, when we want to link to a document, we have to know its URL. However, to date no scalable and open service is available to make possible and support such a consistent re-use of identifiers for entities, and this undermines the practical posibility of a seamless integration of local knowledge into a global knowledge space. And the effect is that – today – Semantic Web technology is mostly used to create local metadata or knowledge repositories, with an identity and reference problem as we have illustrated it in Sect. 1.1. There is a proliferation of identifiers for entities, which makes information integration hard, or almost impossible, and the current approach has been ex-post alignment, which is costly, and hard to achieve.

[10]In fact, this is the underlying assumption of the RDF graph merge [43].

1.4 Mission Statement

The goal of this work is to provide an a-priori solution to the identity and reference problem in the Semantic Web, and thus to enable seamless, entity-centric information integration as it has originally been envisioned. To this end, we will address the following objectives:

1. To define, design and develop a **prototype infrastructure** for enabling and supporting the systematic reuse of global entity identifiers in Semantic Web knowledge bases.

2. To design and develop **prototypical applications**, to illustrate typical use-cases and to prove the feasibility of the approach.

3. To perform **experiments and analyses**, and report their results, to prove the relevance and viability of the approach.

4. To provide **application scenarios** and analyize **research challenges** which can serve as guidelines for further work.

1.5 Overview

The rest of this document is structured as follows. Part I describes the status quo of things that were given and not part of the contributions of this work, with Chapter 2 giving background information which underlines the problem illustrated here in the introduction, and Chapter 3 describing related work.

Part II describes the novel contributions of this work. First, in Chapter 4 we present new concepts, use-cases and requirements that arise from our solution approach.

Chapter 5 describes in detail the approach: an architecture (Sect. 5.1), its implementation (Sections 5.2 and 5.3), and closes with a requirements review in Sect. 5.4.

Three applications that illustrate the usefulness and viability of our approach are described in Chapter 6: OKKAM*4P*, a plugin for the ontology editor Protégé in Sect. 6.1; FOAF-O-MATIC, a new application for the generation of FOAF profiles in Sect. 6.2; and finally, OKKAM Web Search, an application to support the generic re-use of identifiers, in Sect. 6.3.

Chapter 7 analyzes in detail three application scenarios that in our opinion can benefit considerably from the adoption of our approach. First, the area of digital library integration, as described in Sect. 7.1. Secondly, the integration of news and media (Sect. 7.2), and finally entity-centric search (Sect. 7.3).

Analysis, experiments and results that were performed based on the implementation of our prototype are reported in Chapter 8. We describe issues of performance improvement in Sect. 8.1, and an evaluation of similarity metrics that are used by our prototype in Sect. 8.2. An experiment in instance-level ontology integration is detailed in Sect. 8.3. Additionally, two analyses were performed: a cost analysis of entity-level metadata integration is given in Sect. 8.4, and the consequences of rigid designation in Semantic Web formal systems is given in Sect. 8.5.

Chapter 9 concludes with an in-depth description of research challenges that a continuation of this work will face (Sect. 9.1), an illustration of the expected impact and benefits of this work (Sect. 9.2). Finally, we describe further work that is going to be performed (Sect. 9.3) in the context of the European Integrated Projekt OKKAM which we mention in Sect. **??**.

Part I

Background and State of the Art

Chapter 2

Theoretical Background

2.1 Philosophical Background

This work deals with entities, as well their identification and representation. Many discussions of the issues that are described in this document have shown that one will almost inevitably slip into a rather philosophical argument about what "entity" and "identity" actually means, and the outcome is usually not a shared view. It is clearly beyond the scope of this work to give an account of the opinions, attitudes and schools in philosophy that deal with related topics. However, it seems appropriate to at least mention some of the important works in the area, because they motivate the decisions that were taken in the implementation phase of our approach.

Luigi Pirandello (1867-1936), Italian dramatist, novelist, and short story writer awarded with the Nobel Prize in Literature in 1934, authored "One, No one and One Hundred Thousand", a novel in which the protagonist discovers how all the persons around him have constructed in their own minds a specific view of him, and none of these views correspond to the image he has of himself [73].

We can use Pirandello's novel to illustrate an important point: the fact that several agents are describing the same object does not guarantee that these descriptions are identical. The issue is rather straightforward: imagine an everyday case about a person; the person is known to her coworkers in a certain context, they know certain things about her, most (or all) of which are related to the work. She also pursues a hobby, and is known to her friends there for certain other things. As the person tends to keep work and private life separate, the descriptions of the people that know her from these two mentioned contexts would differ considerably, to the point where they are disjoint.

Philosophers have been discussing whether there is something like a set of descriptions which are definite or authoritative for an entity, being its *essence* so to speak. Kripke [54] comes to the conclusion that this is not necessarily the case. The trouble is that descriptions are context-dependent, and they can change over time.

Now Strawson [85] describes the slogan "no entity without identity" as the fact that it is not possible to talk about something without knowing how it can be identified. But how are we to identify something if we accept the fact that descriptions are not authoritative? Strawson suggests that there might be things which can be associated to a *sort* or *kind* which have such criteria of identification, and others which do not, and that finally there is no generic answer to the problem itself. While philosophically it is certainly acceptable to come to such a conclusion, it helps us little when searching to solve a concrete problem in a pragmatic way. The consequence that is generally accepted is that there are potentially unlimited different ways of classifying things.

In this line, Lakoff [56] tells us that "knowledge is relative to understanding", which we can interpret as the problem that even if – by chance – two classifications are identical, it is not given *per se* that the *meaning* of the classifications is in fact the same. The work of Lévy [58], though only marginally related to our problem, is trying to address this issue basing on the assumption that there is a basic (and in fact very large) set of *concepts* shared by all humans, however their *arrangement* in classifications is *not* shared. Consequently, the goal of his work is to describe these concepts in his IEML language and to issue an identifier for them so that they may be used in arbitrary classifications, ensuring however that when such an identified concept is used, its meaning is shared with all other occurences of it.

A natural way of identifying things, especially when an attempt of co-ordination between agents through the comparison of descriptions has failed to provide a solution, would be to seek to *point* to the objects that are under discussion, and see whether they are identical, i.e. the same object. Pointing to something, and saying "this" or "here" is commonly known as a *demonstrative* way of identifying it. While being an optimal solution in the case of things that physically exist and are accessible at the time of the attempted identification, it has the obvious shortcoming of failing otherwise.

This shortcoming is a most substantial problem for computer-based information systems. Due to the lack of physical access to objects of the real world in such a system, the commonly practiced solution is to provide a placeholder for it, which Gangemi calls a *proxy* [38]. Usually, such a proxy takes the form of an identifier issued by the system which describes it.

This practice leads us back to the original problem described by Pirandello, but in a more serious form: while the acquaintances of Pirandello's protagonist might be able to solve an identification problem – should it arise – in a demonstrative way, software agents lack this option. As a demonstrative approach is impossible, their possibilities of co-ordination end with the comparison of identifiers.

2.2 Revenge of the ABox

From the facts presented in the introduction, it should be straightforward that the problem of unique identifiers for resources is crucial for achieving semantic interoperability and efficient knowledge integration. However, it is also evident that the largest part of research effort is made on the problem of (i) designing shared ontologies, or (ii) designing methods for aligning and integrating heterogeneous ontologies (with special focus on the T-Box part of the ontology, which defines concepts and its relations).

Perhaps because of its "practical" flavor, we must recognize that only a very limited effort has been devoted to address the issue of identity management for entities. For example, ontology editors, such as Protégé, support the "bad practice" of creating new URIs in a local namespace for any new instance created in an ontology.

In our opinion, this problem is not only of general interest for the Semantic Web enterprise, but is one of the most critical gaps in an ideal pipeline from data to semantic representation: if we do not have a reliable (and incremental) method for supporting the reuse of URIs for the new entities that are annotated in new documents (or any other data source), we risk to produce an archipelago of "semantic islands" where conceptual knowledge may (or may not) be integrated (it depends on how we choose the names of classes and properties, and on the availability of cross-ontology mappings), but ground knowledge is completely disconnected.

This is what we call the "Revenge of the ABox": the most valuable knowledge is typically the one about individuals, but research on ontology integration has traditionally concentrated on concepts and relations. The current state is that in this direction there is enough recent related work to fill a whole book [33], while (i) a viable, global approach for fostering the systematic re-use of identifiers for individuals does not exist, (ii) related work on the matching of individuals has to be gathered from many different other research fields, and (iii) the very few existing approaches for entity-centric information integration on the Semantic Web suffer from several issues which are described at a later point in this work.

The effect is that a large-scale analysis of Semantic Web data [46] has shown that, e.g. in the case of Fried-of-a-Friend profiles which still consitute a serious share of the Semantic Web data, the alignment URIs for individuals is neglectably small.

Chapter 3

State of the Art and Related Work

In this chapter we will discuss related work that Part II of this work relies upon or addresses. The topics are:

Identity management and identification. The problem of creating, managing and reusing identifiers (Sect. 3.1).

Entity-level information integration. The general problem of matching, mapping or merging data about the same entity from different sources

Identity and Reference on the Semantic Web. How information integration on the Semantic Web is currently performed (Sect. 3.3).

3.1 Identity management and Identification.

The work described in Part II is not the first approach which addresses the general problem of issuing and managing identifiers for various types of entities. To date, there are a number projects, approaches and technologies for issuing object identifiers; in fact, at a very concrete level, every operating system, every computer programming language and every database management system provides a local solution to this problem. Also, the issue of giving electronic identifiers to non-electronic objects is being covered on many levels in computer science: programming languages have memory addresses for variables that may represent something of the "real world", databases issue identifiers for records which can represent real objects. More recently, there has been a lot of interesting work on identifiers on the Web (and the Semantic Web) towards specifying the usage of URIs for referring to any type of resources (including instances on OWL ontologies). We stress that the development of the WWW - and generally the advent of more global information systems - has made evident a strong need for something like global identifiers, which can be used across different formats, in different applications, across languages and cultures. We can split into three broad categories the efforts made in this direction:

Generic identifiers and identification of electronic objects. The most prominent approach in this respect is the URI/URL mechanism for locating documents, which became popular with the WWW[1]. The URL mechanism is not without shortcomings: a URL does *not* guarantee that a specific *resource* is retrieved through it, because URLs in reality denote locations, not resources: (i) the content of a document can be changed; (ii) the whole document can be exchanged; (iii) a document can be "dynamic", i.e. it is machine-generated and changes content based on some underlying program; (iv) the document can be deleted, in which case the URL cannot be dereferenced anymore, etc. However, approaches such as PURL [78] have been conceived to deal with these issues. Other approaches for identifying electronic objects include identifier generation such as ITU UUIDs[2] and Microsoft GUIDs[3].

[1] http://www.w3.org/Addressing
[2] http://www.itu.int/ITU-T/asn1/uuid.html#what
[3] http://msdn2.microsoft.com/en-us/library/aa373931.aspx

Identification of "real-world" objects in electronic applications. The arising philosophical issue of what counts as an object will not be discussed here. Instead we list a range of approaches such as MAC addresses for network components, generic X.500 and LDAP directory services for hierarchically managed structures, EPC[4] and RFID (the "Internet of Things") as well as the Global Data Synchronisation Network[5] for generic product identification, DOI[6] and ISBN for intellectual property resources (e.g., books and publications), LSID[7] for identifying Life Science objects, and many more.

Identification of individuals (persons) in electronic applications. Especially the requirements that emerged in recent years from the E-Commerce domain have had a huge impact in terms of methods and approaches to identification of individuals for electronic transactions. The ITU recommendation X.509[8] for digital certificate and authentication frameworks has a history dating back to the early 1990s. More recent projects promoted by major players in the software industry such as Microsoft CardSpace[9], OpenID[10] and the Eclipse Foundations project Higgins[11] underline the importance of the topic.

With such a wide range of approaches, frameworks and identifier-issuing institutions, the question arises in which respects the state of the art can be advanced. It has to be noticed that all of the above-mentioned examples suffer from two types of limitations:

[4] http://www.epcglobalinc.org/
[5] http://www.gs1.org/productssolutions/gdsn/
[6] http://www.doi.org/
[7] http://lsid.sourceforge.net/
[8] http://www.itu.int/rec/T-REC-X.509/en
[9] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnlong/html/introinfocard.asp
[10] http://openid.net/
[11] http://www.eclipse.org/higgins/

- they are context-, domain- or application-specific, or they address only a certain community (even if these communities are large and important). In short, their orientation is vertical, and they are not integrated;
- they do not provide a widespread and pervasive support for the reuse of these identifiers outside the boundaries of their initial scope and application.

The work we are describing however is a global, horizontal approach, and provides a real integrating infrastructure, thus offering the possibility to overcome existing informational borders and enable real information integration right on the level of identity and identification.

In our opinion, the permanent emerging of new approaches that address the identification problem in vertical domains only strengthens our claim.

3.2 Entity-level Information Integration.

In contrast to schema-level integration, entity-level information integration deals with the actual individuals, not with integration of class structures or entity types. Entity level integration has to deal with deciding whether two entity descriptions refer to the same individual (or entity) and with merging the two entity descriptions (deciding what to include in the joint entity description).

There are several approaches trying to decide if two descriptions describe the same entity. One of the oldest variations of this problem is defined by the database community. An overview can be found in [96, 45, 31]. Here, the problem is to decide whether two relational records with different identifiers/keys provide either exactly the same information, or subsets of information describing a specific entity. The most common names for this problem in the literature are record linkage, de-duplication, data integration, and merge/purge. Suggested algorithms decide if the records describe the same entity by performing a comparison of their corresponding attributes. Others include record linkage or duplicate detection [31], reference reconciliation [29], and entity resolution [7, 39] which all refer to the problem of finding if two descriptions correspond to the same entity, for example, by exploiting associations between references [29], or by applying specific heuristics such as name matching [18].

Another related group of algorithms are the ones that aim at matching entity names by computing the distance between the string values of corresponding entity names. The algorithms included in this group suggest general-purpose methods for computing the similarity between strings [64, 25]. These algorithms are considered important since they are currently used as the basic metric on which more sophisticated approaches are based on. [26] describes and provides an experimental comparison of various string distance metrics.

Very few algorithms have been proposed in the area of metadata management. One is the TAP system [41] which uses the *Semantic Negotiation* process to identify the common description (if any) between the different resources. These common descriptions are used to create a unified view of the data. [7] identifies the different properties on which the efficiency of such algorithm depends on, and introduces different algorithms addressing the possible combinations of the found properties. Another well-know algorithm, is the *Reference Reconciliation* [29]. Here, the authors begin their computation by identifying possible associations between entities by comparing the corresponding entity descriptions. The information encoded in the found associations is propagated to the rest of the entities in order to enrich their information and improve the quality of final results. [1] is a modified version of the reference reconciliation algorithm which is focused on detecting conflict of interests in paper reviewing processes.

The most advanced approaches are the ones that do not take into account only the local similarities between corresponding entity descriptions, but also the existing inner-relationships or associations between these entities. The general idea is to model the given record information into a graph structure and then apply a data mining technique, such as classification, link analysis or clustering. For example, [9] uses a graph structure where nodes are the entity descriptions and links the relations between the entities. The algorithm uses these links to cluster the nodes and the found clusters to identify the common entities. Other used structures include *dimensional hierarchies* (chains of relations) [2], and relations between the existing records [9, 52, 51].

Other approaches deal with schema matching [76] in cases where the entities to be matched are described with a different schemata, e.g. using domain knowledge from ontologies if available [68]. The usage of ontologies also enables using query relaxation techniques as known from database research along ontologies to go beyond the "full match" paradigm. It is possible to mix data-level and schema-level matching using malleable schemas [102] to identify differently named attributes with the same semantics (e.g. "first name" / "given name").

All these approaches apply some kind of value comparison schemes that determine the similarity of the values describing the entity to be matched. Here, we can consider well-known information retrieval methods computing the similarity between text [31], or images and videos [63], even including a populated ontology to improve the matching process [42].

If the expected amount of matched entities exceeds a manageable limit, a paging mechanism is needed, which requires a metric to order the results. The problem is known in relational database systems as top-k queries. A well known approach to address this issue in a general way is using a threshold algorithm [34]. Further approaches have been developed addressing special scenarios, focusing on distributed storage of data, or using heuristics to provide probabilistic guarantees on the determined results [90]. Furthermore, ranking those entities that most closely match the target entity can be based on advanced methods for clustering such as sky-lining [4], as known from databases.

A widely used approach for matching is the combination of several different matching methods by using a weighted function [63]. This approach poses challenges in finding the right matching algorithms and in finding the best function and weights for combining them, which is handled using techniques from data mining such as Bayesian networks.

3.3 Identity and Reference on the Semantic Web.

There are currently two major approaches in the context of the Semantic Web which can be considered relevant for this work.

3.3.1 Consistent Reference Service

Jaffri et al. [49], in their work resulting from the ReSIST project, recently came to a similiar conclusion to what we had already described previously [15, 16], namely that the problem of proliferation of identifiers and the resulting coreference issues should be addressed on an infrastructural level; consequently they propose what they call a *Consistent Reference Service.* While we share this general view, their point about URIs potentially changing "meaning" depending on the context in which they are used, is philosophically disputable: the fact that several entities might be *named* in the same way ("Spain" the football team, "Spain" the geographic location) must not lead to the conclusion that they can be considered *the* *same* unter certain circumstances[12]. Furthermore, their implementation of "coreference bundles" which establish identity between entitites, are in fact very similar to a collection of `owl:sameAs` statements and share their shortcomings, as we discuss below.

3.3.2 Linking Open Data

Another notable approach is the effort of the *Linking Open Data Initiative*[13], which has the goal to "connect related data that wasn't previously linked". The main approach pursued by the initiative is to establish `owl:sameAs` statements between resources in RDF. While the community has made a huge effort to link a respectable amount of data, their approach depends on specialized, data source-dependent heuristics[14] to establish the `owl:sameAs` statements between resources, and it requires the statements to be *stored* somewhere, along with the data. These `owl:sameAs` statements, being introduced *ex-post*, have strong epistemic issues (how can one know that a certain entity is the same as another one, stored somewhere on the Semantic Web), and do in fact not address the problem of multiple identifiers for the same entity, in turn they support their proliferation.

Additionally, reasoning over `owl:sameAs` relations in distributed ontologies is a complex task: the creation of `owl:sameAs` statements among the URIs recognized to identify the same entity implies the computation of the transitive closure, potentially across the whole of the Semantic Web. The transitive closure computation is known to belong to the NL computational complexity class [71, 79]. This operation may become overwhelming from a computational point of view at the moment when the number of created URIs and related `owl:sameAs` statements reach the limits of current DL

[12]see e.g. Kripke [54]

[13]`http://esw.w3.org/topic/SweoIG/TaskForces/CommunityProjects/LinkingOpenData`

[14]`http://esw.w3.org/topic/TaskForces/CommunityProjects/LinkingOpenData/EquivalenceMining`

reasoners.

A possible counter-argument might be that this is a merely technical problem and that with time the computational power will be sufficiently improved. However, in the moment the Semantic Web will continue growing as desired, a proliferation of URIs in combination with the computational effort of a entity transitive closure over `owl:sameAs` statements will in our opinion remain extremely costly and problematic. Indeed, if the Semantic Web is going to approximate the present version of the web, it becomes hard to imagine a reliable system answering queries over massive, distributed transitive closures. This may lead to the conclusion that the Linked Data are more suitable for *browsing* than for reasoning or querying, and do thus not fully attempt to realize the vision of the Semantic Web as a large, distributed knowledge base.

Part II

Novel Contributions

Chapter 4

Concepts, Use-Cases and Requirements

In this chapter we discuss new concepts that our work introduces, as well as use cases and a requirements analysis on which the following chapters will base.

4.1 Concepts

4.1.1 Entity

At this point, we will attempt to provide a definition of the term "entity". Throughout this document, we have used the term as a synonym for and in exchange with "instance of a class", "individual", "non-digital object", "real-world object", and some more. The reason for this slight fuzziness is the fact that on the one hand we have a vision of what kind of things we would like to have stored as entities in the ENS, but on the other hand it is not predictable to which degree the *users* of the service will in the end adhere to this vision, and finally, that there are difficulties in providing a definition to begin with.

As we have hinted already in the introduction of this work, one of the (implicit) goals of the Semantic Web is to extend the realm of discourse from documents to generic objects, called *resources*. In a certain sense, it follows the notion of "individual" of Quine ([74], pages 28ff.) in that an entity is anything that can be taken as a First Order Variable. Now even though this is a straightforward position from a perspective of logical systems because it is very generic, we will have to make some considerations with regard to what needs to be identified by an ENS and what does not.

This analysis can however be performed from different points of view: electronic vs. non-electronic objects; concrete vs. abstract objects; objects which have electronic identifiers and objects which do not, and whether these electronic identifiers are suitable for re-use or not, all of which makes a definition of "entity" which is not vulnerable to a philosophical attack very hard. In the following we will try to go through the most important aspects and finally provide a definition of what criteria make up an "entity".

First of all, we can state that the WWW is already providing URLs and hyperlinks as an identification and reference mechanism for web documents[1]. This means that identifiers for such documents can already by found and re-used without further need for a supporting infrastructure. The existance of the WWW with its unnumerable amount of hyperlinks trivially proves this point. In the same line, we should mention other types of electronic objects for which vertical solutions to identity and reference exist. These have been described in Sect. 3.1, and our point is that while in the long term a unification of approaches between these systems and the ENS might be desirable (and beneficial), at this point the *necessity* for locating and re-use their identifiers is not strictly given in all cases.

Next, as we are dealing with the Semantic Web and its technologies, we should consider classes defined in ontologies. The case for these objects is similar to the one for WWW documents, as they are identified by URIs

[1]Shortcomings of the URL mechanism as mentioned in Sect. 3.1 shall be neglected at this point because solutions already exist.

which base on the URI of the ontology that defines them, and should thus in theory also be locateable and re-usable in the same way as described above. It is however in reality slightly more complicated due to technical reasons: (i) the use of a class name (i.e. its URI) to specify the type of an entity that is described in an RDF graph does not imply or require that the *definition* of this class be accessible, which may make it impossible to evaluate the meaning of the class. In this case the URI of the class is simply a name without furter meaning; (ii) on the one hand finding URIs for classes is not as well-supported as finding URLs for documents on the WWW, as similar search engine technology is only in prototype status; on the other hand, class re-use itself is not as trivial as linking to a document, as it implies consequences on the inference level, a point which is proven by the vast amount of work in the area of ontology matching [33]. Therefore, in our point of view, classes in ontologies do not necessarily represent first-class citizens in an ENS, as the re-use of their identifiers is expected to be low.

As so far we have considered electronic objects, we should now analyze non-electronic objects (NEOs). Here, again, we restrict our analysis to the necessity of supporting the re-use of identifiers. Similar to electronic objects, vertical approaches for identification of some types of non-electronic objects exist (see Sect. 3.1). However, the re-use of these identifiers often is hindered by difficulties of finding them – a shortcoming from which many of the vertical approaches suffer, whether they identify electronic or non-electronic objects. Another distinction within the group of NEOs is whether they are concrete or abstract. Abstract NEOs, may often be ontological classes, and are as such hard to be precisely pinned to an identifier that is suitable for re-use (see Chapter 2 for a discussion).

However, let us for example take *events*: they are usually not considered classes/concepts, but are abstract in the sense that they are not (and were never) physically present. In the special case of events we run into more problems, because the specification and identification of temporal phenomena is generally complicated: should it be possible to have an identifier for every millisecond of the past because it might have been the point in time when something happened? As a pragmatic solution for points in time, we propose to take it as identifiable and referenceable by definition, simply through the use of a suitable data-type property.

Concrete NEOs on the other hand include things that are or have been physically present, with all the difficulties of this definition. Should a grain of sand that is part of a brick which has been used to build a house be an entity in our sense? Probably not. Is a bolt that is part of a beam that has been used to build the wing of an airplane an entity in our sense? Maybe: if it needs to be described and referenced in one or more information systems (e.g. the Quality Assurance systems of the involved parties), then yes.

It is evident that a more detailed analysis of such questions quickly runs into philosophical issues in many kinds of ways, which makes a clean definition of our notion of entity very hard. Focussing on our main issues – the possibility of finding an identifier and its suitability for re-use – one may come to a definition similar to the following:

Definition 4.1 (Entity) *An Entity is any thing, abstract or concrete, electronic or non-electronic, that (1) needs to be referenced in an information system by means of an identifier, (2) does not have an electronic identifier, or (2a) no electronic identifier that can easily be located, or (2b) no electronic identifier that is suitable for re-use in other information systems, and (3) for which an Entity Description can be provided which is enough specific to distinguish it from all other Entities.*

4.1.2 Entity Description

To identify something, it is necessary to distinguish it from other things, which leads to the question how an entity is supposed to be described in a way that sufficiently distinguishes it from all other entities. A straightforward approach would be to classify the entity, and for each class of entities provide a "standard" descriptive schema that has to be instantiated.

However, as a conclusion we draw from the fact that things can be classified in many, maybe unlimited, different ways (see Ch. 2), we decided to drop the idea of classifying entities deliberately. Instead of attempting to provide something which might end up as either the unmanageable set of all conceiveable classifications of things, or an "average" that could be altogether unsatisfying, we describe entities in a semi-structured way:

Definition 4.2 (Entity Description) *An Entity Description is a non-empty set* $\Delta = < n, v >$ *of name/value pairs, with the additional property that* $n = \emptyset$ *is permitted.*

This allows a description to contain pairs such as the URI of a datatype property defined in an ontology elsewhere on the Semantic Web as a name, and a corresponding value; an empty name and the prose description of the entity in a natural language; or even a set of co-ordinates that identify an entity in space. It is obvious that the shift away from a strongly schematized information system towards such a rather free-form structure is posing different (and maybe greater) challenges in terms of query answering, but our standpoint is that any attempt to provide a "one-size-fits-all" classification of things must lead to the failure of our vision because of a lack of adoption.

4.1.3 Entity Profile

On top of the Entity Description, we have created an extended data structure that represents *all that is known* to the ENS about an entity. This is at a minimum its Entity Description and its identifier, but may contain further elements as defined below:

Definition 4.3 (Entity Profile) *An Entity Profile is a tuple*
$E = \langle i, \Delta, R, ID, A, i_{pref}, s \rangle$
where i *is the identifier issued by* OKKAM *for the entity,* Δ *its Entity Description ,* $R = \{\langle type, value \rangle\}$ *a set of typed references to external information sources that are known to refer to the entity,* $ID = \{\langle i, i' \rangle\}$ *a set of assertions of identity between the entity and other entities that are known to be identical,* $A = \{s\}$ *a set of alternative identifiers of the same entity in different other information systems,* i_{pref} *the preferred identifier for the entity which is either* i *or one* $x \in A$*, and finally* s *the Wordnet Synset identifier that helps to describe the high level type that entity is known (or supposed) to have.* $R = ID = A = \emptyset$, $i_{pref} = \emptyset$ *and* $s = \emptyset$ *is permitted.*

4.1.4 Entity Name Service (ENS)

As motivated in the previous chapters, the main objective of this work is to establish a running infrastructure which enables global, pervasive and re-usable identification of entities in the Semantic Web. This infrastructure provides a service that has a certain parallel wich what we know in the Internet as the Domain Name Service (DNS): it provides a lookup mechanism, only that it treats generic entities instead of internet hosts.

The overall vision – as depicted in Fig. 4.1 – is to go beyond the boundaries of RDF graph integration, towards an infrastructure that integrates information across systems and formats.

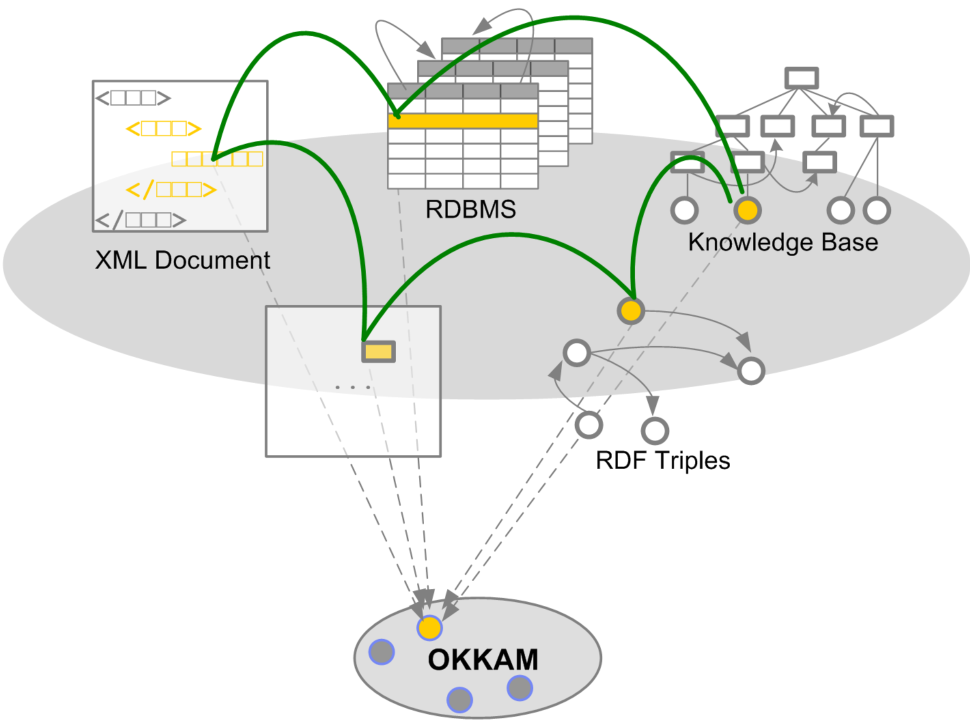

Figure 4.1: OKKAM as an ENS for multiple data formats and applications

To this end, we introduce the notion of an Entity Name Service (ENS) from a functional perspective:

Definition 4.4 (ENS) *An Entity Name Service is a service that (1) contains a set of Entity Profiles EP, that (2) given an Entity Description Δ, returns its globally unique and rigid*[2] *identifier i in a suitable form, or a set of candidate identifiers I if no one-to-one mapping could be found, and (3) given an identifier i that was issued by the ENS, returns the description Δ of the Entity which the identifier denotes, and (3) does so in a deterministic, stable and monotonic way, i.e. the relation between an identifier and an entity may not change once it is established and an identifier may not disappear. Consequently, the ENS is characterized by EP and the following functions:*

[2]The property of *rigidity* of an identifier in logics is commonly understood as denoting *the same* object, whenever or wherever the identifier occurs.

$$i = f_1(\Delta) \tag{4.1}$$

$$I = f_2(\Delta) \tag{4.2}$$

$$\Delta = f(i) \tag{4.3}$$

Note that we are not giving a formal specification of the *components* of an ENS for the reason that the internal structure of an ENS is an implementational issue and thus irrelevant for its intended use.

To introduce a name that is often used throughout this work, we also define OKKAM:

Definition 4.5 (Okkam) OKKAM *is an existing implementation of an Entity Name Service, as described in Ch. 5.*

4.2 Use Cases

Three main use cases have been identified as integral to the process of interacting with an ENS: (i) searching for an entity in the ENS, (ii) publishing a new entity in the ENS, (iii) using identifiers from the ENS in local data sources. These use cases are further described in the following.

4.2.1 Entity Search

At the base of most interaction with the ENS, there is the process of searching for an entity, as illustrated in Fig. 4.2.

A human agent interfaces with the ENS through a client application, and provides either a structured description of the desired entity, or a set of search terms (names, words in natural language), which are transferred to the entity search interface of the ENS. The ENS performs a search in its data store, selects candidates, ranks them and returns the results. The client application can then display – or otherwise use – these results.

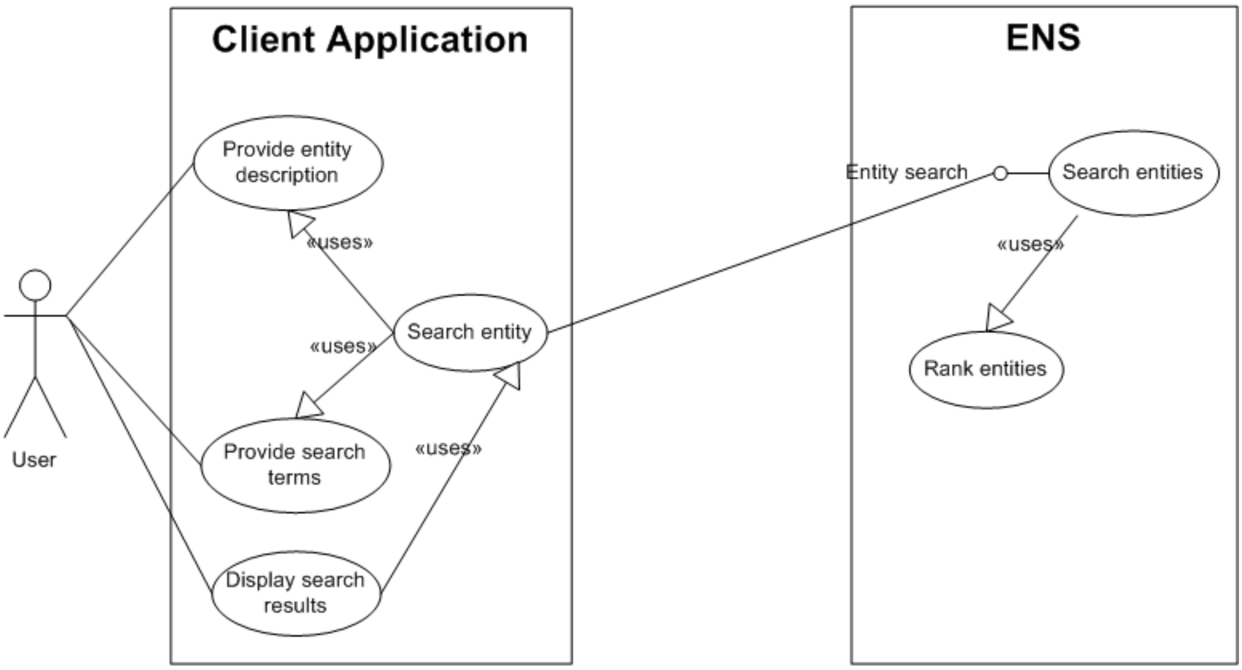

Figure 4.2: The Entity Search Use Case

4.2.2 Entity Publication

A second goal of the ENS is not only to provide identifier for entities that are already stored in the ENS, but also to issue identifiers for new entities. To this end, a use-case of entity publication as in Fig. 4.3 is required.

4.2.3 Okkamization

Typically, when creating structured content or annotating unstructured content, the entities described are issued with a local identifier (e.g. a primary key in a relational database, or a URI relative to the used ontology in the case of OWL). In order to lift this content to a globally integratable level, we have to replace this process of local identification with another one that builds on globally *shared* identifiers. We call this process *okkamization*.

Definition 4.6 (Okkamization of an Entity) *We call* okkamization *with*

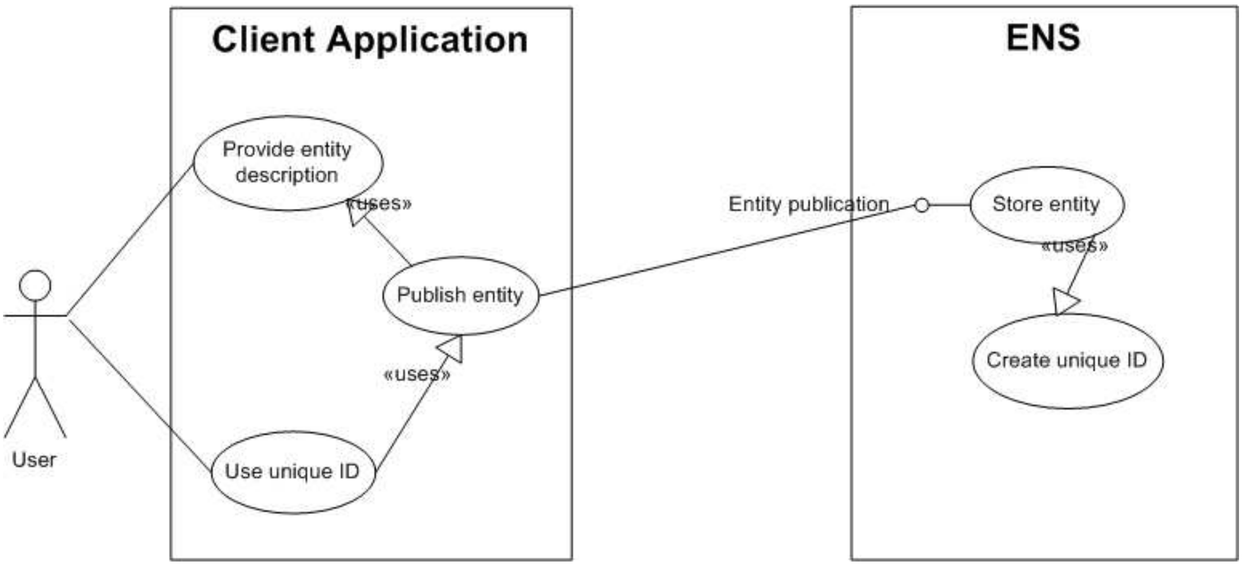

Figure 4.3: The Entity Publication Use Case

respect to a single entity the process of assigning an OKKAM *identifier to an entity which is being annotated in any kind of content, such as an OWL/RDF ontology, an XML file, or a database, in order to make the entity globally identifiable.*

Definition 4.7 (Okkamization of a Data Source) *We call* okkamization *with respect to a data source the process of assigning an* OKKAM *identifier to all relevant entities in the data source. Relevance is defined by the agent who requests okkamization of the data source, through a set of conditions suitable to select the desired set of entites from the data source.*

This okkamization would usually be achieved through functionality provided by a client application which accesses an ENS through its API, and presents (if available) a list of top candidates which match the description for the entity provided within the client application. If the entity is among these candidates, the client agent (human or software) uses the associated OKKAM identifier in the respective information object(s). If the entity cannot be found, the client application can create a new entry for this entity in Okkam and thus cause an identifier for the entity to be issued and used as described before. Due to its procedural nature, we illustrate it in the form of a sequence diagram in Fig. 4.4.

Alternatively, for simple tasks with only a small number of entities, we provide the application OkkamWebSearch[3], which features a simple web-based user interface that allows users to search for entities in order to insert the found identifiers manually in their data sources.

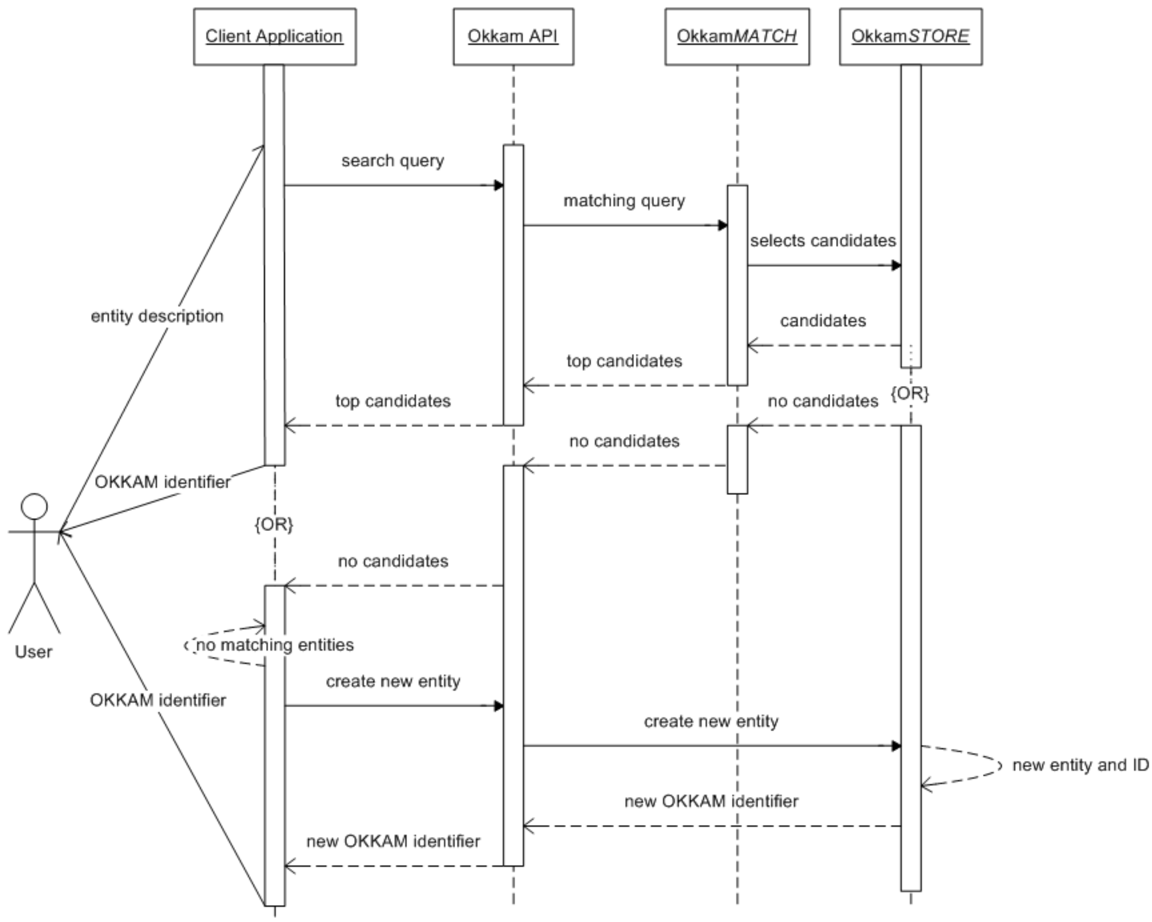

Figure 4.4: Sequence diagram of an Okkam standard use case

[3]http://www.okkam.org/experimentalokkamsearch

4.3 Requirements

In the following we illustrate the most important requirements that arise when attempting to provide a solution to the problem described before. We groug these requirements into the following seven categories:

Functional Requirements. The list of functions (with I/O and/or side-effects) that need to be provided by a system to become usable or useful.

Interface Requirements. The accessibility of a system from other systems or agents is defined through its interface. The types of systems or agents that are expected to use a system define the type of interface(s) that they require.

Operational Requirements. The set of conditions that the system is expected to fulfill regarding its behavior during operation and lifetime.

Performance Requirements. The demands that are made on a system with regards to the resources that the system uses to perform a certain task.

Accuracy Requirements. These requirements relate to the degree of accuracy at which a system fulfills specific measurable parameters, usually bound to certain thresholds.

Socio-economical Requirements. Conditions that a system has to fulfill in order to be acceptable/successful in a society or economy.

For each of the listed categories, we describe the corresponding requirements. These requirements will later be referenced in Section 5.4 to analyze the appropriateness of the proposed approach.

Functional Requirements

R-1 **Search**. One of the basic requirements of a service that issues global identifiers is the search functionality, to retrieve lists of candidates which enable the agent (human or artificial) to decide whether the entity under consideration is already defined or has to be newly created.

R-2 **Creation**. The creation of a new entity, including a description that allows it to be discriminated against all other entities, is the next requirement in the processing chain.

R-3 **Modification**. Finally, the modification (or extension) of the entity and its description has to be considered, and appropriate functionality has to be provided. Note that under certain circumstances it might not be desirable to modify all existing data about an entity, and it should never be allowed to change and entity's identifier.

R-4 **Resolution**. The service must provide functionality to establish identity between two entities ex-post, as a consequence of detecting by analysis that two entries actually describe the same entity.

Interface Requirements

R-5 **Web-based interface**. The functions of the system described in Section 4.3 must be exposed through a web-based interface, to allow for a generic and application-independent access to the system.

R-6 **Programmatic interface**. The functions of the system described in Section 4.3 must be accessible through a programmatic interface (API), to allow application developers the use of the service within software components.

R-7 **Batch data interface**. It is desirable to have an interface that accepts bulk data in a defined format, for fully automatic processing (i.e. enrichment of the data with entity identifiers).

Operational Requirements

R-8 **Distribution**. The system must be distributed to be fully in line with the technical philosophy of the WWW, and to avoid presenting a single point of failure for other systems that make use of or depend on it.

R-9 **Availability**. The system must be highly available to allow an agent (human or artificial) access to its functionality at any time. When the system will be accessed is not predictable due to its global character, so a 24x7 availability should be approximated.

R-10 **Persistence**. Data in the system cannot be volatile for obvious reasons, so an appropriate persistence mechanism must be provided. This mechanism is in close relation with Req. R-1, as the chosen approach must allow for suitable query mechanisms.

R-11 **Data security**. The security of the data in the system must be ensured, especially the deletion of data and the modification of entity identifiers underly special restrictions.

Performance Requirements

R-12 **Execution time**. The desired execution time must be considered for each functional requirement in Section 4.3 and related to the respective interface that is used (Section 4.3), where appropriate. We identify at least the following desired execution times based on user experience considerations and requirements from automatic processing:

a) manual creation: 2sec; b) manual modification: 2sec; c) manual search: performance similar to current search engines, longer execution times for more detailed searches are acceptable; d) programmatic or batch search: as in c); e) programmatic or batch creation/modification: below 1sec.

R-13 **Response time**. The responsiveness of a system is not necessarily identified by its execution time for individual tasks. Multi-tasking environments my have longer execution time for certain tasks, but still stay responsive (i.e. able to accept a request). The envisioned system should be able to respond to several thousand requests per minute.

R-14 **Resource usage**. The well-known trade-off between memory usage and processing speed has to be balanced with respect to the above requirements. No specific requirements apart from realistic values will be made.

Accuracy Requirements

R-15 **Precision**. When searching the system, the precision metric[4] from the area of Information Retrieval (IR) must be kept as high as possible, specifically the *called* precision in the 10 top-most results presented to a human agent and the 100 top-most results presented to an artificial agent, should approach 100%.

R-16 **Recall**. The Recall metric[5] is to be considered secondary to Precision. It is not desirable to "lose" entities that are relevant for a query, but due to their inverse relation, an increase in Precision causes a decrease in Recall, which will be accepted. It must also be noted

[4] http://en.wikipedia.org/wiki/Information_retrieval#Precision
[5] http://en.wikipedia.org/wiki/Information_retrieval#Recall

that insufficiently specific queries will produce a very large number of results, in which case recall will become completely irrelevant as the number of search results stops being manageable.

Socio-economical Requirements

R-17 **Privacy protection**. This may refer to the privacy of the user of the system, and the privacy of entities that are described in the repository. In the first case, appropriate standard procedures have to be taken that are commonly used in WWW information systems. In the second case, the minimal requirement is that no information that was not public in the first place will be published through an automated process integral to the system (i.e. information harvesting).

R-18 **Legality**. The operation of the system obviously has to be fully within legal bounds. This requires in-depth assessment of the relevant legal guidelines, with special focus on information clustering.

R-19 **Trustworthiness**. To achieve a widespread use of the system, trust in the system, its operation and ownership has to be established. This is a very broad requirement and affects operations outside the system level.

R-20 **Cost of use**. To achieve a widespread use of the system, the cost of use of the system must be kept free-of-cost. This applies especially to the search functionalities, as they are vital to the reuse of the global identifiers.

R-21 **Ease of use and Accessibility**. The use of interfaces and functions of the system must be kept as simple as possible, with respect to the target audience. Especially web-based access to the system must be kept absolutely self-explanatory to the standard WWW user. Additionally, relevant accessibility guidelines such as U.S. Section 508 [91], the W3C's WCAG [93] or the European Euracert [32] have to be implemented.

Chapter 5

Okkam – an Entity Name Service

In this chapter we describe a general ENS architecture that has been conceived. Furthermore we introduce OKKAM, a reference implementation of this architecture, and its core component, a matching and ranking architecture for entities. This infrastructure is the base on which the tools and applications described in Chapter 6 and the experiments in Chapter 8 are building. The chapter will close with a review of the requirements presented in 4.3, analyzing to which extent these requirements have been fulfilled by the current implementation.

5.1 The Okkam Architecture

At the heart of an ENS infrastructure there is the central repository for entity identifiers, named OKKAM. This repository can be imagined like a very large phonebook, where semi-structured descriptions of entities are stored and associated to globally unique identifiers for these entities. It furthermore provides the functionality to add entities and their descriptions to the repository that have not existed there so far, and to retrieve their OKKAM identifiers for use in information systems.

The global service OKKAM, which provides for the Entity Repository and a service infrastructure so that tools and applications can make use of this new technology, is architecturally speaking, an instance of what we call Okkam*NODE*, as illustrated in Fig. 5.1. The goal is to provide a distributed infrastructure, to comply with Requirement R-8 described in Sect. 4.3. Distribution can occur on different levels:

1. As every Okkam*NODE* potentially has to deal with a very large amount of data, it may rely on a distributed storage system, as depicted in Fig. 5.1.
2. To fully comply with Requirement R-8, a federation of synchronized Okkam*NODE*s can provide the ENS service in a deterministic or transparent fashion. Furthermore, to comply with Requirement R-17 (Privacy Protection) in corporate environments, a private or corporate Okkam*NODE* is possible that manages entity identifiers which are only relevant and visible *inside* the respective environment.

A single Okkam*NODE* consists of the following components:

Okkam*STORE*. On this layer, many crucial problems of the architecture have to be addressed, as it deals with the mapping from logical to physical representation of entities. On the logical level, this layer provides two repositories: (i) the Entity Repository, which represents the core data necessary to identify and disambiguate each entity, and (ii) the Reference Repository, which stores pointers from the OKKAM entity into other (external) public information systems, such as the WWW, knowledge bases or databases. This reference repository can be used for matching with background knowledge (see Sect. 5.3), or as a starting point for crawlers of (semantic) search engines. On the physical level, this layer takes care of the management of the potentially massive amount of data that has to be stored (if necessary in a distributed or peer-to-peer fashion), and the respective lower-level query mechanisms.

Okkam*MATCH*. On top of the storage layer, we position the matching layer, which provides for higher-level query and matching functionality in the system. One of the main tasks of an OKKAM*NODE* is to match input data from an application against all the entities in the repository and to produce a ranked list of candidate which fulfill the criteria for further processing by the application. The respective methods are situated on this level.

Okkam*ACCESS*. To enable a secure and controlled access, all requests to an OKKAM*NODE* have to pass an access level before queries can be issued. We envision a second level of access control besides the query and matching level which takes care of the issue which data from the storage level may be analyzed to answer a matching or query request. This aspect will not be addressed in this work, but represents a main research challenge for the future, as described in detail in Sect. 9.1.2.

Okkam Empowered Tools. As part of the OKKAM application services, an outcome of the project will be a set of user-friendly horizontal applications that enable the community of users to create okkamized content. Examples include plug-ins for the widely used ontology editors, as well as word-processors or HTML editors that have the ability to annotate entities in their documents with identifiers from OKKAM.

Okkam Services. Every OKKAM*NODE* shall provide a set of largely application and domain-independent services that ease the okkamization of larger datasets, the manual entry of entities into OKKAM by a user, or the bulk import of entities through special input files.

Okkam*DEV*. Covering the OKKAM*CORE* layers as well as the services layer we place the OKKAM developers support - such as APIs and adequate documentation - to enable programmatic access to the respective functionalities. Even though in Fig. 5.1 this appears as a monolithic part in the architecture, there are of course separations with respect to access rights: parts of the development support, e.g. the parts that address OKKAM*CORE*, will remain private to the respective managers of the system.

The current version of OKKAM is an implementation of parts of this architecture, namely a non-distributed version of OKKAM*STORE*, an experimental version of OKKAM*MATCH*, the OKKAM-empowered tools OKKAM*4P* and FOAF-O-MATIC, the application OKKAMWebSearch, as well as a subset set of the developer API and toolkit OKKAM*DEV*. These components will be described in more detail in the following.

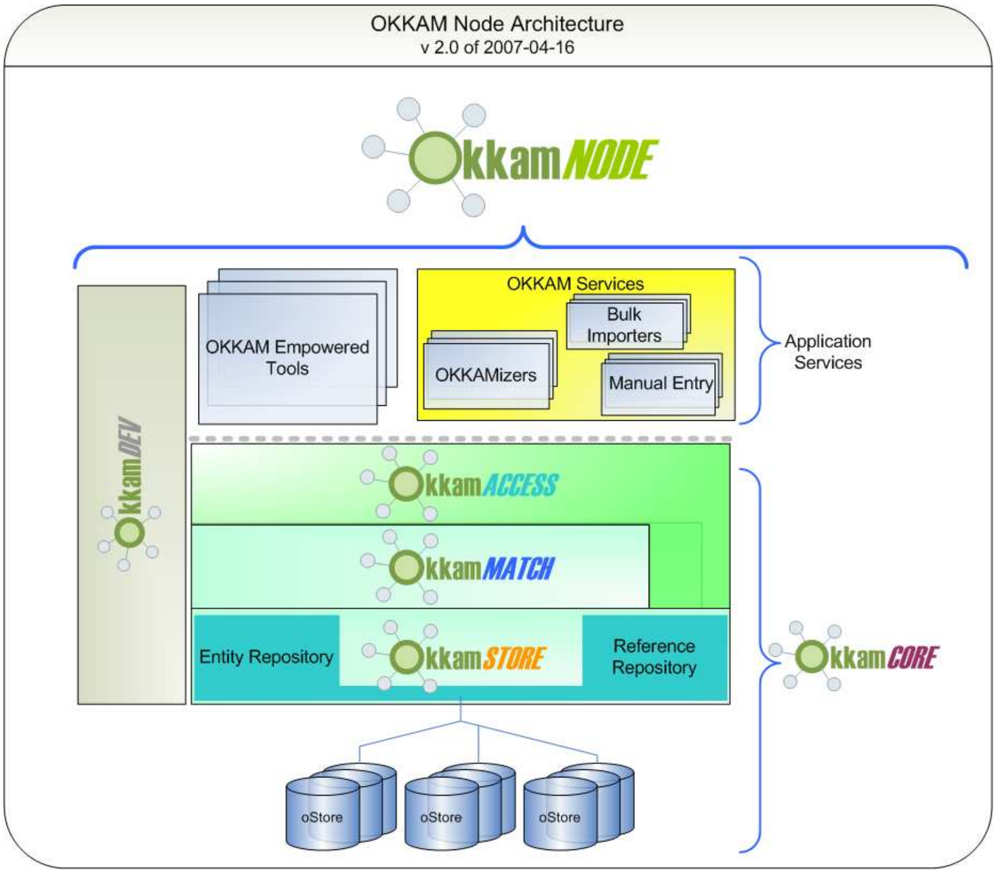

Figure 5.1: Architecture of an OKKAM*NODE* and its application layers

5.2 Prototype Implementation

The OKKAM ENS service is comprised of several main components which are depicted in Fig. 5.2.

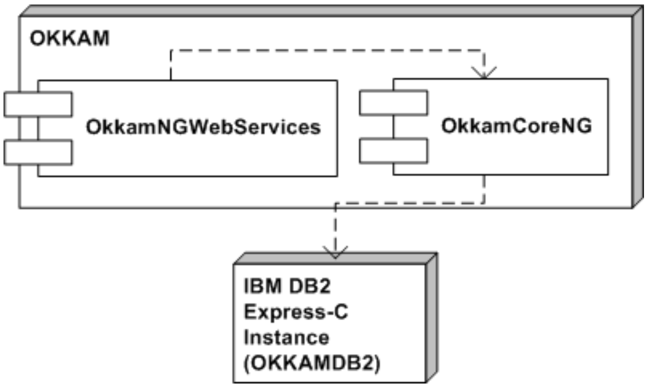

Figure 5.2: Top-level components of the OKKAM implementation

`OkkamNGWebServices`[1] is the component that exposes the OKKAM services to the outside world, on a programmatic level; `OkkamCoreNG` implements the core functionality; the data themselves are stored in an instance of a relational database[2].

The decision to use a relational persistence back-end is partly motivated in Sect. 5.2.2, but the most important reason is that an expected "population" of the repository (also for the prototype) is going to be in the millions, and that alternative representations that would maybe favour more the process of entity matching, such as Bayesian Networks (see Sect. 3.2) or logical models (e.g. OWL/RDF) were not reported to be functional – in terms of storage and especially *query* capabilities – for such an amount of data when the architecture was designed[3].

[1]The acronym "NG" that appears in several of the component names reflects the fact that the implementation described here is the "Next Generation" of our first prototype.

[2]The freely available database IBM DB2 Express-C, V9.1.

[3]At the time of this writing, the situation has slightly changed with regards to RDF stores, which report to be able to store billions of triples (see e.g. `http://esw.w3.org/topic/LargeTripleStores`). However, these reports do not describe the behaviour of the systems with regard to non-trivial queries

To give an idea of the size of the prototype implemenation in its current state, Table 5.1 reports about several measures of the source code of its main components: it consists of roughly 90 classes, 400 functions and 3500 non-comment source statements (NCSS)[4].

Component	Classes	Functions	NCSS	Javadocs
OkkamWebServices	3	6	58	9
Java Client Library	23	122	875	35
OkkamCoreNG	65	265	2627	230
Total	91	393	3560	274

Table 5.1: Code size of OKKAM's main components

In the following we will describe the implementation in more detail, first from an external view in terms of interfaces and data structures, and then from an internal view, describing processes and algorithms.

5.2.1 A Black-box View on Okkam

The functionality of OKKAM is most easily explained by describing the possibilities of interaction with its services. To this end, there are three aspects that are of interest: (i) data structures that serve as input or output, (ii) API interfaces, and (iii) the client library that has been developed to provide a low-boundary access to the features of OKKAM.

Data Structures

The AnnotatedQuery Data Structure For posing a query to the OKKAM ENS on a programmatic level, we have developed a suitable data structure, as depicted in Figures 5.3 and 5.4. The structure holds the query

(e.g. substring search in datatype properties) or even inferencing, which makes it very hard to judge their usefulness.

[4]Not counted are several hundred lines of XML Schema and SQL data definition.

itself, as well as additional information that may be used by the matching component of OKKAM to retrieve candidate entities.

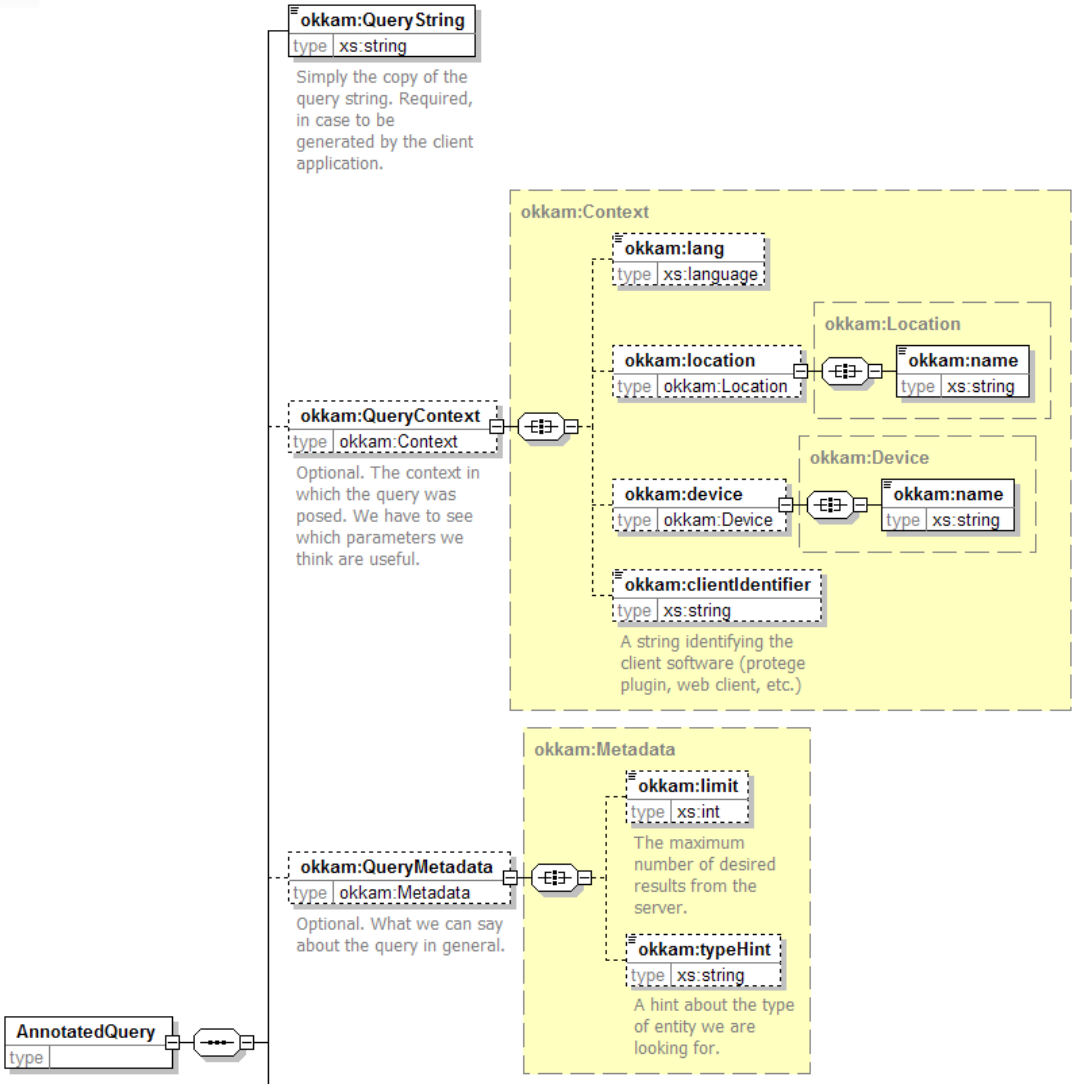

Figure 5.3: The AnnotatedQuery Schema (Part 1)

The query itself is represented in two ways:

1. in the form of a query string (element `okkam:QueryString` in Fig. 5.4) as provided by applications such as OKKAM Web Search (see Sect. 6.3). This representation is important because it may allow a matching component to perform further analysis of the complete query that cannot

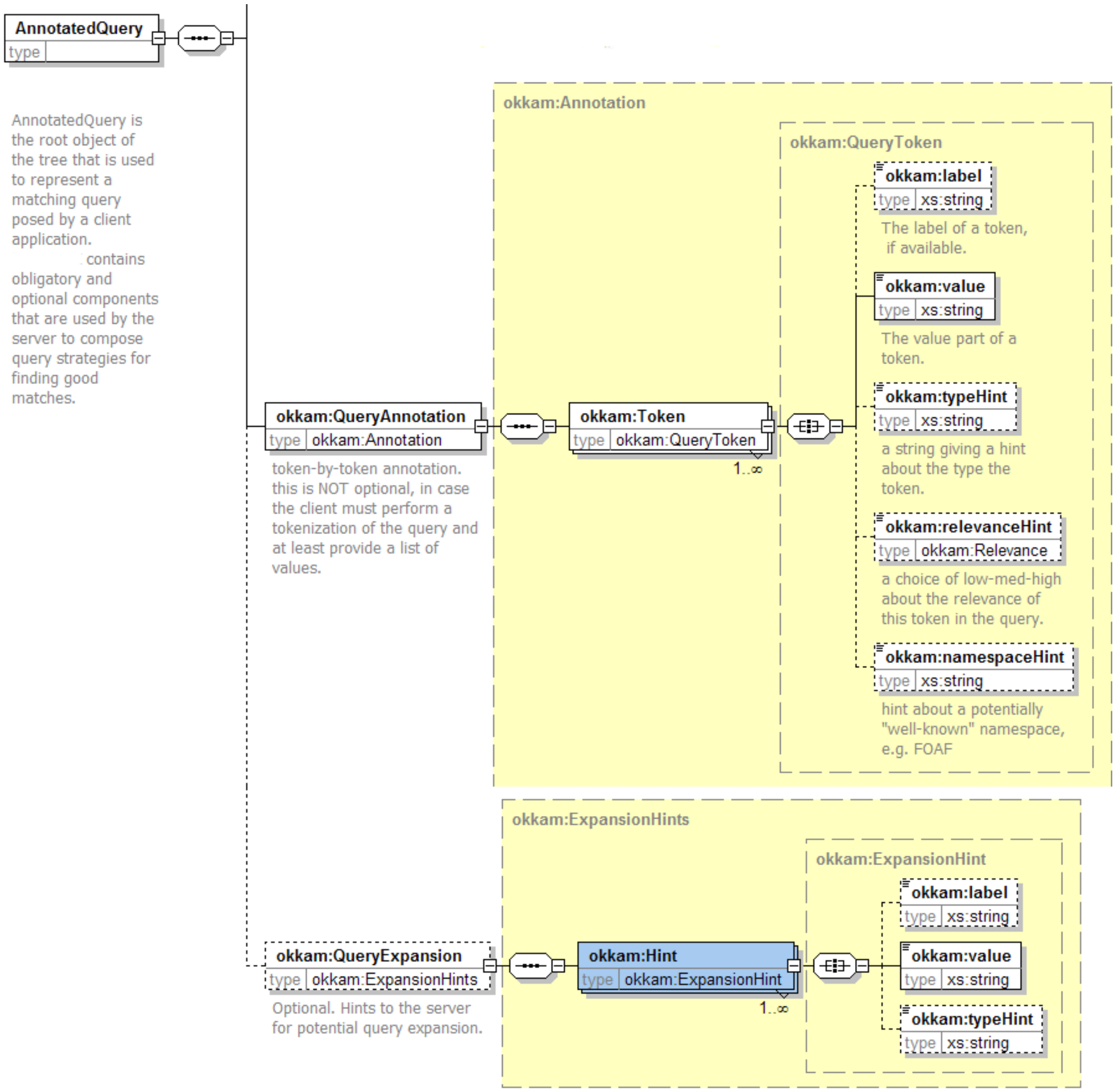

Figure 5.4: The AnnotatedQuery Schema (Part 2)

be foreseen at this point.

2. in the form of individual, annotated tokens (element `okkam:QueryAnnotation` in Fig. 5.4), which facilitates the input by client applications that can provide more structured input, such as the ones described in Sections 6.1 and 6.2.

The tokenized representation, apart from holding the name/value pair that represents the query token, may contain a hint about the ontological type of the token in the form of a URI that points to a class definition (`okkam:typeHint`), a hint about the relevance of this token with respect to the complete query (`okkam:relevanceHint`), and a hint about the namespace that the name in the name/value pair originates from (`okkam:namespaceHint`).

The query may also hold several query expansions that a client can propose to be used in the query evaluation (`okkam:typeHint` in Fig. 5.4). They may consist of a name/value pair, as well as a type hint as described above. These expansions can originate from background knowledge that a client has, which are however not part of the original query (posed by a user, for example).

Furthermore, the AnnotatedQuery data structure may contain query metadata (`okkam:QueryMetadata` in Fig. 5.3): first, a desired limit for the number returned candidates that is important especially for clients that have limited display capabilities; second, a type hint as to which kind of entity the agent is looking for (as described above).

Finally, it is possible to provide OKKAM with a representation of the user context under which the query was created (`okkam:QueryContext` in Fig. 5.3). From our experiences in contextual knowledge representation [14, 82, 83, 84] we can report that a definition of "context" is a moving target that in many cases leads to a philosophical discussion, which makes it next to impossible to find a generic representation that suits every notion of context. For the current problem we have tried to pursue a pragmatic approach, and provide a manageable representation of the parameters that we think might influence the production of a query result that is suitable under a certain situation. Our representation of context may thus include: (i) a specification of the human language of the user environment; (ii) the location at which the query was posed, represented as a string[5]; (iii) the device that is being used in the user context, represented as a string[6]; (iv) an identifier for the software client that is being used, e.g. to recognize which client application or library has created the query.

It is clear that many of these additional query components are not available or known under every circumstance. But to guarantee a sufficient interface stability, we decided to try to foresee most of the relevant metadata that might lead to higher-quality query processing on the server side. A minimal example of an AnnotatedQuery may for this reason simply consist of a query string and its tokens, and name/value pairs with empty names, and in Listing 5.1:

Listing 5.1: Example AnnotatedQuery XML Representation

```
<?xml version="1.0" encoding="UTF-8"?>
<okkam:AnnotatedQuery
  xmlns:okkam="http://www.okkam.org/schemas/AnnotatedQuery"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.okkam.org/schemas/AnnotatedQuery␣
     http://www.okkam.org/schemas/AnnotatedQuery.xsd␣
     AnnotatedQuery.xsd">
  <okkam:QueryString>John Doe</okkam:QueryString>
  <okkam:QueryAnnotation>
    <okkam:Token>
      <okkam:value>John</okkam:value>
```

[5]This representation has obvious shortcomings and is not sufficient for direct use in a geographical information system. However, it can hold the OKKAM identifier for a location, which we hope can alleviate this problem.

[6]With similar shortcomings and proposed solution as before.

```
      </okkam:Token>
      <okkam:Token>
        <okkam:value>Doe</okkam:value>
      </okkam:Token>
    </okkam:QueryAnnotation>
</okkam:AnnotatedQuery>
```

The OkkamURIResult Data Structure After submitting an AnnotatedQuery to the OKKAM ENS, the service returns an instance of the OkkamURIResult data structure, the schema of which is illustrated in Fig. 5.5.

The data consist of four components (the element names given below correspond to Fig. 5.5):

`Result`: A list of top-k URIs, ranked in descending order, that were found in OKKAM as candidate matches for the query.

`Confidences`: A list of confidences, that describe how good the match is in comparison with the input query (see Sect. 5.3 for details). This list is sorted in one-to-one correspondence to the list of candidate URIs.

`Code`: An integer code that describes the contents of the result structure (whether an actual result is contained or an error occured.

`Message`: A text string for display to the user that describes the content of the structure, relative to the above code.

The EntityProfile Data Structure The schema for the description of a single entity (see Sect. 4.1.3) is depicted in Fig. 5.6.

It is both used for retrieving the description of an existing entity from OKKAM, as well as providing the description of a new entity to OKKAM. The EntityProfile consists of seven main elements (see Fig. 5.6):

`Labels`: A list of name/value pairs of type String.

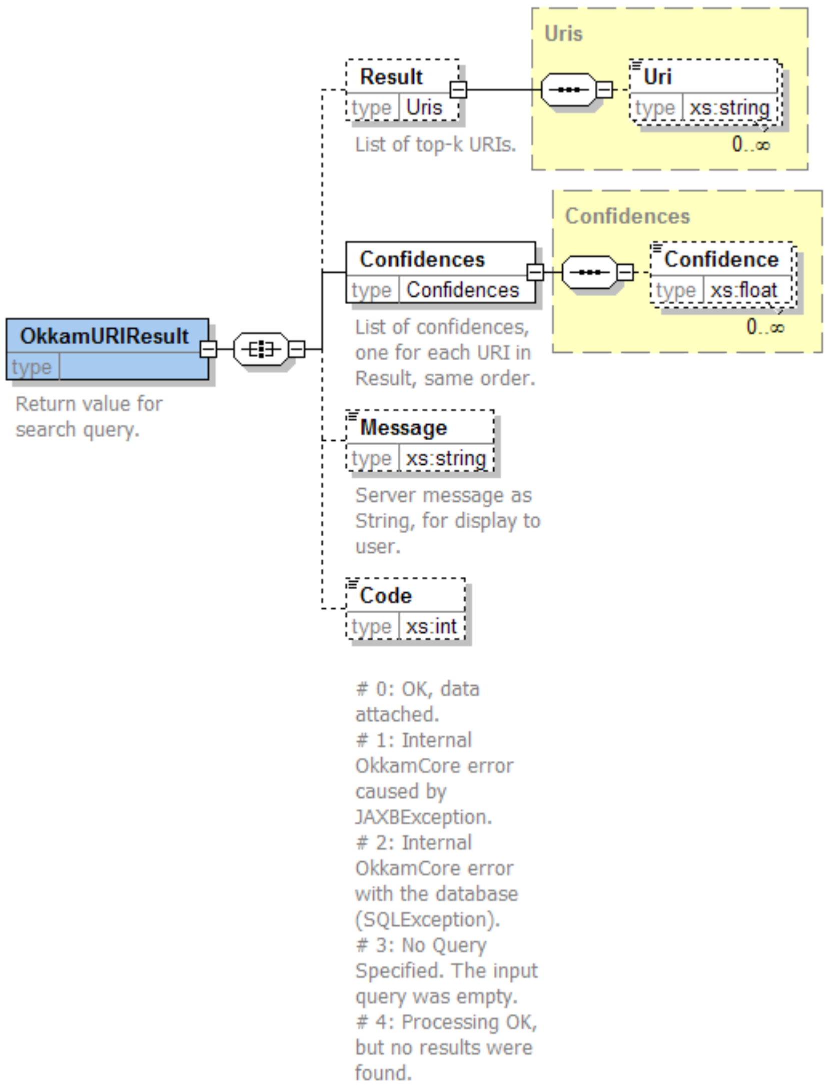

Figure 5.5: Schema of return value for search queries.

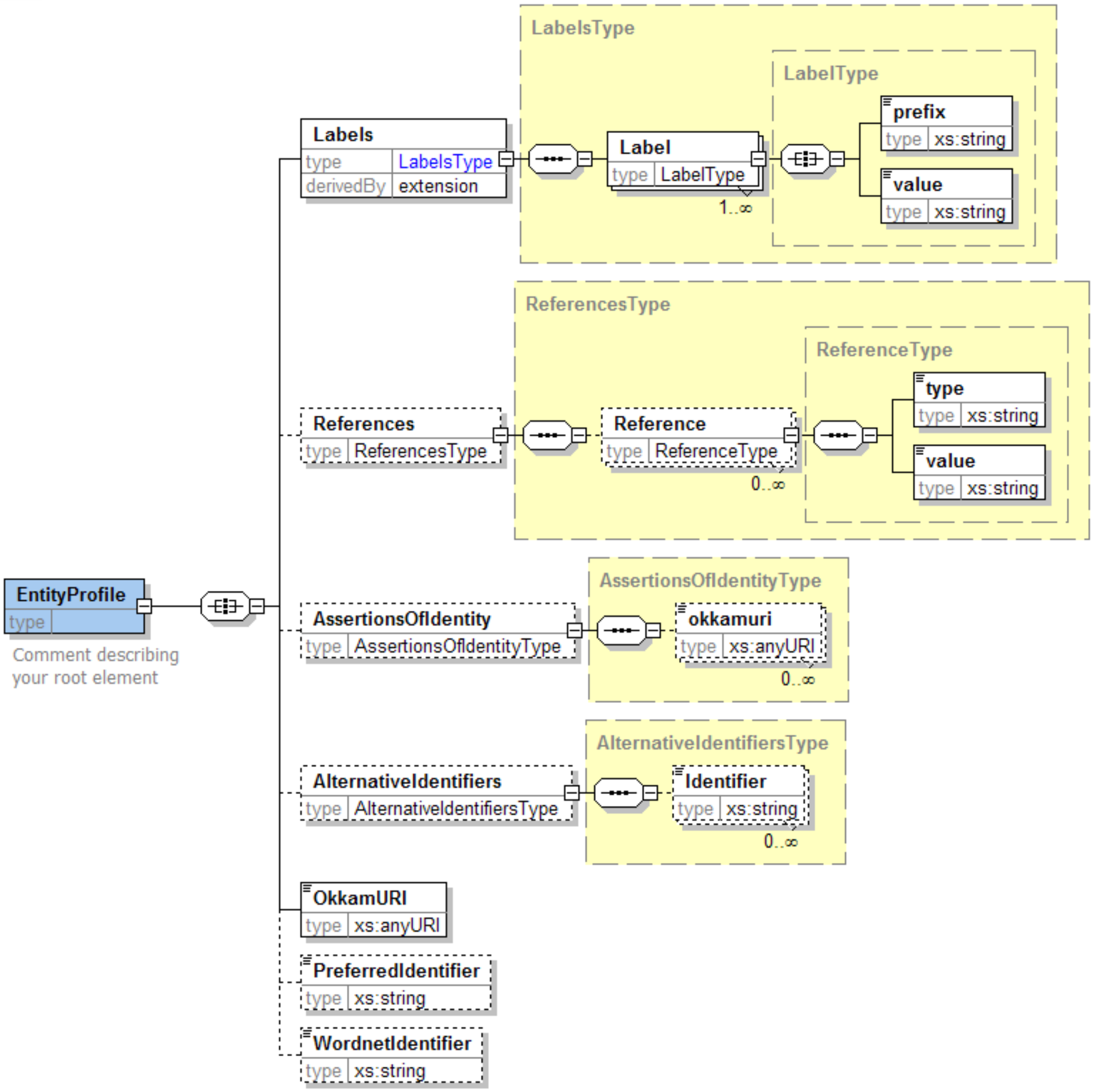

Figure 5.6: Schema of EntityProfile

`References`**:** A list of references to external sources. The reference itself is of type String, the field `type` describes the type of reference, e.g. an HTML document, an ontology, an XML document, etc.

`AssertionsOfIdentity`**:** A list of OKKAM URIs that this entity is known to be identical with.

`AlternativeIdentifiers`**:** A list of strings that identify the same entity in other systems.

`OkkamURI`**:** The OKKAM URI of the entity.

`PreferredIdentifier`**:** The preferred identifier of the entity (either its OKKAM URI or one of the alternative identifiers.

`WordnetIdentifier`**:** The Wordnet Synset ID that describes best the high-level type of the entity.

For the reason that not all of this information is available under any circumstance, some of the elements of the EntityProfile are optional. The minimum information required to publish a new entity is a populated `Labels` element (i.e. a list of name/value pairs that are sufficient to distinguish this entity from all other entities in OKKAM).

Okkam's Public Interfaces

As depicted in Fig.5.7, OKKAM exposes three Web Services:

EntityProfilePublicationService. This service is the implementation of the use-case of Entity Publication, as described in Sect. 4.2.2. It accepts as input an `EntityProfile`, and returns as result the newly-created OKKAM URI if publication succeeded, `null` otherwise.

SearchNG. This service implements most of the use case of Entity Search (Sect. 4.2.1). It accepts an `AnnotatedQuery` as input, and returns an `OkkamURIResult` as output. As described above, the `OkkamURIResult` does not contain the entity profiles of the resulting entities, but only their URIs. This decision was made for performance reasons, as implementational experiments have shown that transmitting a full dataset can – due to its size – be unneccessarily slow. Instead, we added the following service to complete the use case:

ProfileRetriever. This service accepts as input an OKKAM URI and returns the respective `EntityProfile`, or `null` if the URI is not known to the system.

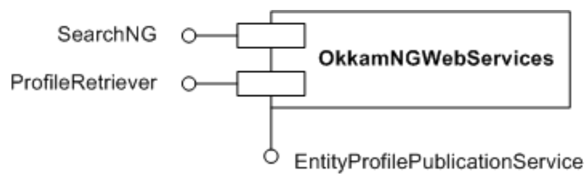

Figure 5.7: The Web Services exposed by OKKAM

The combination of a call to `SearchNG` and subsequent calls to `ProfileRetriever` enables client applications to provide a smoother user interaction, because first results can be shown quite soon after the query was executed, instead of having the user wait for all data to be transferred.

The Java Client Library

As integral part of the OKKAM*DEV* developers' toolkit that we described in Sect. 5.1, a client library for the Java programming language has been developed and deployed. In order to provide low-boundary access to the Web services. Figure 5.8 presents the main classes and methods necessary to make use of the services described in Sect. 5.2.1.

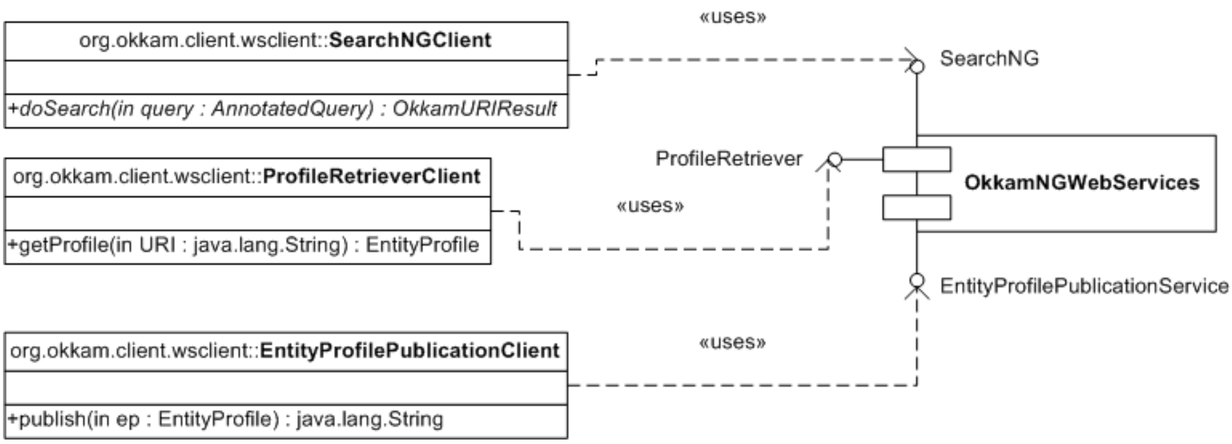

Figure 5.8: Main classes of the OKKAM Java client library

The library hides the implementational details of web-service access and comunication, and consists of the following classes:

SearchNGClient: wrapper for the entity search web service.

ProfileRetrieverClient: easy access to an EntityProfile for a given OKKAM URI.

EntityProfilePublicationClient: creates a new entity in OKKAM with the given EntityProfile.

By use of these Java methods the remote procedure calls and data transfers are executed transparently to the developer. Additionally, the three input and output data structures AnnotatedQuery, EntityProfile and OkkamURIResult do not have to be supplied as XML documents, but are themselves conveniently mapped into normal Java classes by means of the Java Architecture for XML Binding (JAXB)[7]. This mapping allows the developer to create standard instances of Java classes (so-called POJOs – "Plain Old Java Objects"), and eliminates the necessity of interacting with XML APIs such as DOM, or even creating XML documents in string form, both of which are tedious and error-prone procedures.

[7] `https://jaxb.dev.java.net/`

5.2.2 The Internals of Okkam

OkkamCoreNG

OkkamCoreNG is the component that provides the business logic of the ENS prototype. The whole component consists of 65 classes the description of which is beyond the scope of this document[8]. We will thus concentrate on the most important aspects of the architecture, and describe its main functionality – entity matching – in Sect. 5.3.

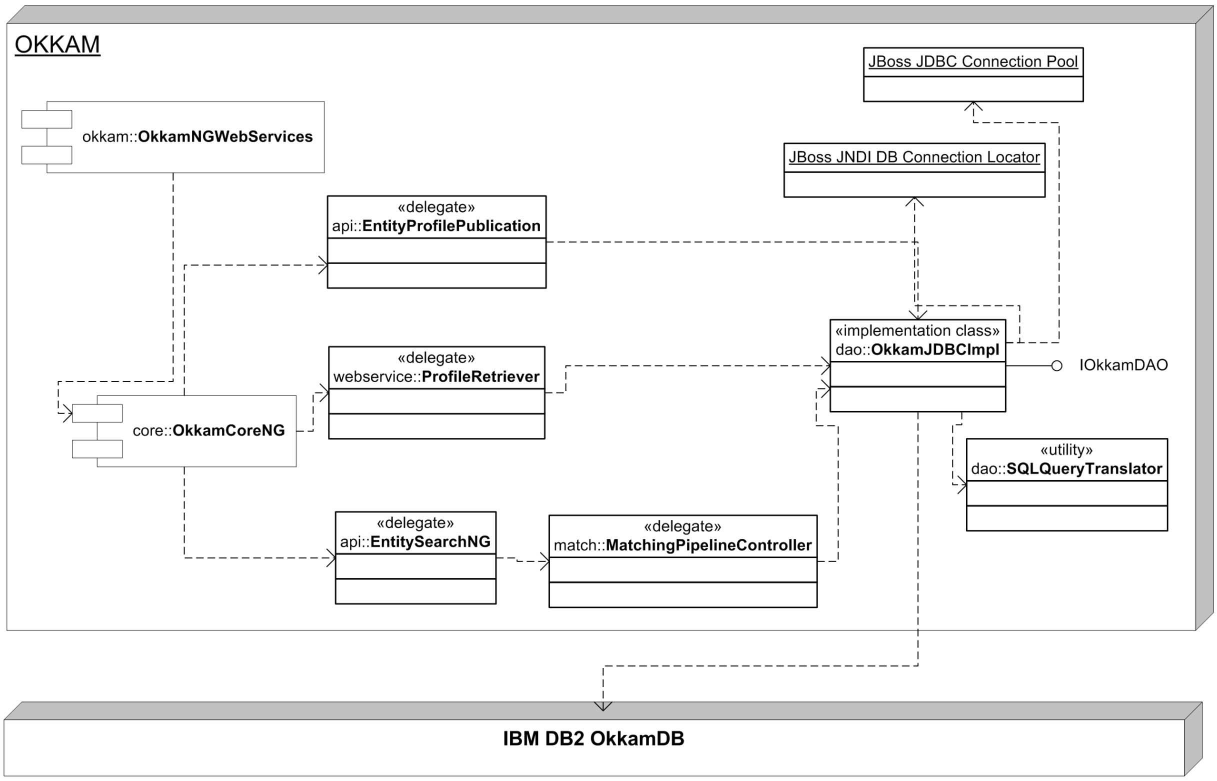

Figure 5.9: Deployment diagram of OkkamCoreNG's main components

Figure 5.9 illustrates the main components of the architecture. The Okkam core functionality is implemented as a Java Enterprise Application which is deployed in the JBoss[9] application server.

[8]The API documentation of all classes is available online: `http://okkam.dit.unitn.it/okkam-apidocs/`

[9]`http://www.jboss.org`

Central to the implementation there is a plugin architecture, named `IOkkamDAO`, which combines the Data Access Object and Factory patterns [87, 37] for access to storage backends which in theory allows for the exchange of relational databases or the use of other types of backends such as XML databases or RDF triple stores without a recompilation of the source code.

For the current prototype we have implemented the plugin `OkkamJDBCImpl` which realizes access to the underlying relational database that is described in Sect. 5.2.2. For performance reasons, the implementation uses a database connection pool which is managed by JBoss, because opening database connections is a time-consuming process which can often be avoided in this way; the database pool is provided by JBoss and accessed via lookup in a JNDI[10] directory.

As the internal counterparts of the exposed web services, we have the following three Business Delegate Objects [86]:

EntityProfilePublication: As the publication process for entities is relatively simple, the business logic is directly delegated to `OkkamJDBCImpl`.

ProfileRetriever: ditto.

EntitySearchNG: This class is delegating to the `MatchingPipelineController` which implements the process of entity matching and ranking. This functionality is described in depth in Sect. 5.3.

For data exchange between a client and the core, the Java mappings of the data structures described in Sect. 5.2.1 are implemented as so-called Transfer Objects [88]. As we will see later, especially the `AnnotatedQuery`

[10] "The Java Naming and Directory Interface (JNDI) is part of the Java platform, providing applications based on Java technology with a unified interface to multiple naming and directory services." (cf. `http://java.sun.com/products/jndi/`).

transfer object plays an important role, as it is potentially enriched/extended by the matching component and then passed through `OkkamJDBCImpl` to the `SQLQueryTranslator` (see Fig. 5.9) which creates an SQL query from its contents, suitable to the relational back-end.

Database

For the first prototype of OKKAM (the pre-decessor of the one described in this work), we implemented two different data persistence modules: one based on a native XML database, and another one built on top of a relational database.

The first rapid prototyping was performed with an XML Database based backend because it allowed us to have a version running in a very short time. The backend is based on the Open Source database *eXist*[11]. Although the flexibility of the XML native database together with the XQuery [94] expressiveness enabled us to complete the backend relatively quickly, we experienced scalability issues. It turned out that the number of entities that can be managed by this backend ranges in the tens of thousands, which is far below our desired goal, for the reason of which we decided to abandon this approach in favor of a proven, industry-strength relational database based backend.

As evident from Fig. 5.10, the mapping from the logical representation of an EntityProfile to a physical table space is performed by providing a master table `Entity` and several dependent tables for the profile components as described in Sect. 5.2.1. The split is straight-forward and self-explanatory.

Two additional tables are present which are not directly explained in the EntityProfile data structure: (i) `ReferenceType` contains records that describe what kind of reference to an external data source a `Reference` has, e.g. an ontology reference, and HTML document, etc. and (ii) the

[11]`http://exist.sourceforge.net/`

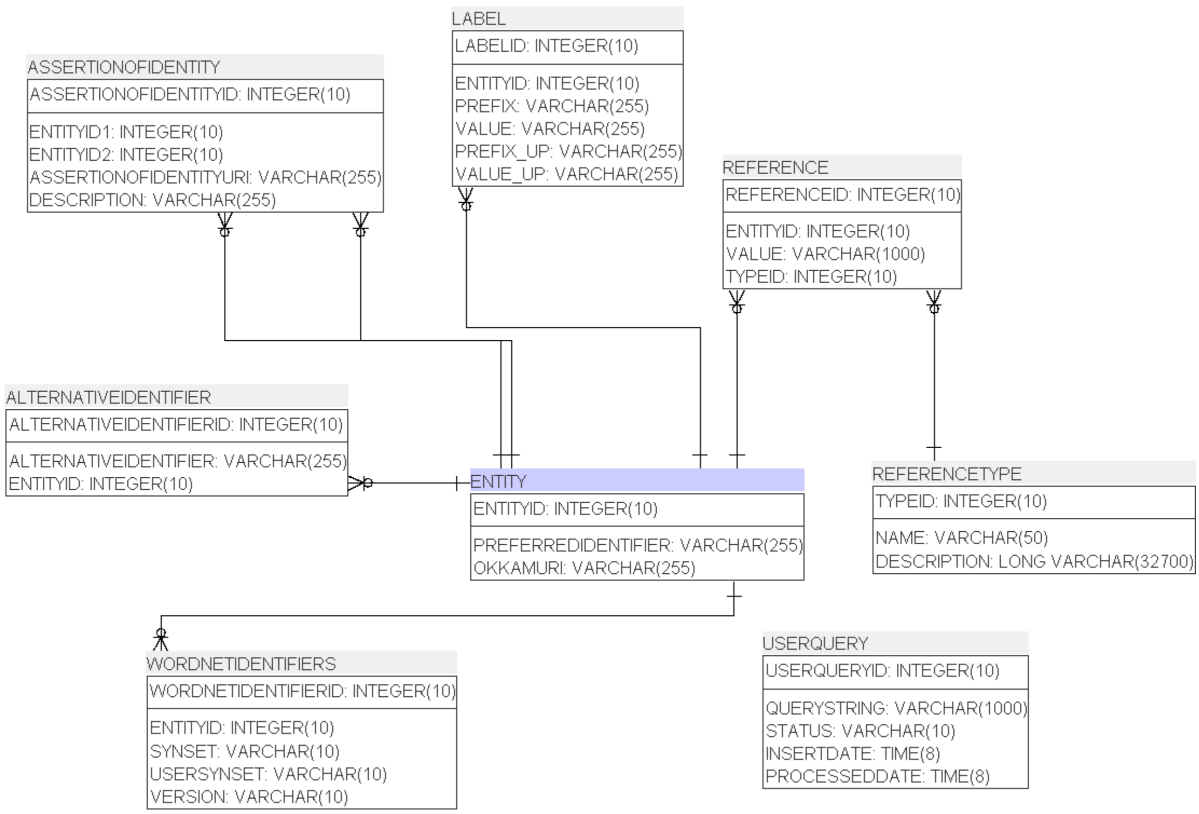

Figure 5.10: Entity-relationship Model of the OKKAM database

table `UserQuery`, which provides a way to keep track of which queries were submitted to Okkam, which status they returned and how long the processing took, in order to facilitate potential machine learning mechanisms in the core.

5.3 Entity Matching and Ranking in Okkam

5.3.1 The Matching Problem

As we have described in Sect. 3.2, there is a substantial amount of related work that deals with the problem of detecting whether two records are the same or describe the same object.

The problem that we are dealing with in Okkam is in our opinion very similar to these, because in relation with what we have defined in Sect. 4.1.4, we see the process of searching for an entity as a matching problem of an entity description Δ against the set EP of all entity profiles[12].

The matching problem in Okkam is however substantially different from the following points of view:

1. The description Δ of the entity that is searched for can be generated by client applications that are of very different nature, i.e. they can have limited capabilities and e.g. only provide a simple query string, while others have additional background knowledge available or can provide (semi-) structured descriptions. It is thus not foreseeable which name/value pairs a Δ contains.

2. Similiar to the first point, the set of entity profiles EP is untyped, semi-structured and may as well hold arbitrary values. The combina-

[12] Note that Δ and EP are "compatible" for matching in the sense that every element $E \in EP$ contains a Δ by definition.

tion of (1) and (2) make a required solution very different from most record-linkage approaches in that these rely on fixed (and/or identical) schemas, because OKKAM cannot provide any meaningful schema.

3. The objective is not deduplication (or Merge/Purge etc.) but rather the production of a ranked list of candidate matches within a time frame of a few seconds. For this reason, unoptimized approaches that perform deduplication by iterating over EP in a serial fashion have to be avoided.

The setting of our problem is thus very similiar to what Pantel et al. describe about their *Guspin* system [70]: due to the high level of heterogeneity on the schema level (or in our case, the absence of such a level), we will pursue a purely data-driven approach for entity matching.

When analyzed in more depth, the possibilities of implementing such a matching component are manyfold. Figure 5.11 provides a mind map of dimensions that can be serve as starting points for an implementation.

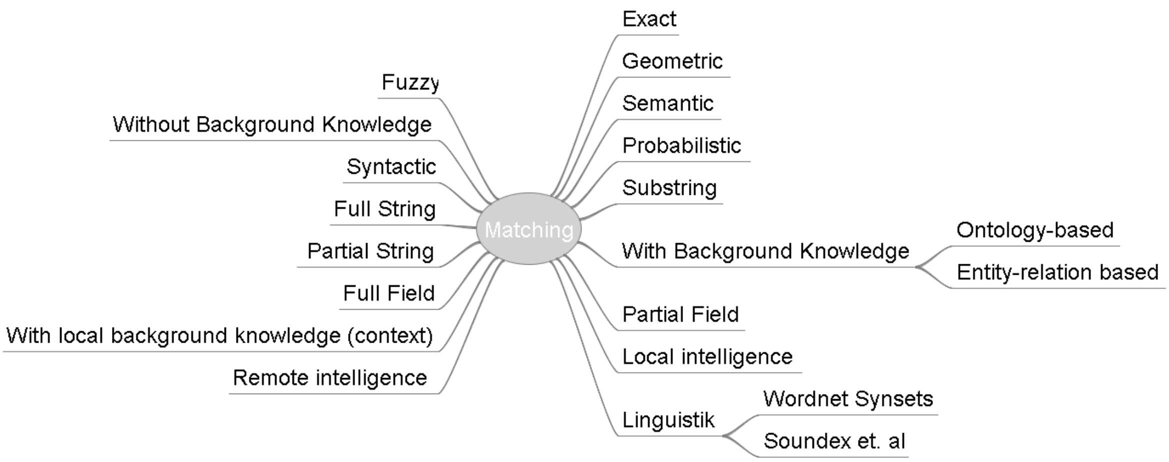

Figure 5.11: Dimensions of entity matching – a brainstorming

Unfortunately, these dimensions are neither always completely disjoint (e.g. matching with background knowledge can in theory be both used to produce fuzzy or exact results), nor is it always possible to define their properties formally in a satisfying way (e.g. is it required for an "exact" match that Δ matches an E perfectly, or that $\Delta \in E$, or is an exact match one that produces the desired entity as the hit with the highest ranking of the result set?).

Furthermore, it is imaginable that different approaches – or even a combination of approaches – produces best results for a certain type of query, and that such approaches be combined in an adaptive way, during runtime.

We come to the conclusion that due to the multi-dimensional character of possible matching approaches, it is most promising not to produce something that relies solely on a single approach, but instead implement an *experimental matching architecture* that allows for pluggable algorithms, which facilitates experimentation and potentially also the creation of an adaptive matching approach in the future.

5.3.2 An Experimental Matching Architecture

According to the "no free lunch" theory of Wolpert and McReady[13], to achieve high performance which is better than the one of a generic algorithm, a suitable set of specialized algorithms is required, which is one of the motivating factors for our implementation of the matching architecture in Okkam. As however the implementation of (i) a whole set of matching algorithms and (ii) an adaptive layer that selects/combines these algorithms to reach an optimal outcome is beyond the scope of this work, we decided instead to provide an infrastructure that is flexible and extensible, and can be equipped with number of algorithms.

Figure 5.12 we give an overview of the main components of the complex internal structure of the matching architecture. The depicted diagram is a hybrid of an UML Activity Diagram (on the left) and an UML Class

[13]See `http://www.no-free-lunch.org/` for a rich source of information.

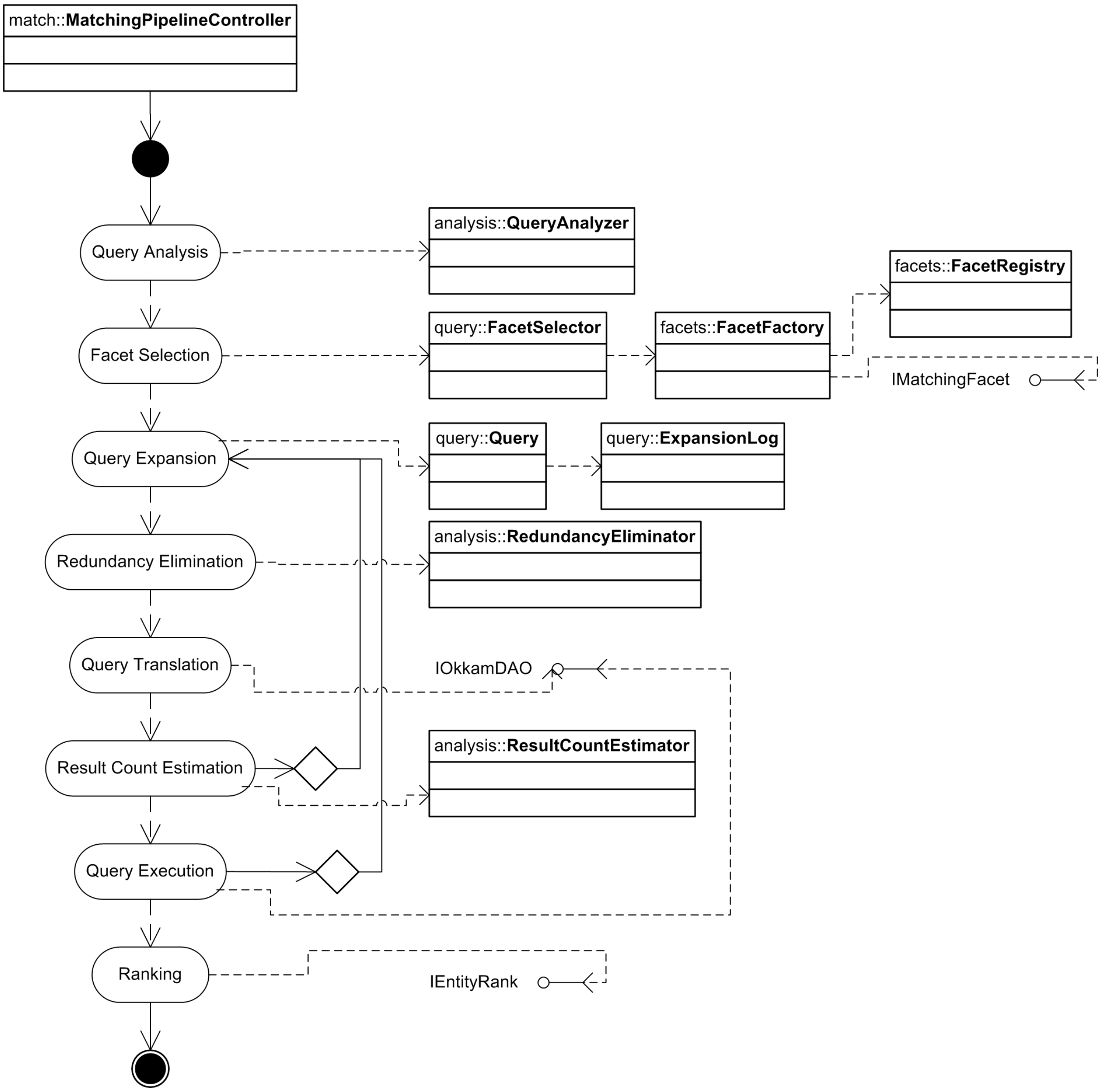

Figure 5.12: OKKAM*MATCH*: Sequence and components

Diagram (on the right), and tries to illustrate the matching process as well as the relevant classes that contain the business logic.

The matching process consists of eight steps:

1. Query Analysis: The AnnotatedQuery object that is received from the external interface of OKKAM*CORE* is analyzed and validated.

2. Facet Selection: In OKKAM, an implementation of a matching dimension as discussed above is called "facet". The plugin-architecture, consisting of `FacetRegistry`, `FacetFactory` and one or more implementations of `IMatchingFacet` (see Sect. 5.3.3 below) constitute the heart of the experimental matching component, as they provide the `FacetSelector` with information about which kinds of facets are available to the system. The `FacetSelector` will then provide a list of suitable facets that can be used in the next step.

3. Query Expansion: The approach for achieving better/more detailed query results in OKKAM is seen as a matter of query expansion (and performed on the `Query` object). The input query is provided to the matching facets, which can add additional conditions to the query – depending on their implementation – which may help to compose a more specific or relaxed query to pass to the storage backend. The expansions are traced in the `ExpansionLog`, with a link to the facets that produced them. In this way, it is later possible to either further expand the query if the result set was too large, or to implode the query if the result set was too small.

4. Redundancy Elimination: It is possible that different matching facets produce the same query expansion. The class `RedundancyEliminator` iterates over the query expansions and removes duplicates.

5. Query Translation: The plugin responsible for interfacing with the

storage backend (a plugin of type `IOkkamDAO` as described in Sect. 5.2.2) translates the (expanded) query into a language suitable to the storage backend.

6. Result Count Estimation: An estimation is generated by `ResultCountEstimator` about how many query results are to be expected. Depending on the number of desired results that have either been supplied in the AnnotatedQuery or taken from configuration defaults, the estimate is either too low, and a query implosion is initiated, or it is proceeded to the next step.

7. Query Execution: The translated query is executed by the storage backend and the result set is retrieved.

8. Ranking: The result set is analyized and ranked against the input query. The ranking mechanism is again implemented as a plugin structure, so the ranking is performed by an implementation of `IEntityRank` that can be selected during runtime. This step is further detailed in Sect. 5.3.3.

The benefit of this approach is that it not only enables the development of a prototype, but – more importantly – it provides an **extensible infrastructure for experimentation**, and thus lays the base for future research. The implementation is made in a way that allows other developers to provide (binary) plugins for (i) exchanging the storage backend, (ii) performing query expansion and (iii) ranking search results, which are the three main aspects of entity matching in Okkam.

5.3.3 An Exemplary Matching Algorithm

To illustrate the viability of the above-mentioned approach, we have implemented an exemplary matching and ranking algorithm that integrates with the matching architecture of Okkam. The objective was to provide an approach that solves the matching problem described in Sect. 5.3.1 and can serve as a baseline and benchmark for future developments of Okkam.

Matching and ranking in Okkam is a two-step process: first, a set of candidate matches is retrieved from the storage backend, which, in the second step, is ranked with respect to the input query. With this approach we try to alleviate the problem that while storage backends such as relational databases perform extremely well in its main purpose, the production of ranked query results is not a "native" feature and thus hard to achieve. Furthermore, it allows us to apply methods for ranking that such storage backends simply do not provide.

Due to the differences between the matching problem in Okkam and much of the related work, as discussed in Sect. 5.3.1, we decided to pursue an approach that is both schema-independent (entities in Okkam are not described with a fixed schema) and type-independent (entities are untyped). The solution we came up with is to see the EntityDescription Δ_e of an entity as a type of document which we can compare against the EntityDescription Δ_i that was provided in the input query. By computing a similarity between the two, and doing so for all candidate matches, we are able to provide a ranked query result.

The resulting algorithm, called `StringSimilarityRank`, is the following (with Δ_e being denoted by *de* and Δ_i by *di*):

```
d = concatenate(valuesOf(di))
forall candidates
   c = concatenate(valuesOf(de))
   s = computeSimilarity(d,c)
   rankedResult.store(s)
rankedResult.sort()
```

The function `valuesOf()` returns the value parts of the name/value pairs that form part of Δ, while `concatenate()` creates a single string from a set of strings; the combination of the two creates a "document" that can be matched against another, which is performed by the function `computeSimilarity()`.

To compute the similarity between two descriptions, we have selected the Monge-Elkan algorithm [64] as the result of extensive testing and evaluation of different algorithms, as described in detail in Sect. 8.2. The matching results that can be achieved with this approach are satisfactory as a baseline, and are reported in Sect 8.3. The runtime performance of the implementation meets the requirements and is reported in Sect. 8.1.

It is important to note that this matching approach is completely generic and "un-semantic", in that it neither uses background knowledge nor any kind of type-specific heuristics to perform the described matching.

5.4 Requirements Review

In this section we will provide a short review of the requirements established in Sect.4.3, to analyize to which extent the current prototype is a suitable implementation of an ENS.

R-1	This requirement has been complied with (see Sect. 5.2.1).
R-2	This requirement has been complied with on the server side (see Sect. 5.2.1) and through the two client applications described in Sect. 6.
R-3	Modification of existing data and addition of data about an entity has not been implemented yet.
R-4	Identity detection is not yet implemented, but the data model provides for the representation of identity (see Sect. 5.2.1 and 5.2.2).
R-5	An experimental web interface for search is available (see Sect. 6.3).
R-6	Web services for query and publication of data are available (see Sect. 5.2.1).
R-7	A *generic* batch interface for the okkamization of data has not been implemented yet. A prototype for the okkamization of RDF graphs has been developed for the experiment described in Sect. 8.3.
R-8	The current prototype is not distributed.
R-9	No special means for high availability have yet been implemented.
R-10	Persistence has been implemented on top of an RDBMS.
R-11	No deletion of records about entities is currently permitted. Apart from that, no special security measures have been implemented.
R-12	The average execution time currently lies in the range of 2 seconds (see Sect. 8.1).
R-13	The system allows for multi-tasking and accepts parallel requests.
R-14	The system runs on a standard dual-processor server and uses a maximum of 512 MB or RAM. The RDBMS runs on the same machine and uses additional RAM, usually 512MB. No further analysis have been performed regarding resource usage.
R-15	Experimental results are described in Section 8.2.
R-16	Experimental results are described in Section 8.2.
R-17	This issue has not been addressed yet.
R-18	This issue has not been addressed yet.
R-19	This issue has not been addressed yet.
R-20	The service is free-of-cost.
R-21	Usability has not been evaluated yet.

Table 5.2: Requirements review of the OKKAM prototype

Chapter 6

Okkam Applications

To illustrate the viability and the usefulness of the approach, we have developed two exemplary applications that both have been strategically selected from the area of content creation. The reason for this selection was that the success of an approach such as Okkam depends entirely on a certain saturation of suitable content ("critical mass"), and in effect on the availability of tools for the creation of such content.

The first tool is called Okkam*4P* [17], which is in fact a plugin for the widely-used ontology editor Protégé. This plugin enables the creator of an ontology to issue individuals with identifiers from Okkam, instead of assigning local identifiers that bear the risk of non-uniqueness on a global scale. The choice for this tool was made based on two criteria, namely the target audience being rather 'expert' users of the Semantic Web, and, secondly, the very wide usage of the Protégé editor, which makes it a promising candidate for a rapid distribution of the tool.

The second application is called foaf-O-matic [12], a WWW-based service for the creation of okkamized FOAF profiles. Indeed, FOAF is in our opinion one of the few success stories of the Semantic Web so far, as it is one of the few applications that really contributed to the creation of a non-toy amount of RDF data, with the special restriction that the agreement on URIs for persons is extremely low [46]. As content creation tools for FOAF are mostly rather prototypical, we decided to create a completely new application that both serves the user due to state-of-the art technology and creates okkamized FOAF profiles.

As an additional, generic tool for the re-use of identifiers, we have developed OKKAM Web Search, which helps finding entity identifiers under circumstances where no specialized applications as the ones mentioned above are available.

The applications are described in more detail in the following.

6.1 Okkam*4P*

The application we are going to present makes use of the public OKKAM infrastructure, in the area of ontology editing. It aims at demonstrating the advantages of such an approach as a way to converge on common URIs for newly created semantic content. Indeed today, a common practice in ontology editing is the creation of new *local* URIs for any newly created instance. Here we present a Protégé plugin, named OKKAM*4P*, which supports the good practice of looking up for pre-existing *global* URIs when editing a new RDF/OWL knowledge base. The plugin is an extension of the "individual" tab of Protégé's OWL module. The main difference is that, when an instance is created, the user has a chance of looking for a pre-existing URI for the corresponding individual in OKKAM, and to assign this URI to the instance. The plugin is available and tested for the latest official release of Protégé, version 3.3.1, and the beta version 3.4.

6.1.1 User Perspective

In our vision of a functioning OKKAM infrastructure there is the notion of the so-called "OKKAM-empowered tools", which are standard end-user applications (e.g. word processors, HTML/XML/OWL editors, web-based authoring environments – like blogs, forums, multimedia publishing and tagging applications, etc.) extended with functionalities which facilitate the creation of okkamized content through the use of the OKKAM*PUBLIC* infrastructure. Protégé falls into this category. It is probably the most widely used editor for the creation of RDF/OWL knowledge bases (KBs), and provides vast extensibility through a plugin architecture, which makes it highly suitable for empowering it with OKKAM functionality.

The plugin presented in this paper essentially assigns a global unique identifier called (the "OKKAM ID") to a newly created individual, rather than relying on manual input of the user or the standard automatic mechanism of Protégé. To this end, it implements the use-case illustrated in Fig. 4.4: based on the data about an individual that are already provided in the KB developed by the user, it queries OKKAM*PUBLIC* to see whether an identifier already exists which can be assigned to the new created individual, otherwise a new identifier would be created.

To use this plugin, the user selects an individual and right-clicks on it. A context menu will pop up, in which the item "Get Okkam ID" is the entry-point to the functions of the plugin, as illustrated in Fig. 6.1.

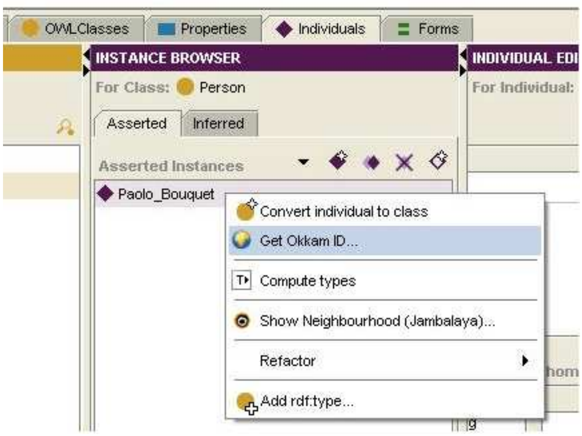

Figure 6.1: Assigning a global identifier to an individual

Once clicking on this menu, the plugin starts to collect the properties of this individual as specified in the KB, and presents a new dialog (see Fig. 6.2) displaying the information that is available for querying OKKAM*PUBLIC* in order to see whether an identifier for this entity already exists.

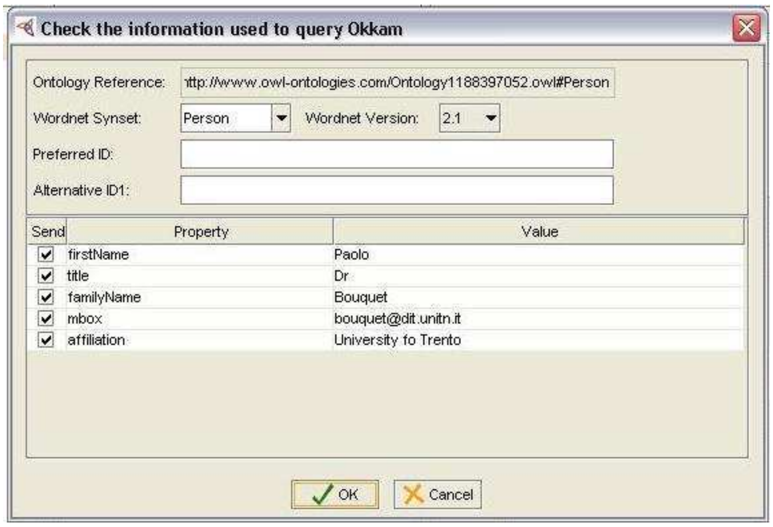

Figure 6.2: Selecting query parameters in OKKAM*4P*

The properties that are gathered by the plugin to construct a query are the following:

- Ontology Reference: it is the reference of the ontology which the chosen individual belongs to. It is loaded automatically by this plugin, and it is read-only for users. If the ontology is publicly available, it can potentially be of use for the server-side matching mechanisms to improve search results for the individual.
- Wordnet Synset and Wordnet Version: provides a hint about a top-level class which the chosen individual belongs to. This has to be set by the user.
- Preferred ID and Alternative ID1: if the user wishes to use another

identifier in other systems to identify the chosen individual, a user can input this identifier here. These two items are optional.

- Individual Properties: the plugin loads each property of the chosen individual automatically. The user can also deselect some properties which are thought to be unnecessary to find the Okkam ID of the individual at hand.

After submitting this form, the plugin launches a thread to query Okkam for matching entities by calling its web service. After searching, a list of entities that match the description for the new created individual will be visualized to the user, as illustrated in Fig. 6.3

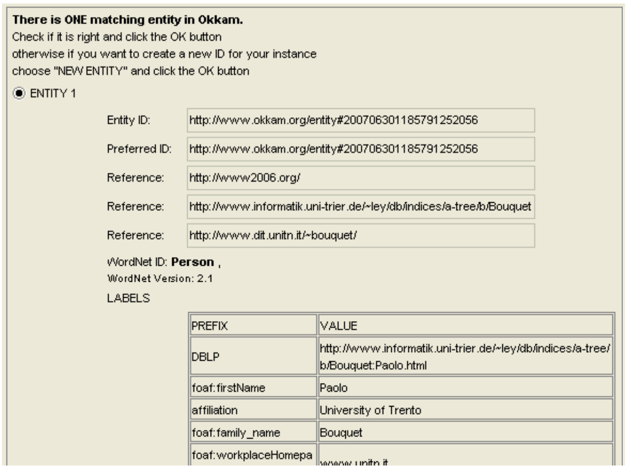

Figure 6.3: Query result of with matching entities that already have an identifier in Okkam.

The user now has the option to select one list entry as "the same" as the newly created individual and re-use the global identifier in the local KB (therefore the ID of the newly created individual will be replaced by the Okkam ID in the KB); otherwise the user can choose to create the individual as a new entity in Okkam*PUBLIC*, in which case the information selected in Fig. 6.2 will be inserted into Okkam repository, the new Okkam ID will be retrieved and assigned to the local individual.

6.1.2 Technical Perspective

The hierarchy of primary classes provided by and used in this plugin is illustrated in Fig. 6.4 in the appendix. In the following we describe the function of each class displayed in Fig. 6.4.

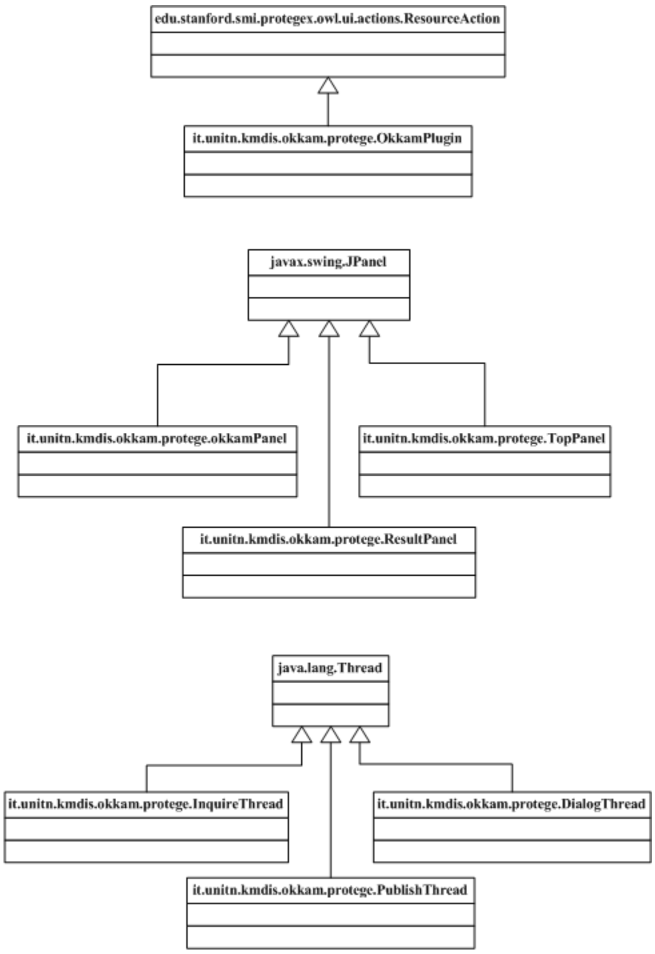

Figure 6.4: Main classes of OKKAM*4P*

The class *OkkamPlugIn* is the most principal class. To extend the “Individuals” tab in protege, it needs to inherit the class *edu.stanford.smi.protegex.owl.ui.actions.ResourceAction*. This effects that the menu item “Get Okkam ID...” will appear in context-menu when the user right-clicks on a individual.

The class *okkamPanel* and *TopPanel* are used to compose the information window (see Fig. 6.2); the class *ResultPanel* is used to show the query result window(see Fig. 6.3). All of them inherit the class *javax.swing.JPanel* to present a window to users.

In this plugin, we make use of web services to interact with OKKAM. The tasks of searching for matching entities and publishing a newly created entity are fulfilled by calling the webservice "EntitySearch" and "EntityPublication" respectively. As complex queries may have a considerable runtime, in the initial version of OKKAM4P, users would see nothing but a gray window until the result returned from the webservice. In the current version, we moved the plugin to a multi-threaded architecture. Three classes which inherit class "java.lang.Thread" are new to this version.

The class *InquireThread* is used to call the webservice "EntitySearch", it is launched when the user submits the information to search for matching entities. The class *PublishThread* is used to call the webservice "EntityPublication", it is launched when the user decides to publish a new entity to OKKAM. The class *DialogThread* is used to show a dialog during the process of searching or publishing, this dialog is meant as a user-friendly interface to inform the users that the process is running.

6.1.3 Benefits

To achieve a substantial diffusion of okkamized content, a set of user-friendly OKKAM-empowered tools is necessary, because – as the rather slow adoption of Semantic Web technologies has shown – the mass of content creators (i.e. the users of the WWW) seem not to be extremely motivated to follow developments beyond the coding of HTML documents.

With OKKAM4P we are making the first and very important step towards the creation of such a suite of tools. We address the community that is "closest" to the issues addressed by the approach, and provide them with the means of creating okkamized RDF/OWL KBs. The aim is to prove that – with the systematic a-priori use of global identifiers for entities – the vision of RDF documents as a single, global, *decentralized* and *meaningful* knowledge base can in fact become reality, without having to deal with many of the difficulties of information integration, such as the ex-post alignment of entities.

6.1.4 Future Work

Several improvements are scheduled in the near future. One is the general "elevation" of the tool to a more production-quality standard, including the usual aspects such as extended documentation, code improvements, etc. Secondly, as the plugin is currently implemented as an extension to the OWL part of Protégé, which has the negative effect that Knowledge Bases developed in plain RDF(S) cannot benefit from its functionality – a circumstance which we are currently investigating. Finally, additional features such as offline and batch operation, as well as automatic retrieval and assignment of OKKAM identifiers to existing KBs, are already in the design phase.

6.2 foaf-O-matic

6.2.1 FOAF and the Problem of Identity

The FOAF initiative[1] provides the definition of a set of specifications and tools based on the W3C's RDF language [43] that allow agents (people, organization, groups etc.) to describe themselves, their place of work, their main interests, education institutes etc. Furthermore, the set of properties associated to a FOAF agent are conceived to state some relationship involving other agents. The most important and used is the "`foaf:knows`" object

[1] The web page of the project is: `http://www.foaf-project.org`

property relating FOAF Person resources. In simple words, by means of this object property it is possible to state who is friend of whom and share this knowledge on the web in a machine readable way.

The vocabulary of FOAF is reasonably expressive, although still in evolution, and allows to express different types of information describing a person [2].

Analyzing the set of properties describing a FOAF *person* entity, it becomes clear that the best identifier *currently* available is the unique code obtained by encoding a person's email address in the property `foaf:mbox_sha1sum`. Indeed, an email address is uniquely identifying a mailbox of a person. Furthermore, often people use the same email address for long periods of time and this fact make the email address useful to identify persons along this period.

Any FOAF file describing a person represents an RDF graph. Every single graph is supposed to be merged with other graphs collapsing the nodes identifying the same person. Namely, if in two graphs somewhere the same unique code derived by the mail address is used, then both graphs contain some kind of information about the same person, therefore the graphs can be merged enlarging the network of "friendship". By means of this procedure it is possible to build a bigger graph containing all the information stated by the respective social network.

Analyzing superficially this process, everything seems to be at the right place, but going a little bit deeper some problems arise. The problems are related mainly to the weakness of the use of the email address based code as identifier. Indeed, an email address is not a good identifier for the following reasons:

- people change email address (change work/study institution, choose

[2] For more detail about the FOAF vocabulary see `http://xmlns.com/foaf/0.1/`

better provider, drop over-spammed[3] email address, etc...)

- people use more than one email address depending on the context of use (work, on-line gaming and shopping, 'night activities', family and friends relationship, etc...);
- email addresses can act as proxies for more than one person.

The facts listed above raise the following problem: different actors could use different email address to identify the same person (agents). Thus, a complete merging of all the information regarding a person is no more even possible.

The "state of the art" in applications that generate such FOAF profiles to date has been a web-application called foaf-a-matic [4]. Strikingly, the RDF descriptions created by this tool issue no identifier at all for the entity that represents the author of the description (i.e. "me"), nor for the friends. Instead, an RDF blank node is used, which by no means produces the expected result: instead of a simple set of triples of type

`A foaf:knows B`

it generates a structure that in Description Logics syntax can be described as

$$\exists\ A\ (\ A \circ X \wedge A\ foaf : knows\ (\exists B (B \circ Y)))$$

with the $\circ X$ and $\circ Y$ operator representing the properties that describe A or B, respectively. This means that in the case of an RDF graph merge of several of these descriptions, a trivial query such as "all the people who say that they know me" first of all has to be posed as "all the people who say that they know something which has my `foaf:mbox_sha1sum` property",

[3] over-spamed means that this address is a constant target of spam email

[4] `http://www.okkam.org/projects/foaf-o-matic/`

and the result would be a set of blank node identifiers, which by definition have the properties of (i) being volatile, e.g. they potentially change every time the same experiment is run, and (ii) being only valid identifiers within the resulting merged graph.

These shortcomings have motivated us to develop the foaf-O-matic application, to showcase how this identification and reference problem can be tackled and resolved by means of adding globally unique identifiers that are not dependent from the context of use.

6.2.2 User Perspective

As illustrated, what is missing in a FOAF Person description is a unique and sole identifier.

The approach applied for tackling the analyzed problem is to provide a tool allowing users to create/integrate FOAF person descriptions with identifiers contained in, or generated by, Okkam. Thus, what is needed is a new application extending the functionalities provided by the foaf-a-matic application. In order to underline the historical relation with the former application, this new web-based tool has been named foaf-O-matic (with the 'O' underlining the integration with Okkam.)

It is important to notice that the aim of creating the foaf-O-matic application is not only to replace the slightly 'obsolete' foaf-a-matic application and providing a pretty layout and new description fields. The focal point of the new application is to allow users to integrate Okkam identifier within their FOAF document in a user-friendly way. In this way, it will be possible to merge more precisely a wider number of FOAF graphs describing a person's social networks, enhancing the integration of information and reach more easily the goal of the FOAF initiative.

A view of the new layout of the application is given in figure 6.5. As it is possible to see from the figure, the main layout is split in two columns:

Figure 6.5: FOAF-O-MATIC The main interface of FOAF-O-MATIC.

the left one for the `foaf:PrimaryPerson` description, and right one for the friend management. On the top of this two columns facilities to upload already defined FOAF files are presented. At the bottom, a "generate FOAF" button is present that trigger the generation and visualization of the FOAF file in a text area.

Without going too much into details, the FOAF-O-MATIC is meant provide the following set of functionalities:

- **Upload a FOAF file**. This functionality is meant to allow the upgrade of already defined FOAF descriptions and enhancing it with OKKAM identifiers. The file can be loaded providing either its Web URL, loading the file from the file system as is possible to see in the area marked with 1 in figure 6.5.

- **Describe the `foaf:PrimaryPerson` aka 'yourself'**. This functionality supersedes foaf-a-matic by providing of a wider choice of description fields some under testing FOAF properties. For a matter of dimension, the input for has been split in three collapsible panels presenting in the top part the standard description fields, in the middle part some extra information fields (i.e. `birthday`), and in the bottom part some `chat-id` related information fields (i.e. `yahooChatId`). A view of this part of the application is presented in figure 6.5 in the area marked with 2.

- **Add and describe friends**. This functionality is meant to allow users to provide a description of the friends they want to add to their social network. The information provided will be used to inquire OKKAM and retrieve a list of candidate entities corresponding to the described friends. If no entities will be found in OKKAM a newly created entity identifier will be provided. If none of the candidate entities match the user requirement in terms "recognition" a new identifier will be provided as well. "Okkamized" entities[5] will be marked in a special way. A view of this part of the application is presented in figure 6.5 in the area marked with 3 and 4. Notice that an OKKAM identifier is now part of the description of the described friend.

- **Select one Okkam entity for each described person**. This functionality is meant to allow the user to choose which is the entity representing the described person among the one matching such description within OKKAM, if any. The chosen entity identifier is used in the definition of the RDF FOAF file as value of `rdf:about` attribute of the described person. The list of candidate OKKAM entities is presented in a pop-up panel. The user can select the correct entity by pressing the "Select" button associated to the entity, or to state that none of the retrieved entities correspond to the describe person by pressing the button "None".

- **Retrieve the new FOAF description**. The FOAF RDF description containing the informations provided by the used is presented in a text area below the description areas. The FOAF RDF description containing the information provided and integrating an OKKAM identifier where chosen, is generated every time the "generate FOAF" button is pressed. The content of the file reflect the present state of the description provided by the user.

6.2.3 Technical Perspective

The framework used for the development of FOAF-O-MATIC is ICEFaces[6] open source project. ICEfaces is the most widely distributed enterprise

[5] entities which has been assigned an OKKAM identifier

[6] `http://www.icefaces.org/`

Ajax[7] framework on the market today, providing a rich library of Ajax components. The main benefit of Ajax is that it gets rid of the usual submit/reload mechanism of Web forms and enables the creation of very user-friendly interfaces comparable to modern desktop windowing systems.

The primary goal behind the ICEfaces architecture is to provide a familiar Java Enterprise development model, and completely isolate them from the complexities of low-level Ajax development in JavaScript. The key to the ICEfaces architecture is a server-centric application model, where all application logic is developed in pure Java, and executes in a standard Java Application Server runtime environment.

The ICEfaces Framework is an extention to the standard JSF[8] framework, with the key difference in ICEfaces relating to the rendering phase. In standard JSF, the render phase produces new markup for the current application state, and delivers that to the browser, where a full page refresh occurs. With the ICEfaces framework, rendering occurs into a server-side DOM and only incremental changes to the DOM are delivered to the browser and reassembled with a lightweight Ajax Bridge.

6.2.4 Benefits and Future Work

FOAF-O-MATIC is an extended service for the creation of FOAF profiles, which relies on the OKKAM infrastructure for issueing the "friends" with globally unique identifiers, and thus solving a-priory some of the issues of social network applications, illustrated for example in [69].

For the next steps we plan to extend FOAF-O-MATIC in order to get some experience with OKKAM and its matching algorithms. The benefit of the FOAF application is that there are many FOAF files distributed over the Internet which provide a good training base for the matching

[7] Asynchronous JavaScript and XML - http://www.ajaxprojects.com/

[8] JavaServer Faces - http://java.sun.com/javaee/javaserverfaces/

algorithm. With the FOAF application we want to tune OKKAM's and FOAF-O-MATIC's algorithms. With that experience we explore further application and improve OKKAM over time and application domains. Also a scalable architecture with a fuzzy entity identification are subject of investigation.

6.3 Okkam Web Search

OKKAM Web Search is showcase for the generic search and re-use of entity identifiers. It allows users of systems for which no dedicated OKKAM functionality exists, to search for entities and use the respective identifiers in their information systems. This is especially applicable for cases where only a small amount of entities is to be annotated that does not require automated software support.

The search interface on the web consists of only one field into which users can enter a set of strings to characterize the entity they are searching for. Subsequently, the application presents a list of results retrieved from OKKAM through its standard web service API. It furthermore provides a details page for every entity, which displays the EntityProfile, as depicted in Fig. 6.6. The OKKAM URI for an entity appears at the top of the page, and can be copied and used in any application for annotating entities.

The search functionality is OpenSearch[9] compatible. OpenSearch is an initiative to describe search engines in a structured, uniform way so that they become generically usable in search clients. This means that with the help of the OpenSearch Description Document [24] that has been developed for OKKAM Web Search, it is possible for example to integrate it directly into the search toolbars of web browsers, as illustrated in Fig. 6.7 in the case of the popular *Firefox* browser.

[9] http://www.opensearch.org

Figure 6.6: Okkam Web Search displaying entity details

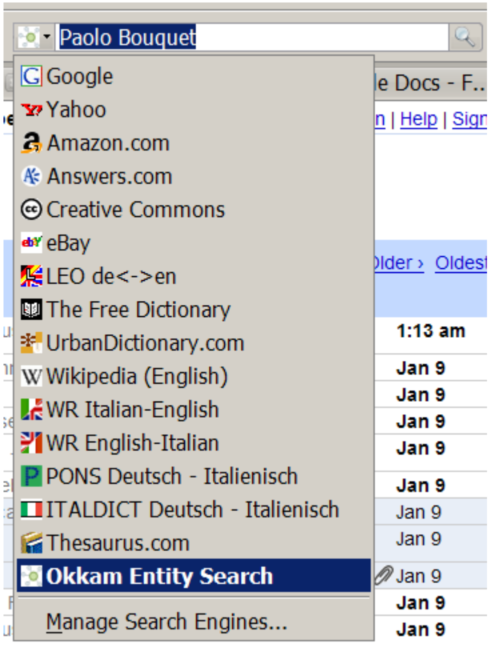

Figure 6.7: Okkam Web Search as Firefox search plugin

Future plans for the application include providing entry points for editing entity data or publishing new entities through the web. Another important goal is to make the search results available in RDF format, in order to make OKKAM's EntityProfiles directly usable in Semantic Web browsers, such as the *Tabulator* application[10] promoted by Tim Berners-Lee and the W3C.

[10] `http://www.w3.org/2005/ajar/tab`

Chapter 7

Application Scenarios

Entity-aware metadata open up opportunities for new entity-centric applications and eases the implementation of other applications and tasks especially in the area of information integration. The core benefit of the proposed entity-centric approach is that we introduce a way to know when in two places one speaks about the same entity (because the same identifier is used) easing the interlinking and integration of information accessible in or through metadata.

In this section we consider applications in the area of digital libraries, in the area of News and Media and in the area of the Semantic Web, which can directly profit from our approach.

7.1 Digital Library Integration

The main purpose of digital libraries (DL) is to mediate between available content and a targeted library user community. This is achieved by content preselection, content structuring, content enrichment with metadata, and by the provision of library services mainly for enabling content selection. In addition, digital libraries also play a role in the preservation of digital content.

Metadata and its management are a central issue in digital libraries as it has already been in traditional paper libraries. Metadata is a medium for structuring and enriching the content managed in the library collection and for easing content management, access and use.

Digital libraries have seen a substantial progress in the last years. Various digital libraries have been set up and are now operational see e.g. the ACM digital library[1] or the Alexandria Digital Library [50]. Furthermore, there was a considerable development in the creation of Digital Library Management Systems. These are generic systems that ease the setup of new digital libraries, e.g. Fedora[2], BRICKS[3], and Greenstone [97].

Currently, in the digital library domain there is a clear trend towards:

1. the federation and reuse of existing resources (content, metadata, services)[4]

2. the provision of additional value-added services beyond simple search

3. a growing involvement of the community in content creation and enrichment processes triggered by the success of systems such as Wikipedia, Flickr, etc.

Entity-aware metadata management can play an important role for information and metadata integration, as it is in the core of federation and the reuse of existing resources (trend 1). Also, for the development of value-added (entity-centric) services on top of the managed and integrated metadata (trend 2).

[1] `http://portal.acm.org/`

[2] `http://www.fedora-commons.org/`

[3] `http://www.brickscommunity.org/`

[4] as an example for federation, the ACM digital library now also offers metadata for publications of other publishers. See `http://portal.acm.org/guide.cfm`.

7.1.1 The Problem of Unidentified Entities

Using metadata for information reuse and integration requires some form of coordination with respect to the formats and values used in creating metadata records. In a controlled environment, this can be enforced by design, but in more decentralized environments this is a major challenge. For this reason, large investments have been made in creating shared metadata schemata - in order to control the situation on the schema level-, and controlled vocabularies - in order to reduce the problem on the level of the metadata attribute values-, together with rules and conventions for the creation of metadata records.

For example, in the (digital) libraries area we can mention the controlled vocabularies of the ACM classification schema[5], and the Library of Congress Subject Headings [21], and the shared metadata schemas of MARC[6], METS[7], and Dublin Core [95]. For the situations, in which no common schema and/or vocabulary is adopted, methods and techniques have been developed for matching and aligning metadata and vocabularies in an automatic or semi-automatic way [76].

The standardization efforts listed above, including the techniques for aligning different vocabularies, have strongly reduced metadata heterogeneity on the schema level. However, integration on the instance level remains a major challenge, to a substantial part due to the heterogeneity of describing the same entity between different metadata collections.

To stay in the area of (digital) library management systems, the usual process for describing artifacts is to fill *data* into a certain schema. Even in more advanced systems this make heavy use techniques from Information Retrieval and search for access to their content, e.g. the one described

[5] `http://www.acm.org/class/1998/`
[6] `http://www.loc.gov/marc/`
[7] `http://www.loc.gov/standards/mets/mets-home.html`

in [36], a certain basic schema is used as the minimal description of an artifact. During this annotation process, *names* for entities such as persons, institutions, or events, are commonly inserted into the chosen schema as textual strings, either by hand, or by some underlying information extraction process.

Let us for example refer to the definition of `dc:creator` in DublinCore. The comment to the element set goes as follows: "Examples of a Creator include a person, an organization, or a service. Typically, the name of a Creator should be used to indicate the entity" (cf. [30]). As we already pointed out, in general using names for referring to an entity is not very satisfactory, because of the danger of ambiguities, the lack of precision, the use of variants and the failure to support integration.

One aspect that illustrates the difficulty of identifying entities by matching their actual string values are the possible causes of variations. Different naming may arise from issues such as mis-spelling, the use of abbreviations or the actual change of the name over time. More complex causes are variations between the different contexts (e.g. the identifier of a person in an email is given by the email address whereas in a publication is given by the author's name) or variations between different workspaces.

Our point can be underlined by considering DBLP author search[8]: on a quest for an article of which we know that it was written by an author named "Lee" we encounter more than 5800 *author* entries; if we happened to know more details, searching for "Yong Lee" still delivers 65 results, and the decision whether the paper we are looking for was written by "Jae Yong Lee", "Jae-Yong Lee" or "Jaeyong Lee", and the decision whether these strings all describe the same person or not is left to the user (which still might be unlucky if the paper in question is registered as written by "J. Y. Lee").

[8] `http://www.informatik.uni-trier.de/~ley/db/indices/a-tree/index.html`

Some systems try to solve this problem by providing (local) lists of already-registered names for certain fields, with relatively small effect: first of all, the above problem is not solved, and secondly, the effect of such a "standardization" is of local character: if information from different systems would have to be integrated, one would again run into disambiguation, deduplication and entity reconciliation problems, which are generally hard and very costly to solve (see Section 8.4).

7.1.2 Metadata Integration

The wide adoption of XML as exchange and representation format has solved some of the syntactic and low-level processing problems in metadata integration e.g. in parsing, and the work in the definition of metadata standards, the creation of metadata crosswalks [27] and metadata mappings contribute to easing problems of schema level integration. This still leaves open the issue of entity level (or record level [101]) integration as introduced in section 1.2.

The transition to entity-aware metadata management using global IDs for entities does not only ease the entity-level integration of metadata. It also leads to a deeper integration, potentially improving the quality of the integration result considerably. The fact that references to the same entity can be recognized without any processing effort (use of the same ID) contributes to making this process cheaper. An analytical evaluation on the potential speed up of entity-level metadata integration can be found in Section 8.4.

Furthermore, consistency checking would be improved, since it would be easier to recognize whether conflicting attributes actually belong to the same entity. However, conflict resolution and, if required, data conversion on attribute values are still required on top of the ID-based metadata record alignment.

The referencing of entities in metadata records via strings representing entity attribute values leads to multiple representations of the same attribute(s) of an entity, if this entity is referenced more than once. This does not only results in redundancy. Also, it may lead to inconsistencies, if different variants are used for the attribute value as it has been illustrated in Section 7.1.1 for the example of author names. This is already true within a single metadata collection, but the problem gets worse when different metadata collections, which have been developed independently of each other, are integrated. The redundancy and the resulting inconsistencies can be avoided by using entity-aware metadata management, where every entity (and its attributes) is only represented once in each metadata collection.

Furthermore, a deeper integration is achieved, because individual metadata records are explicitly interlinked with each other via jointly referenced entities, e.g. the same author. This also holds true within an individual metadata collection. With traditional string-based metadata management this interlinking stays implicit, although it is an important information about the underlying information resources. This deeper integration provides an important basis for the creation of value added services on top of the metadata as discussed below.

7.1.3 Additional library services

The use of global entity IDs does not only stronger interlink the metadata records with each other. It also enables the integration of the metadata records with other (richer) sources about the considered entity, e.g. RDF knowledge bases, which can again be interlinked via the global IDs. The usefulness of such richer knowledge layers on top of or in place of the metadata management has already been recognized by DL management systems. The systems BRICKS and FEDORA, for example, provide an

RDF-based knowledge layer for the management of their metadata.

Such a knowledge layer can be used to provide additional services to library users that enable them to gain a better understanding of the domain covered by the respective library. This includes services for analyzing the mutual influence of publications and other information artifacts, services for making the development of trends and ideas visible, for analyzing the influence of individuals and groups in the respective community etc. Some DL systems such Daffodil [53] already started providing such services, e.g. computing and displaying bibliographic citation networks. As one of its R&D highlights, the VIKEF[9] European project has implemented a technique called "semantic infusion" [92], which is used to enhance content objects by adding further semantic information, tailored to the user task, and strictly bases on RDF metadata.

Without global IDs, the integration of data from different systems to provide such additional services is however an expensive and error-prone process and the services highly depend on the quality of this integration process. Through the introduction of global IDs, very accurate services - even doing on-demand integration at run time - could be provided across system boundaries, enabling also the seamless high precision integration of different types of resources.

7.2 News and Media Integration

The news and media industry is an ideal scenario for entity-based applications, and indeed it has always been a primary source of content for data mining and named entity recognition tests. Every day, a large amount of content (e.g., newswires, reports, articles, videos, podcasts) is produced which contains a huge number of references to entities such as people, lo-

[9] `http://www.vikef.net`

cations, organizations, and events. Being able to extract this information, making it explicit, classifying and integrating it with information stored in different sources is therefore a very promising domain, and is relevant for critical applications in the area of business intelligence, trend analysis, competitive analysis, homeland security, among others.

To enable and support these applications, the typical metadata about newswires (e.g. date, author, topic, format, etc.) may not be enough. Information about entities, their type, and the relations between them are at least as necessary. However, most content is not structured, and thus the integration requires a fair amount of preprocessing for recognizing named entities, computing co-references, establishing their types, discovering what the relation is between them in complex contexts (something like "High-level talks are currently under way between Alitalia and Air France-KLM in regards to a possible acquisition of the Italian airline by the French-Dutch carrier, inside sources said"[10]).

To improve this situation, substantial effort has been devoted to the creation of standard metadata schemas for news, like for example NewsML [65] and NITF [67]. Interestingly, these schemas typically allow making reference to a controlled vocabulary, which lists the names which are accepted as values for some elements or attributes (see e.g. the notion of `FormalName` in NewsML). This is an example of how this mechanism is used in NewsML:

```
<Property FormalName="Location">
<Property FormalName="Country" Value="IRQ"/>
<Property FormalName="City" Value="BAGDA"/>
</Property>
```

where `Location`, `Country` and `City` are formal names, and `IRQ` and `BAGDA` are allowed values for `Country` and `City`.

[10]See `http://www.ansa.it`, 12th September 2007

This is of course an evidence of the strong need of creating IDs which are not affected by the potential ambiguities of simple free strings. However, solutions like the one we described above are not fully satisfactory for several reasons. First of all, they only cover a limited number of categories and entities, typically well-known people, locations, organizations; very little can be done to extend the vocabulary with names and categories which do not have a global relevance or suddenly gain relevance due to unexpected events. Secondly, this method tends to mix the information that an entity is named in a text with some knowledge about the entity itself, as it pre-categorizes things (for example, the fact that Bagdad is a city, and that a city is a location), but this can be a strong limitation for objects which are not so easily categorizable (for example, how should we categorize an event like "September 11th"?). Third, the schema is used to provide some kind of "header" in the newswire, but the respective content itself is not directly annotated. As a final observation, we notice that the metadata record and the controlled vocabulary are language dependent.

Efforts have been made to design more sophisticated schemas using RDF and OWL ontologies, such as in the NEWS project [81]. However, even this type of approach suffers from the usual problems as far as entities are concerned. Indeed, the conceptual schema is well referenced through the use of URIs, but entities are assigned local URIs which cannot be used to perform an automatic integration with corpora based on the same ontology but developed at a different location or within a different application.

Entity-aware metadata management would support the development of interesting applications in this domain. Some examples are: creation of authoring environments which can use a news archive as background knowledge for the creation of new content (e.g. by providing inline links to past articles in which an entity was already referred to, or a profile of the entity based on pre-existing content); new metaphors for navigating across a news archive (including multimedia content), where links would follow relation between entities and not between documents; more efficient systems for information extraction, which do not focus on named entity recognition but on the extraction of relations across global entities identified through global IDs.

7.3 Entity-centric Search

An Entity-centric Search Engine would provide new and alternative ways to explore information and knowledge spaces, allowing people (and machines) to move from a keyword-based, document-oriented search paradigm to a knowledge-oriented paradigm by taking advantage of the *Web of Entities*. Information about an entity, e.g. the city of Rome (identified by a global identifiers) can be combined together in a search result summary which includes, for example, a list of statements about it (e.g. that Rome is an Italian city, that it is the Italian capital, that has 3.5 million inhabitants, ...), and a list of pointers to available resources (including RDF and OWL knowledge bases, web pages, documents, databases, films, photos, text, audio, etc.) in which this entity is mentioned. In this way an Entity-centric Search Engine would provide a fast and reliable integration of results including unstructured data, by exploiting the *Web of Entities*.

Novel functionalities not possible with current web search engines, increased precision and recall and the possibility to automate search tasks would be available since entities would be unequivocally represented, the disambiguation needs would almost disappear and different representations of the same entity would be integrated without efforts in the results of a query. Moreover, relations between entities could be taken into account in order to provide higher value results to complex queries by integrating and relating data coming from very different sources and making this relations explicit.

Chapter 8

Analyses, Experiments and Results

8.1 Performance Improvement

Our first prototype had a runtime behaviour that ranged in the area of 20 seconds per search request, and up to several minutes if the query string exceeded three keywords. It is obvious that for any sensible use of such an infrastructure for online processing, these values are unacceptable.

An analysis of the situation revealed two main issues. First, the implementation for connecting to the underlying database was (i) not optimized, i.e. one search request would cause a whole number of connections to be opened and closed during runtime, and (ii) it used unpooled JDBC[1] connections which are time-consuming to establish (e.g. because the database backend needs to perform authentication). We resolved these issues by optimizing the code that accesses the database and by using a pooled connection infrastructure as described in Sect. 5.2.2, which provides permanently open connections and thus helps to overcome these performance issues to a great extent.

The issue of long execution times for queries with more keywords could be traced back to the SQL queries generated by our relational database plugin `OkkamJDBCImpl`. It turned out that we had been using an overly

[1] Java Database Connectivity, a standard extension of Java.

complicated way to realize queries over the contents of table `Label` (see Fig. 5.10 on page 66), thus creating a large number of self-joins which slowed down query execution beyond the acceptable limit. Figure 8.1 displays the query plan generated by the DB2 query explanation tool for an example AnnotatedQuery.

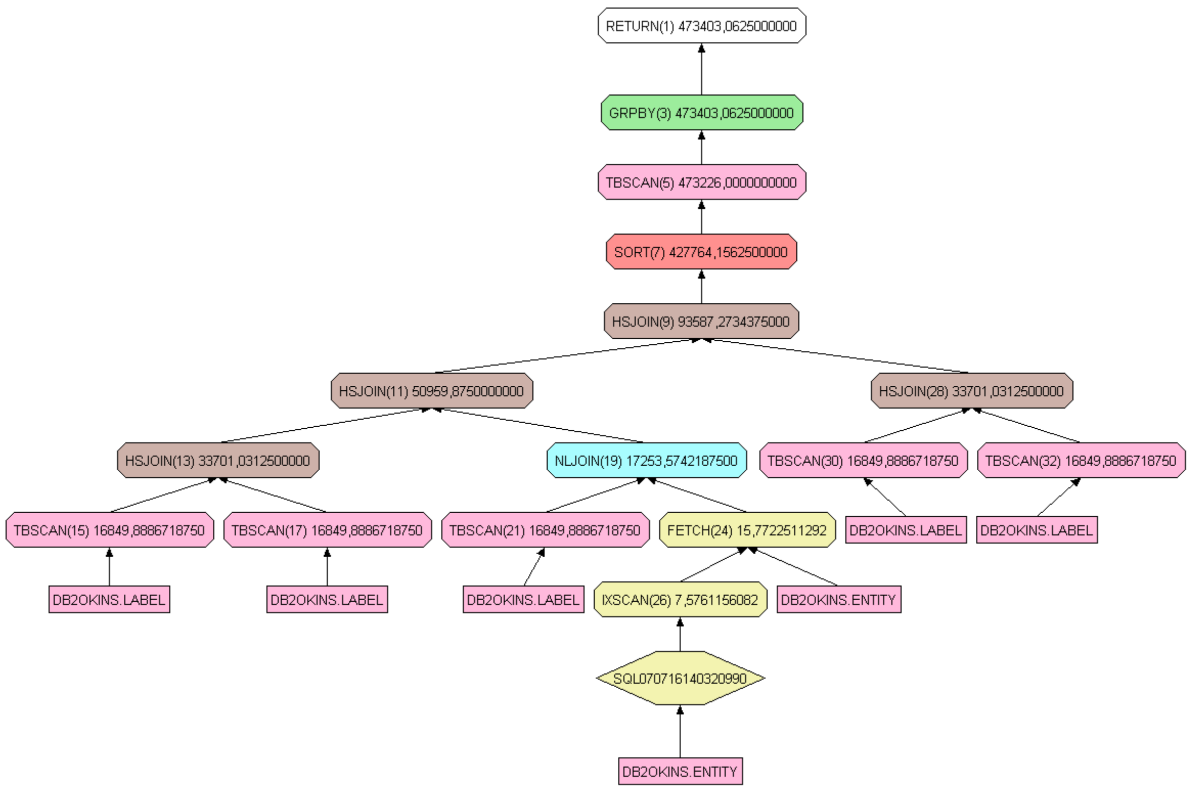

Figure 8.1: Query plan of search query before optimization

After analyzing the queries that were created, we re-implemented the algorithm to move from a model of self-joins to a nested SELECT approach. The query plan for the same AnnotatedQuery with the new approach is depicted in Fig. 8.2 and shows a significantly less complex structure (note especially the drastically decreased amount of table scans).

With this simplification of the generated query, we were able to move – in the example – from a cost of 473.403 timeron units[2] to a cost of 6.010

[2] "A unit of measurement used to give a rough relative estimate of the resources required, or the cost,

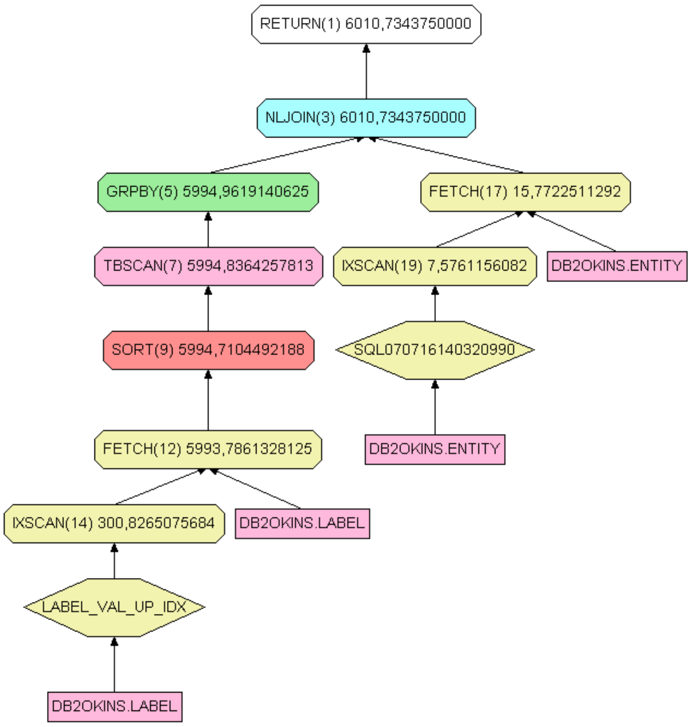

Figure 8.2: Query plan of search query after optimization

timeron units, which means that the query was processed 78 times faster than before.

The outcome of the combination of the two optimizations was measured in the course of the ontology integration experiment described in Sect. 8.3. The execution of 875 different search queries produced an average execution time of 721msec per query. This value includes the complete execution chain of the OKKAM architecture described in Sect. 5 and satisfies our performance requirements for the prototype to the fullest.[3]

for the database server to execute two plans for the same query. The resources calculated in the estimate include weighted processor and I/O costs." (cf. [48]).

[3]It should be mentioned however that the experiments were run on the same machine that also hosts OKKAM, which drastically speeds up the data transfer that is performed by the web services. In a setting where these web services are actually used over the internet, we experience a typical execution time including the data transfers of around 2 seconds.

8.2 Evaluation of Similarity Metrics

The current implementation of the entity matching functionality, motivated and described in Sect. 5.3, relies on a probabilistic approach that computes the similarity between an *entity profile* and a query posed by a client agent. To select a suitable algorithm that can provide such a measure, we have evaluated the 21 similarity metrics described in [23] [4].

First, we created three `AnnotatedQuery` representations (see Sect. 5.2.1) of the following queries[5]:

*q*1: **"Paolo Bouquet Trento"** . This query is supposed to simulate a typical search for an entity of type *person* with a european name and a location specification.

*q*2: **"trento Italy"** . This query is supposed to simulate a typical search for an entity of type location.

*q*3: **"xin liu jlu trento"** . This query is supposed to simulate a typical search for an an entity of type *person* with an asian name, which is a more specific problem as it consists of tokens with only a few characters, and some interference caused by the character sequence "jlu" which is an acronym for a university name[6].

Next, we established a "golden standard" for every query *q*1, *q*2, *q*3, which consists of a ranked list of the top-5 entities sorted in descending order of similarity, that a human evaluator would expect as the perfect answer to the query, based on the dataset stored in OKKAM at the time of

[4]These metrics have been implemented by Sam Chapman in the course of the AKT and Dot.Kom projects, and are available at the following URL: `http://www.dcs.shef.ac.uk/~sam/simmetrics.html`. They have the invaluable characteristic that they produce a normalized measure which makes them directly comparable, as opposed to other implementations available. As the aim of this work is not the creation of a new similarity measure, we decided to base our prototype on this publicly available library.

[5]All spelling variations have been introduced on purpose to perform a more realistic comparison.

[6]JiLin University, Changchun, China.

the experiment. Every entity is issued with a weight w which is supposed to reflect the quality of the result with respect to the query and is calculated as follows:

$$w = m * \frac{1}{n} \tag{8.1}$$

with n being the number of tokens in the query, and m being the number of tokens of the query which also occur in the *EntityProfile* of the entity.

Subsequently, we ran a test which evaluates all queries, using all similarity measures:

```
foreach similarity_measure s:
   use s in OKKAM
   foreach query q:
      resultset = execute(q)
      compare(resultset, golden_standard)
```

The function `execute()` returns the top-10 matches in descending order of similarity, based on a similarity measure s.

The function `compare()` evaluates how close the results are to the respective golden standard. Let M be the set of matches, i.e. the entities from the resultset which are contained in the golden standard, m_k the k-th element of M, and w as established in Eq. 8.1. Then we calculate closeness c as follows:

$$c = \sum_{k=1}^{|M|} w(m_k) * p(m_k) \tag{8.2}$$

with p being the weight of the position of the entity in the resultset (defining $p = \frac{1}{position\ in\ ranking}$, e.g. the top item in the set has $p = 1.0$, the last $p = 0.1$).

The result c in Eq. 8.2 reflects (i) the existance of a match returned from OKKAM and (ii) its distance from its optimal position, with respect to the golden standard. We computed c for each query and each similarity measure, resulting in a matrix.

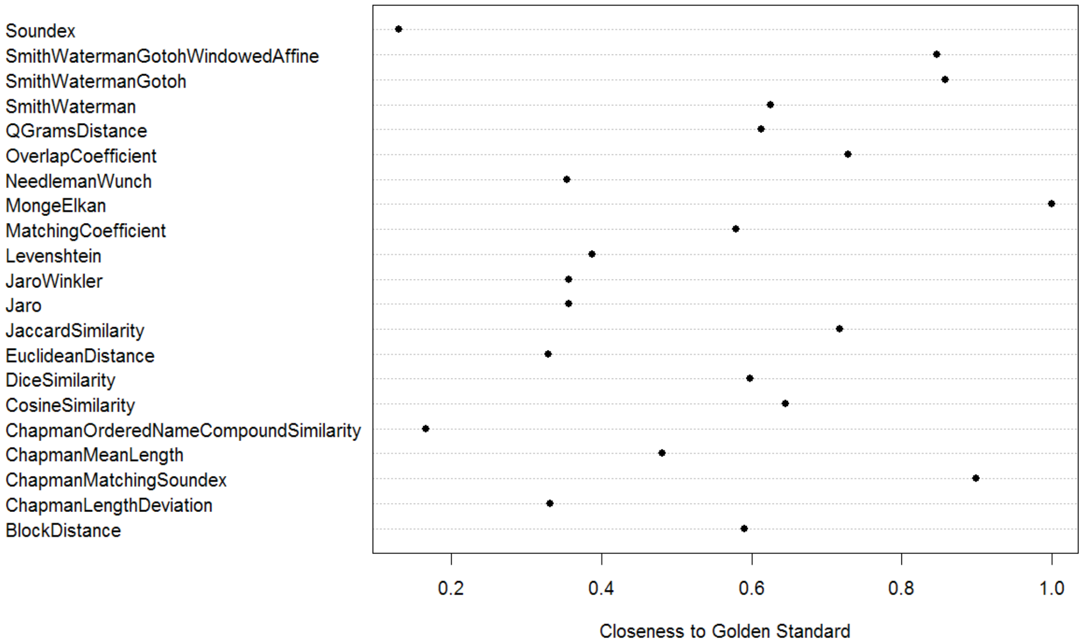

Figure 8.3: Performance of Similarity Measures for Entity Matching in OKKAM.

We report the *average* closeness for every similarity measure, inferred from this matrix, in Fig 8.3. It is evident that with respect to our example queries $q1$-$q3$, the Monge-Elkan algorithm [64] delivers the best results.

As a by-product of this experiment, we also registered the runtimes of the matching process for each similarity measure. Table 8.1 illustrates the normalized[7] runtime behaviour of the Monge-Elkan algorithm in comparison. It is evident that the algorithm has a runtime behaviour that lies roughly ten percent above average, but is significantly lower than the

[7]The table does not report the *absolute* values for the reason that the experiment was performed in a setting that required a time-consuming lookup procedure for a database connection to be performed for every run, which is not the case in a "production" setting, and is thus not representative for the performance of the overall system. The time for this procedure is however constant, which allows us to report relative measures in this case, and to illustrate the general behaviour of the algorithm.

maximum runtime.

Maximum average runtime	1.000
Minimum average runtime	0.310
Average runtime (overall)	0.505
Monge-Elkan average runtime	0.617

Table 8.1: Normalized Runtime Behaviour of Monge-Elkan Algorithm

Consequently, for the combination of good results in the matching tests and an unobtrusive runtime behaviour, we chose this algorithm for the current implementation of the OKKAM prototype and as the base for the experiment described in Sect. 8.3.

8.3 Instance-level Ontology Integration

In this section we will describe with an experiment of instance-level ontology integration that the proposed approach of aligning identifiers of different local data sources against a global repository of identifiers is viable and achievable.

We integrate in an automated way the Semantic Web ontologies of the conferences ISWC2006 and ISWC2007[8]. While this is not a "typical" use-case of OKKAM, as it constitutes an *ex-post* alignment which we do not propagate as best practice, we set up this experiment to test and improve the performance of the current OKKAM prototype. Furthermore, automatic processes such as bulk entity import or on-the-fly entity annotation, which are planned as future work, can base on the findings of this experiment.

The aim is to perform unsupervised entity consolidation on entities of

[8]These ontologies were originally available via `http://data.semanticweb.org`. At the time of this writing, the website has been continuously unavailable; for this reason we make the data available at the following URL: `http://okkam.dit.unitn.it/swonto/`

type *foaf:Person*, to evaluate several aspects of this process, and consequently, to establish a threshold for entity identity on which processes such as automatic alignment can rely. In the following, we evaluate three steps:

1. Establishing a similarity threshold t_{fp}, which can be considered a reasonable value below which a best match found by OKKAM should be considered a false positive.

2. Establishing a golden standard to evaluate the results of the merging process grounded on the threshold t_{fp}.

3. Performing an unsupervised ontology merge and analyzing its results.

8.3.1 Establishing t_{fp}

In OKKAM, deciding whether an entity e matches a query q relies on a similarity threshold t_{fp} below which e should be considered a false positive, i.e. a decision on the basis of a measure of similarity s $(0 \leq s \geq 1)$ between e and q, compared to t_{fp}.

The goal of the first experiment is to fix t_{fp}. To meet this goal, we run the system on a sample of queries corresponding to a list of person entities in the ISWC2006, ESWC2006 and ISWC2007 metadata sets. For each query, the system returns an OKKAM URI and a corresponding similarity s. Subsequently we check by hand the performance of the system, comparing the data attached to the source URI with the profile of the OKKAM URI, to verify the correspondence.

In this way we collect matching examples to test the performance of the system on a suitable range of similarity values. First we group the data in similarity classes ($S = \{s_1, ..., s_j\}$) and for each class we calculate the

S_j	Expert assigns YES	Expert assigns NO
System assigns YES	TP_j	FP_j
System assigns NO	FN_j	TN_j

Table 8.2: Contingency table for evaluation of golden standard

frequency [9] and the number of the correct responses[10]. Subsequently, we evaluate how the the performance of the system changes, by varying the threshold on the range of the similarity classes ($t_1 = s_1, ...t_j = s_j$) and for each class we compute the contingency table (see table 8.2), including the values for True Positive (TP_j), True Negative (TN_j), False Positive (FP_j) and False Negative (FN_j).

Here, TP_j (True Positives with respect to the threshold t_j) is the number of entities correctly identified by the system when the threshold is t_j, TN_j is the number of entities that the system correctly did not identify for threshold t_j, FP_j is the number of the entities that have been incorrectly identified by the system when for threshold t_j, and FN_j is the number of entities that the system incorrectly did not identify.

The first evaluation that we performed was comparing the trend of TP with respect to FP. This analysis is motivated by the goal to find the threshold that provides a minimum of FP but preserves a good level of TP.

In general, if the number of FP is too high we risk on the one hand that the results returned by the system would be polluted by irrelevant information, on the other hand if the same threshold is used to perform the entity-merging, two non-identical entities would be collapsed. The latter is a very undesirable circumstance because it leads to "losing" entities in the target ontology and asserting wrong information to the collapsed entity.

[9]number of entities returned by the system with the value of similarity corresponding to the class

[10]number of entities correctly retrieved by the system with the value of the similarity corresponding to the class

In order to determine an acceptable $TP - FP$ trade-off we adopt a distance measure between TP and FP (the absolute value of the difference between TP and FP, $|TP - FP|$, or Manhattan distance) to establish the value of similarity in respect to which this distance is maximized.

In figure 8.4(a) we plot TP and FP and the absolute value $|TP - FP|$. The graph shows that FP decreases more rapidly compared to TP when the similarity increases and the trend of difference $|TP - FP|$ shows a maximum correspondence level of similarity equal to 0.75. On this level, the system presents $TP = 0.47$ and $FP = 0.10$.

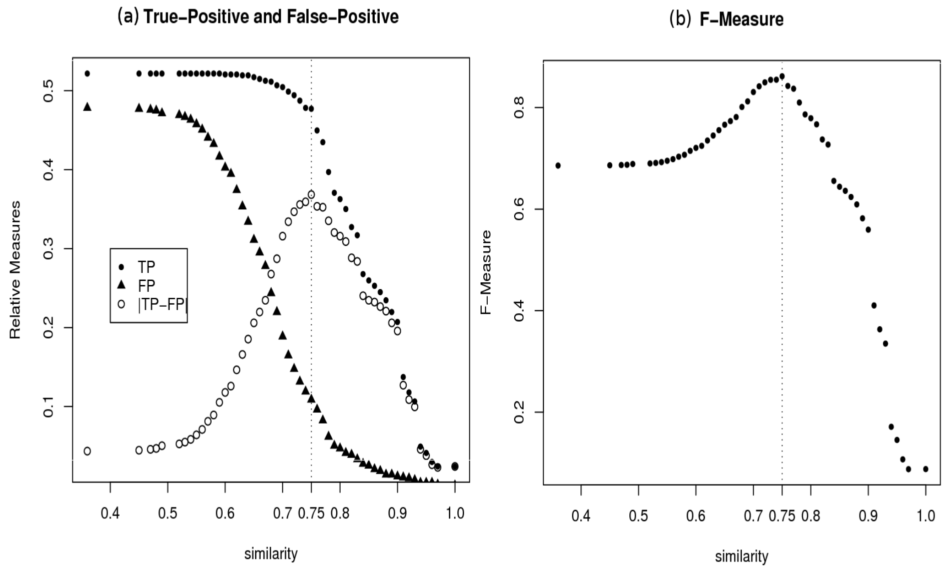

Figure 8.4: Evaluation results of the ontology merge.

In order to confirm our result, we evaluated the performance of the system, measuring its effectiveness by means of Precision (P), Recall (R) and F-Measure (F)[11].

[11]See `http://en.wikipedia.org/wiki/F-measure` for an overview of these performance measures from the field of Information Retrieval.

Precision is the percentage of entities correctly matched by the system with respect to all entities matched by the system:

$$P = \frac{TP}{(TP + FP)}$$

Recall is the percentage of entities correctly matched by the system with respect to all entities matched

$t_{fp} = 0.75$	TP	TN	FP	FN	P	R	FM
	0.47	0.36	0.10	0.04	0.81	0.91	0.86

Table 8.3: Performance of entity matching for $t_{fp} = 0.75$

For each similarity class, we calculate these evaluation measures to find out which similarity value ensures the best performance of the system. We present the results relative to the F-Measure, which gives a good overall description of the performance.

In figure 8.4(b) we show how the F-Measure varies as a function of similarity. We can see that the F-Measure increases up to a similarity of 0.75, and then dicreases rapidly. This evidence confirms the result of the first analysis, indicating the best threshold $t_{fp} = 0.75$. On this level we register a value of F-Measure equal to 0.86, corresponding to $P = 0.81$ and $R = 0.91$. Table 8.3 summarizes the performance of the system when the threshold is $t_{fp} = 0.75$.

8.3.2 Evaluating the Ontology Merge

In order to evaluate the performance of the system with respect to the results of the merging process, we have to define a benchmark that we consider as the gold standard in our analysis.

For this purpose we took into account two (ISWC2006 and ISWC2007) out of three Semantic Web ontologies considered in the first phase of our

by the human:

$$R = \frac{TP}{(TP + FN)}$$

F-measure combines in a single measure Precision (P) and Recall (R) giving a global estimation of the performance of the system. In this case we assign the same weight ($\alpha = 0.5$) to P and R, as a consequence:

$$F = \frac{2 * PR}{(R + P)}$$

	$t_{fp} = 0.75$	$t_{fp} = 0.90$	$t_{fp} = 0.91$	Gold standard
Total Positives	70	68	43	48
True Positive	46	45	25	48
True Negative	380	385	405	403
False Positive	24	20	20	0
False Negative	1	1	1	0
Precision	0.66	0.69	0.56	1
Recall	0.98	0.98	0.96	1
F-Measure	0.78	0.81	0.7	1

Table 8.4: Results of the merging process

evaluation analysis. The choice to reducing the size of the sample was motivated by the procedure of determing the gold standard that involved a manual check of the entries of the data sets, which needed to stay at a manageable size.

At first we compared manually the two data sets to detect which URLs in the first refer to the same entities (persons) to which point the URLs in the second, and vice versa. This comparison returns the exact set of pairs $G\{< x_i, y_j >\}$ with x_i being a URI from the first ontology and y_j a URI from the second, which an optimal system would have to detect. This set G with $|G| = 49$ represents the gold standard of our analysis.

In the second step of the analysis we perform an *automatic* merge of the same data sets (ISWC2006 and ISWC2007), and compare it to the gold standard. In the table 8.4 we report the results of our analysis with respect to three exemplary thresholds t_{fp} that we examined.

If we consider the first column of the table in wich we have the results respect to a value of $t_{fp} = 0.75$, we can notice that the correct mappings amount to 46. If we compare this value to the gold standard $g = 48$ we can see that the system returnsalmost all the correct mappings. However, the number of False Positive is still quite high and it reduces precision to

$P = 0.65$. In other words, the system recognises some mappings between entities that in fact are not identical. The effect of this circumstance is that the merging of these entities leads to losing entities in the joint ontology, and to assert wrong information about the collapsed entity. Obviously this is what we want avoid, and it induces to search another (more conservative) threshold that garantees a lower number of FP, preserving a satisfying number of TP.

8.3.3 Establishing t_{id}

In the second column of table 8.4 we report the evaluation measures with respect to $t_{fp} = 0.90$. On this threshold, Precision improves because FP decreases (from 24 to 20) but without sacrificing significantly TP. Finally, if we consider $t_{fp} = 0.91$, we can notice that the performance of the system degenerates. Indeed, TP decreases to 25 while FP does not reduce. We take this as evidence that in the current setting it is recommendable to increase t_{fp} in the merging process to a value near to 0.90 to find a good trade-off between sacrificing TP and reducing FP.

Our experiment showed that it is possible to move from two data sources that should set-theoretically present a certain overlap but syntactically do not[12], to a situation where good recall of matches can be reached through an alignment against Okkam. The approach presented here requires no ad-hoc implementations or knowledge about the representation of the entities, as opposed to other approaches, such as the ones described in Sect.3.3, or e.g. [46] which relies on special characteristics of a property in the RDF description of a FOAF profile[13] for establishing identity between entities.

Summing up, we have shown that with the help of Okkam it is possible

[12] in fact, the two data sources present an overlap of zero identifiers for person entities

[13] it relies on the inverse-functional property of the the `foaf:mbox_sha1sum` property in a FOAF profile which is used to relate a hash-code generated from a person's email address to its "entity identifier". See also Sect. 6.2.

to perform entity consolidation over heterogeneous data sources with a reasonable outcome. It has however to be stated clearly that establishing a threshold that will serve as the key parameter for identity decisions should focus on producing a minimal number of false positives. We conjecture that a straighforward set of heuristics that respect the *type* of entity under question can improve the results presented here considerably. Such source-independent[14] heuristics can include the variations in spelling of peoples' names, the matching for location names in a different major languages, or a filtering on more and less relevant parameters in the entity profile to be used for the matching.

8.4 Entity-level Metadata Integration: A Cost Analysis

When talking about metadata integration, which includes RDF metadata, we think about the integration of an additional data set into an existing one (e.g. the import of the metadata about conference proceedings into a digital library). The realization of such a scenario in which the metadata are aligned on the entity level ("okkamized") may involve costs which may not appear as completely justified on first sight. To discuss this objection, we present an analysis of integrating (non entity-aware) metadata records from different sources, and show that the usage of global identifiers significantly reduces this cost.

The integration process we envisage is as follows. Assume we have a local "integrated" representation which contains the okkamized metadata that our approach has already processed. We now want to add a new metadata source, which contains records that all follow the same schema.

[14] i.e. they do not rely on the presence of a special datum in the entity profile derived from a special data source.

Each record from the new data source has to be integrated and added to the local "integrated" representation. We assume that the addition to the local representation has a fixed cost (insertion cost as a constant), but the integration cost varies. The integration of a new record into an existing integrated record can be seen as a series of integrations between the corresponding attribute-values pairs contained in the two records. How this operation is performed depends on the actual system, the data, and the strategy. We assume the use one of the various exiting approaches for this procedure (presented in Section 3.2), and thus we consider it as a black-box.

It is quite intuitive to see that in scenarios using global identifiers for entities the cost of integrating several independent sources at query time (e.g., in federative search environments) is less than in cases where global identifiers are not used. Imagine a dataset constructed from independent sources. An approach not using global identifiers would need to perform a matching between content objects every time a query is posted, but if all sources would share a global identifier for the same referenced entity, then the integration task becomes less complex. Additionally, having all sources using the global identifiers for each entity diminishes the cost of adding new sources to the federation.

Cost of integration without global identifiers. We will now compute the cost for integrating query results without using global identifiers for entities. Let us use symbols r_1 and r_2 to denote metadata records. The i-th attribute value pair of r is denoted with $avp_i(r)$ where the value represents the symbolic name of an *entity* (as opposed to e.g. a date). The total number of *avp*'s in the metadata record r is given by $size(r)$. The cost for finding a match of an *avp* in a target metadata record r is given by $amc(avp, r)$, and the cost of representing *avp*'s in locally integrated metadata records is given by lrc. Then, the cost of integrating metadata record r_1 into a locally represented record r_2, $ric(r_1, r_2)$, is given by:

$$ric(r_1, r_2) = size(r_1) * lrc + \sum_{k=1}^{size(r_1)} amc(avp_k(r_1), r_2) \tag{8.3}$$

For adding the record r into a local integrated collection of records L, we must integrate it with all (already locally integrated) metadata records $r_i^L \in L$. The cost of this operation is:

$$lsic(r, L) = \sum_{j=1}^{|L|} ric(r, r_j^L) \tag{8.4}$$

Consider now having to integrate a set of metadata records M. The cost of integrating all records of M into L is:

$$ic(M, L) = \sum_{i=1}^{|M|} lsic(r_i^M, L) \tag{8.5}$$

Cost of integrating with global IDs. In contrast to the previous paragraph, we will now compute the costs for integrating records that use global identifiers for the entities.

The cost of integrating a new metadata record r_1 with an already integrated metadata record r_2 is as follows:

$$oric(r_1, r_2) = size(r_1) * lrc \tag{8.6}$$

The difference with Equation 8.3 is that we here consider that the involved metadata records already have global identifiers in their *avp*.

For computing the integration cost of a new result by using sources with global identifiers, Equation 8.5 still holds as defined, if we replace $ric(r_1, r_2)$ with $oric(r_1, r_2)$ (from Equation 8.6) in Equation 8.4.

If we take into account the network externality effect by considering the probability p of encountering a metadata record using global IDs (and with probability $1 - p$ of encountering a metadata record without global IDs) we can compute the cost as presented in Equation 8.7:

$$ic(M, L) = \sum_{i=1}^{|M|} \sum_{j=1}^{|L|} \left(size(r_i^M) * lrc + \underbrace{(1-p) * \sum_{k=1}^{size(r_i^M)} amc(avp_k(r_i^M), r_j^L)}_{matching\ cost} \right) \quad (8.7)$$

From the computed expression in Equation 8.7 we can see that the cost of integration depends on: (1) the size of the records, and (2) the cost of representing in the local format a new metadata record. Additionally, we can see that the cost of integrating metadata records strongly depends on the method used for performing the *avp* matching (illustrated as *matching cost* in Equation 8.7).

The analysis of Equation 8.7 shows that if a high number of sources (re-)use global identifiers, the higher costs computed in the *matching cost* part of Equation 8.3 can be disregarded, and the overall integration cost becomes a linear function of the number and size of the records to be integrated.

What is not calculated in the above formulae is the cost of actually *issuing* metadata records with global identifiers instead of names for entities. It can be argued that this leads to faulty calculations about the benefits of the approach, as part of the cost are "ignored". However, the reasons for not including this factor into the calculations are the following: first of all, the cost of issuing a metadata record with global identifiers depends on several factors in itself, e.g. the question whether this process is performed manually, or automatically on the side of the metadata creator, or by a centralized public service, and can thus not be generically quantified.

More importantly, the approach proposed in this work partly relies on the economic principle of *network externality*, which can be defined as "a change in the benefit, or surplus, that an agent derives from a good when the number of other agents consuming the same kind of good changes" (cf. [59]). This means that the participation of other agents positively influences the situation: as soon as an approach of this kind is taken up by others, the benefits analyzed in Equation 8.7 hold.

Take for example the introduction and utilization of common data formats such as EDI[15], or XML. Both required (considerable) investments during adoption; however, the expected network effects were high, with the effect that EDI and XML became widely-used standards for data exchange. Everyone who today is using one of these formats is benefitting from the investment of others, and at the same time also contributes to the benefit of others, which is exactly the type of situation which we tried to formalize analytically in this section.

8.5 Rigid Designation and its Consequences

The compelling vision of the Semantic Web is to provide languages and tools that allow us to build something that can be described as a huge, global, distributed knowledge base: sets of statements about individuals (A-Boxes in Description Logics), potentially stored in different locations, use vocabularies from formalizations (T-Boxes), which are again distributed, and integratable via an import mechanism. The common denominator and pivot are identifiers for resources in these T- and A-Boxes: URIs. A client agent should consequently be enabled to profit from this distributed knowledge, and an answer to a query about a resource should – optimally – consider "all that is known" about that resource.

When analyzing this vision of a global, distributed knowledge base, it

[15]Electronic Data Interchange, see also `http://en.wikipedia.org/wiki/Electronic_Data_Interchange` for an informal description.

is imaginable to defend two opposite standpoints:

1. It is necessary to introduce *rigid designators* for resources: an identifier that denotes the exact same object, wherever and whenever it occurs. This, applied to classes and individuals alike, trivially results in a set of globally pre-aligned A- and T-Boxes that only need to be syntactically merged in order to answer queries.

2. Due to massive heterogeneity, the vision must fail. The past has shown that a complete agreement on the concept level cannot be achieved[16], and that an agreement on URIs for individuals is unrealistic[17].

We believe that the truth lies somewhere in the middle. On the one hand, we accept the claim that a global alignment on the concept level, by acceptance of top-level ontologies such as CYC [57] or SUMO [66], or any other such agreement, has not been proven to be viable or even feasible – a main reason for this being the differences in *intended meaning* that we relate to concept names, as discussed in Sect. 2.1. A way to overcome this problem is the introduction of ways to explicitly establish mappings between concepts, as pursued in the last years e.g. by the efforts around Distributed Description Logics (DDL) [11, 89].

On the other hand however, we defend the previously introduced point that what is commonly regarded as an "individual" in a Semantic Web ontology underlies such differences to a vastly lesser extent: the heterogeneity from the concept level does not necessarily apply do individuals. Typical types of objects, such as people, events or locations, are rather unique by character, and it should thus be possible to uniquely and *rigidly* identify these objects, also in a distributed fashion.

[16]See e.g. [10] for motivations; another strong indicator for this claim are the vast efforts in schema and ontology matching, which have lead to a large number of approaches, described in Euzenat and Shvaiko's book on Ontology Matching [33], which counts an estimated number of over 300 bibliographic references.

[17]as claimed by the authors of the OWL Recommendation, in Sect. 5.2.1 of [5]

From a formal point of view, to capture the full implications of the described vision, we propose a hybrid approach: a logics that allows for heterogeneous conceptualizations with mappings between concepts, and the global and rigid interpretation identifiers for individuals. The outcome is a "system" with capabilities that lie between the extreme positions described above. On the one hand, less integration is possible than in the first standpoint, because the concept mapping cannot be assumed to be complete. On the other hand, it is less fatal than the second standpoint, because concept mappings are possible and an agreement on individuals is potentially far-reaching.

To give a more formal account of our notion of rigid designation, let us consider Distributed First-Order Logics as used by Serafini et al. in [77] [18]. They offer a definition for rigid designation in heterogeneous domains as follows: if *every* constant c of two languages that forms part of a pairwise domain relation between the respective domains is interpreted in two corresponding objects, then rigid designation between these domains is given. Note that Serafini et al. are interested in rigid designation as a property of an *entire* Distributed First Order Logics model, for which reason they investigate the interpretation of *every* constant; note also that they only consider constants that form part of a domain relation, because their underlying theory of Local Models Semantics [40] defines individual domains of interpretation, which means that they need a relation between objects of the domains to be established before they can decide about rigid designation.

The existance of ENS-issued identifiers for entities is a slightly different case as the one previously described. First, we define rigid designation:

[18] The relevance of these works for the Semantic Web lies in the deep relation of FOL and Description Logics which has lead to the development of DDL [11] and C-OWL [13]. DFOL has been used in [77] as a formalizm to analyze and describe ontology mapping languages.

Definition 8.1 *For any two RDF graphs g and h, any URI c with $c \in g \wedge c \in h$ is called a* rigid designator *if c is interpreted into the exact same object.*

What we propose is the following:

Definition 8.2 *Every URI c_{ENS} issued by the* OKKAM *ENS is a* rigid designator.

Note that we are offering a different standpoint with regards to domains than DFOL does. In fact, what the ENS achieves is rather to provide the possibility to have a single domain for all graphs/ontologies, being the set of all URIs that the ENS issues. This means that for languages that use our identifiers, no domain relation between objects needs to be present.

From the viewpoint of DFOL, we can express this property as follows: let $i \neq j$ be two OWL ontologies and x be an individual as defined by the OWL semantics [72], then

$$i : (x = c_{ENS}) \rightarrow j : (x^{i\rightarrow} = c_{ENS}) \tag{8.8}$$

$$i : (x^{j\rightarrow} = c_{ENS}) \rightarrow j : (x = c_{ENS}) \tag{8.9}$$

A common criticism of ontology mapping approaches is the question who is supposed to *establish* these mappings. It is straight-forward and desirable in a heterogeneous world such as the Semantic Web to have means to express relations between concepts of different formalizations, and to provide the theoretical and practical mechanisms to reason over them. However, it seems that getting to a point where a system (or user) is actually able to make use of these foundations, i.e. to know how elements in different ontologies are related, is usually considered "a different problem".

We are not going to claim that the existance of rigid designators in the Semantic Web is going to revolutionize ontology mapping approaches.

On first sight, one might be tempted to reason that an analysis of the extensions of two classes (e.g. the discovery that the two classes have an identical extension), might allow us to perform inductive reasoning about the relationship of the classes (in this case that the classes are identical). In fact, there are ontology matching approaches such as [20] which precisely perform such a process to support their matching. However, induction itself has been object of much controversy as to whether it actually produces acceptable inferences. We hold the position that while a mere extensional analysis is probably not strong enough to decide about relations between classes, the existance of rigid designators can ease considerably the decision to which extent class extensions overlap, as it is directly evident from comparing the sets of identifiers, instead of having to compute the similarity between individuals as it is done e.g. in [20]. And for this reason, we believe that rigid designation for entities can make a noticable contribution to approaches that use instance matching as part of their decision process about concept mappings.

Chapter 9

The Future

9.1 Research Challenges

9.1.1 Large-scale Repository Management and Evolution

In a larger, more production-quality setting, the architecture described in the previous chapters requires an efficient storage and access mechanism for a huge collection of entities. Furthermore, the storage has to support the complete lifecycle of the content.

Data Storage and Access

The standard approach of handling large data collections are all kinds of data management systems from relational to object-oriented database systems. While stand-alone database servers can handle an amount in the area of several 10 million entities, there exist distributed, parallel and in-memory solutions which can handle bigger data collections or process similar data amounts faster.

The challenge is to provide fast access to the content without requiring a strict, explicit database schema. Relational database solutions dependent on a fixed, well-known[1] and understood schema, thus sufficient indices can

[1] well-known by software layers that access the database

be created supporting the query execution. In case of attribute-value pairs as a semi-structured description of entities as currently applied in OKKAM, the evaluation of queries containing a selection of several attributes results in several joins which only scale in case of small table sizes or a high discrimination of the attributes with regard to the corresponding entities.

An alternative is the integration of the technology of existing Internet search engines. They are managing a vast amount of data and enable access to the content ordered by a metric. As described in [22], the underlying infrastructure is highly optimized for this kind of search and is less general than a relational database system, as it does not make assumptions about the types of "things" that are described.

However, the search provided by these engines is generally text-based search as used in information retrieval. An approach could be to use information retrieval methods as an index structure to store different combinations of attributes per entity, for example using an n-gram based approach consisting of combinations of attributes. Such an approach is costly with regard to storage. However, the extension of such an approach from sets of attributes to sets of attribute-value pairs is not straight forward.

Another alternative approach is a database system supporting semi-structured content like for example XML databases. Here, the queries can be translated into a combination of path queries. However, joins will also only scale in case of good discrimination of XML elements. A related approach is semi-structured data storage recently evolving from Semantic Web projects. An example of a storage supporting quadruples consisting of a subject, a relation, an object, and a context is the clustered semantic web search engine [47], or the Swoogle Semantic Web Search Engine [28].

The different approaches mentioned so far provide different properties with regard to transactions. We expect that the main workload of a production OKKAM storage will be on querying, followed by much less insert operations of entities. The requirement of updates and deletions of attribute-value pairs is however hard to predict and has to be further investigated.

Based on the current understanding of the requirements especially with regard to updates, the OKKAM storage has weaker constraints than relational databases. As a consequence, to support the answering of top-k queries (see Sect. 3.2), a pre-ordering of the content within an index structure seems imaginable and should be provided by the primary storage. The challenge here is to provide the pre-ordering to enable fast query processing and on the other hand a guarantee on the quality of the result. From recent experience in information retrieval it seems that further optimization is needed to make the above discussed approach scalable. Further investigation in the advantages of using user query heuristics to organize index structures [80] and the effect of the entity and attribute-value pair life cycle on the index structures during the operation of the system should be performend.

Scalability of Storage

Scalability of storage size and query performance can be addressed in different ways: on the one hand side there are distributed systems maintained at a single organization with optimized communication mechanisms between the distributed storages like for example cluster or parallel database systems. On the other hand side there are more or less decentralized and self-organizing peer-to-peer storage systems which provide scalable storage dependent on the number of contributing peers.

The OKKAM storage should combine both options, since the access to a limited amount of data has to be fast, the communication between the distributed storages has to be optimized and therefore a communication over the Internet does not fulfill this requirement. However, OKKAM users want to have access to all attribute-value pairs stored in the OKKAM storage and therefore a decentralized secondary storage with higher response times will be capable to handle the data volume.

Another aspect of scalability is the load and the availability of the OKKAM storage. A potential approach is follow the well-known principle of data replication - mainly on the primary storage. Access to the replicated installations can be accomplished using a load balancing up-front. Based on state-of-the-art replication and load balancing mechanisms, it is possible that a more advanced OKKAM storage might advance the state of the art by combining primary and secondary storage as well as replication.

Lifecycle

A repository such as OKKAM faces the challenge of effectively and efficiently managing entities within the repository, in the mid- and long-term perspective. It has to ensure sustainable and scalable entity and identifier management that can adapt, react to evolution, and learn from usage patterns and incrementally acquired knowledge about entities.

A well-managed *entity lifecycle* is in the core of sustainable entity management. The lifecycle starts at the time an entity first enters in the OKKAM repository. At this point, decisions have to be made as to whether the entity is already stored (identity decision), and what information to store in the entity repository about the entity. The entity lifecycle and identity decisions have to be based upon a well-founded understanding of entity and entity identity. Subsequently, entities are accessed and additional information about an entity becomes available, e.g. as a consequence of OKKAMization processes. This might require or result in the revisiting of entities and revision of identity decisions, e.g. as a consequence of repository purging processes run over the repository. This includes:

Entity and Repository Evolution. As we have stated previously, it is not the idea to store *all* the information that can be collected about one entity in the OKKAM repository. This would result in a global knowledge base or information repository for entities, which is neither feasible nor desirable. The OKKAM entity evolution model will include the aging of rarely used information and forms of "forgetting" information which is not or no longer of core relevance or distinctive for deciding about the identity of entities.

Foundations of Entity Identity. In addition to what has been described in this work, entity lifecycle and its adequate management should also deal with the foundations of entity identity, treating questions such as when are two entities the same, what is the influence of entity update on the identity, are their inherently identifying attributes available (database keys, artificial identifiers), and how are relationships like aggregation related to the identity of the components and the composed entities. Here more theoretical background work is required to establish a well-defined foundation, on which the OKKAM entity and identity life cycle can base.

Repository Purging and Revisiting of Identity Decisions. Identity decisions, i.e. the decision whether two entity profiles describe the same entity, are performed based on the information in the entity profiles plus eventual background information. However, evolution might necessitate that identity decisions have to revisited. For both cases, strategies and methods have to be developed with the goal of improving the quality of the repository, while, at the same time, minimizing the negative effects on applications that already use the assigned identifiers. In addition to revisiting identity decisions, purging will include the cleaning and merging of attribute values and further methods for quality improvements.

9.1.2 Models of Security, Privacy and Trust

Decentralized systems frequently contain objects with heterogeneous security and privacy requirements that pose important challenges on the underlying security mechanisms. In the simplest case, the distributed nature of these systems introduce the need for secure communication between the nodes, but most frequently distribution and decentralization is used as a means to manage very large systems which means that the problems of user identification and authorization (which in small systems are easy to solve) become much more complicated. Additional complexity is introduced by the fact that distributed approaches tend to be used for open systems (where everyone is a potential user) as opposed to closed systems (where the set of users is mostly static). Finally, the decentralized nature of these systems, with the inherent lack of trust in the different nodes, introduces the need to provide means to guarantee that the information is stored, protected and processed consistently independently from its actual location.

The OKKAM architecture falls precisely in this category. In fact, we could say that it is a paradigm of distributed, heterogeneous and large-scale system with highly dynamic security requirements, a large number of users and very crucial security and privacy requirements. In particular, the need for the OKKAM repositories to be very open; the vast variety of applications that can be supported by OKKAM services; and the nature of the information stored in the OKKAM repositories, together with the possibilities for misuse that the existence of such repositories create, will require the design and development of very flexible security mechanisms [100].

In the area of security and identity different research projects and initiatives have provided advances that will be useful for OKKAM. Among the vast number of such initiatives we can highlight FIDIS, PRIME, SWAMI and PRIDIS on a European level. Also relevant are the Australian National Identity Security Strategy, the US Personal Identity Verification, Shibboleth and Real ID, and some industry initiatives like Liberty Alliance, CardSpace, OpenTC and OpenID. All these projects approach identity from the point of view of authentication, which constitutes the main difference between them and Okkam.

Some of the new scenarios where distributed systems are emerging share some common problems. The most remarkable ones are the following. Firstly, it is usual that objects are accessed by previously unknown users. Therefore, subscription-based schemes are not appropriate in this case. Secondly, the execution of copyright agreements, payment or other activities must be bound to the access to the objects [55]. Finally, the originator or owner of the object must retain control over it regardless of its physical location and even after it is accessed by users [62]. Other requirements are: (i) that a high degree of flexibility is required because of the heterogeneous nature of the objects, (ii) that being able to change the access control parameters dynamically and transparently is also essential and, (iii) due to the large amount of objects, it is important to be able to establish access conditions in an automatic way based on information about objects.

One of the main pillars for these security mechanisms is access control (supported by identification and authorization). Paradoxically, access control in distributed systems often relies on centralized security administration. Centralized control has obvious but important disadvantages: (i) the control point represents a weak spot for security attacks and fault tolerance, (ii) it reduces system performance because it introduces a bottleneck for request handling, and (iii) it usually enforces homogeneous access control schemes that do not fit naturally in heterogeneous user groups and organizations.

Role based access control (RBAC) is commonly accepted as the most appropriate paradigm for the implementation of access control in complex scenarios. RBAC can be considered a mature and flexible. Numerous authors have discussed the access properties and have presented different languages and systems that apply this paradigm. Commercial implementations exist based on RBAC schemes. The main problem with role based access control is that the mechanisms are built on three predefined concepts: "user", "role" and "group" [60]. The definition of roles and the grouping of users can facilitate management, especially in corporation information systems, because roles and groups fit naturally in the organizational structures of the companies. However, when applied to some new and more general access control scenarios, these concepts are somewhat artificial.

In current access control models, the structure of groups is defined by the security administrator and it is usually static. Although the grouping of users can suffice in many different situations, it is not flexible enough to cope with the requirements of more dynamic and open systems where the structure of groups can not be anticipated by the administrators of the access control system. In these scenarios new resources are incorporated to the system continuously and each resource may possibly need a different group structure and access control policy. Furthermore, the policy for a given resource may change frequently.

We believe that a more general approach is needed in order to be used in these new environments and in particular in Okkam. For example, in the referred situations, groups are an artificial substitute for a more general tool: the attribute. In fact, groups are usually defined based on the values of some specific attributes (employer, position, access level, etc). Some attributes are even built into most of the access control models [98]. Similarly is the case of the user element; the identity is just one of the most useful attributes, but it is not necessary in all scenarios and, therefore, it should not be a built-in component of a general model.

Finally, in distributed computing environments, there are many different situations where it is desirable that the owner of each resource is able to retain the control over it and to change the access policy dynamically and transparently regardless of the location where the resource is stored. This property is called originator-retained-control [61]. A full-fledged version of the OKKAM architecture should pay especial attention to this property, as it is relevant for some future scenarios that can be imagined. Additionally, because the creation and maintenance of access control policies is a difficult and error prone activity, the ability to automatically validate policies will become important [99].

9.2 Expected Impact

OKKAM is providing a global infrastructure that enables the creation of the Web of Entities. One of the main features of the approach is that it has a big potential for triggering new ideas and developments for innovative and productivity-enhancing entity-centric services for the knowledge society.

A wide variety of applications can benefit from the creation of the Web of Entities as a global space of identifiers in a global knowledge space. Some examples of such applications are:

- Publishing and Media (e.g. creating a global space of multimedia resources, authors, organizations, topics; entity-level integration of digital libraries; supporting intelligent multimedia authoring environments; helping the production of intelligent content);
- Research and innovation (from document-oriented to knowledge-oriented search for information about relevant entities, like people, organizations, products, publications);
- National and European public administration and government (e.g.

enabling unambiguous reference to laws, institutions, regulations, people, events, etc.);

- Healthcare (e.g. tracking patients and health centers across the continent, making unambiguous references to drugs and pharmaceutical companies, making treatments comparable);

- Financial control (e.g. making possible the aggregation a large amount of heterogeneous data about the same individual across different countries and organizations);

- Homeland security (mining entity-level integrated information sources about people, organizations, events, purchases, money transfers, etc.).

The Okkam technology thus has the potential to drive and stimulate product, service, and process innovation in a large number of areas and in wide domains.

The entity-centric approach, as it is fostered by Okkam, supports different forms of condensation and consolidation of organizational as well as of the global information and knowledge spaces, by reducing redundancies, linking together things that are related with each other, although they are created and managed autonomously, and by entity-level information integration. This eases many tasks along the value chain for digital resources like content creation by combining existing content, information structuring and search across the borders of individual collections, targeted and personalized distribution as well as content re-use, since these operations can rely on a consolidated global Web of Entities instead of just on a partially linked web of documents.

9.3 Future Work

Apart from the open challenges for providing a really complete ENS infrastructure and the application scenarios in which OKKAM may play an important role, the are of course more immediate potentials for improvement in the current prototype implementation.

We see two major issues which do not concern the "usual suspects" of software engineering such as code cleanups, better documentation, more extensive testing, usability improvements, etc.

First, as a promising next step in terms of research, we believe that substantial work can and should be done on the idea of an adaptive matching approach that implements some or all matching facets we proposed in Sect. 5.3.1, and provides intelligent algorithms for a runtime combination of these facets depending on the input query that is posed to the system. Starting with rather simple facets such as creating spelling variations for people's names with the help or algorithms from the area of name matching, we think that the precision of matching results can already be increased. Providing a mechanism that decides about which facets to apply under a certain condition can provide interesting research results in terms of algorithms and experiments.

Finally, after taking a look back at the big picture, we come to the conclusion that the bounds between physical/persistance layer (database) and a logical layer (entity matching and ranking) are probably stronger than we expected. The capabilities of a matching approach are hard to separate from the low-level query capabilities of a persistance layer. We tried to overcome this problem with a separation of steps, by first retrieving candidates from the database and ranking them afterwards. But this approach has its limitations because there are many things that can be envisioned in terms of matching facets which are very hard to translate into an SQL query that can be executed with a sensible performance. In this respect, we believe that either a respectable research effort should be paid to improve the query translation mechanism as it is today, or – as hinted at in Sect. 9.1.1 – the architecture might have to be changed in favour of a whole new backend for entity storage and persistance that directly supports the main requirements of intelligent matching.

Bibliography

[1] Boanerges Aleman-Meza, Meenakshi Nagarajan, Cartic Ramakrishnan, Li Ding, Pranam Kolari, Amit P. Sheth, I. Budak Arpinar, Anupam Joshi, and Tim Finin. Semantic Analytics on Social Networks: Experiences in Addressing the Problem of Conflict of Interest Detection. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 407–416, New York, NY, USA, 2006. ACM Press.

[2] Rohit Ananthakrishna, Surajit Chaudhuri, and Venkatesh Ganti. Eliminating Fuzzy Duplicates in Data Warehouses. In *VLDB*, pages 586–597, 2002.

[3] Periklis Andritsos, Ariel Fuxman, and Renée J. Miller. Clean answers over dirty databases: A probabilistic approach. In *Proceedings of the 22nd International Conference on Data Engineering, ICDE 2006, 3-8 April 2006, Atlanta, GA, USA*, page 30, 2006.

[4] Wolf-Tilo Balke, Ulrich Güntzer, and Jason Xin Zheng. Efficient Distributed Skylining for Web Information Systems. In Elisa Bertino, Stavros Christodoulakis, Dimitris Plexousakis, Vassilis Christophides, Manolis Koubarakis, Klemens Böhm, and Elena Ferrari, editors, *EDBT*, volume 2992 of *Lecture Notes in Computer Science*, pages 256–273. Springer, 2004.

[5] Sean Bechhofer, Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein. *OWL Web Ontology Language Reference - W3C Recommendation*, February 2004. `http://www.w3.org/TR/owl-ref/`.

[6] Omar Benjelloun, Hector Garcia-Molina, Heng Gong, Hideki Kawai, Tait Eliott Larson, David Menestrina, and Sutthipong Thavisomboon. D-Swoosh: A Family of Algorithms for Generic, Distributed Entity Resolution. In *27th IEEE International Conference on Distributed Computing Systems (ICDCS 2007), June 25-29, 2007, Toronto, Ontario, Canada*, page 37. IEEE Computer Society, 2007.

[7] Omar Benjelloun, Hector Garcia-Molina, David Menestrina, Qi Su, Steven E. Whang, Jennifer Widomr, and Jeff Jonas. Swoosh: A Generic Approach to Entity Resolution. Technical report, Stanford InfoLab, 2006.

[8] Tim Berners-Lee. What the semantic web isn't but can represent. published online, 1998. `http://www.w3.org/DesignIssues/RDFnot.html`.

[9] Indrajit Bhattacharya and Lise Getoor. Deduplication and Group Detection Using Links. In *Proceedings of the 2004 ACM SIGKDD Workshop on Link Analysis and Group Detection*, aug 2004.

[10] Matteo Bonifacio, Paolo Bouquet, and Paolo Traverso. Enabling distributed knowledge management: Managerial and technological implications. *Informatik - Zeitschrift der schweizerischen Informatikorganisationen*, 1:23–29, 2002.

[11] Alexander Borgida and Luciano Serafini. Distributed description logics: Assimilating information from peer sources. In Stefano Spaccapi-

etra, Salvatore T. March, and Karl Aberer, editors, *J. Data Semantics I*, volume 1 of *Lecture Notes in Computer Science*, pages 153–184. Springer, 2003.

[12] Stefano Bortoli, Heiko Stoermer, and Paolo Bouquet. Foaf-O-Matic - Solving the Identity Problem in the FOAF Network. In *Proceedings of the Fourth Italian Semantic Web Workshop (SWAP2007), Bari, Italy, Dec.18-20, 2007*, December 2007. `http://CEUR-WS.org/Vol-314/43.pdf`.

[13] Paolo Bouquet, Fausto Giunchiglia, Frank van Harmelen, Luciano Serafini, and Heiner Stuckenschmidt. C-OWL: Contextualizing Ontologies. In Dieter Fensel, Katia P. Sycara, and John Mylopoulos, editors, *International Semantic Web Conference*, volume 2870 of *Lecture Notes in Computer Science*, pages 164–179. Springer, 2003.

[14] Paolo Bouquet, Luciano Serafini, and Heiko Stoermer. Introducing Context into RDF Knowledge Bases. In *Proceedings of SWAP 2005, the 2nd Italian Semantic Web Workshop, Trento, Italy, December 14-16, 2005. CEUR Workshop Proceedings, ISSN 1613-0073, online http://ceur-ws.org/Vol-166/70.pdf*, December 2005.

[15] Paolo Bouquet, Heiko Stoermer, and Daniel Giacomuzzi. OKKAM: Enabling a Web of Entities. In *i3: Identity, Identifiers, Identification. Proceedings of the WWW2007 Workshop on Entity-Centric Approaches to Information and Knowledge Management on the Web, Banff, Canada, May 8, 2007.*, CEUR Workshop Proceedings, ISSN 1613-0073, May 2007. online http://CEUR-WS.org/Vol-249/submission_150.pdf.

[16] Paolo Bouquet, Heiko Stoermer, Michele Mancioppi, and Daniel Giacomuzzi. OkkaM: Towards a Solution to the "Identity Crisis" on

the Semantic Web. In *Proceedings of SWAP 2006, the 3rd Italian Semantic Web Workshop, Pisa, Italy, December 18-20, 2006. CEUR Workshop Proceedings, ISSN 1613-0073, online http://ceur-ws.org/Vol-201/33.pdf*, December 2006.

[17] Paolo Bouquet, Heiko Stoermer, and Liu Xin. Okkam4P - A Protégé Plugin for Supporting the Re-use of Globally Unique Identifiers for Individuals in OWL/RDF Knowledge Bases. In *Proceedings of the Fourth Italian Semantic Web Workshop (SWAP2007), Bari, Italy, Dec.18-20, 2007*, December 2007. `http://CEUR-WS.org/Vol-314/41.pdf`.

[18] Karl Branting. A comparative evaluation of name-matching algorithms. In *ICAIL*, pages 224–232, 2003.

[19] Steven Carmody, Walter Gross, Theodor H. Nelson, David Rice, and Andries van Dam. A hypertext editing system for the /360. In M. Faiman and J. Nievergelt, editors, *Pertinent concepts in computer graphics*, pages 291–330. University of Illinois, Urbane, Ill., March 1969.

[20] Silvana Castano, Alfio Ferrara, Davide Lorusso, and Stefano Montanelli. The hmatch 2.0 suite for ontology matchmaking. In Giovanni Semeraro, Eugenio Di Sciascio, Christian Morbidoni, and Heiko Stoermer, editors, *Proceedings of SWAP2007, Fourth Italian Workshop on Semantic Web Applications and Perspectives*, 2007.

[21] Lois Mai Chan. *Library of Congress Subject Headings, 30th edition*. Libraries Unlimited, 2000.

[22] Fay Chang, Jeffrey Dean, Sanjuay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and

Robert E. Gruber. BigTable – A Distributed Storage System for Structured Data. In *Proceedings of OSDI 2006*, 2006.

[23] Sam Chapman. *SimMetrics*. University of Sheffield, 2006. `http://www.dcs.shef.ac.uk/~sam/simmetrics.html`.

[24] DeWitt Clinton. Opensearch description document specification, version 1.1, draft 3. Published online. `http://www.opensearch.org/Specifications/OpenSearch/1.1`.

[25] William W. Cohen. Data Integration Using Similarity Joins and a Word-based Information Representation Language. *ACM Trans. Inf. Syst.*, 18(3):288–321, 2000.

[26] William W. Cohen, Pradeep Ravikumar, and Stephen E. Fienberg. A Comparison of String Distance Metrics for Name-Matching Tasks. In *Proceedings of the XVIII International Joint Conferences on Artificial Intelligence (IJCAI) - Workshop on Information Integration on the Web (IIWeb)*, pages 73–78, Acapulco, México, 9-10 August 2003.

[27] Willy Cromwell-Kessler. Crosswalks, Metadata Mapping, and Interoperability: What does it all mean? In Murtha Baca, editor, *Metadata: Pathways to Digital Information*, pages 19–21. Getty Information Institute, 1998.

[28] Li Ding, Timothy W. Finin, Anupam Joshi, Rong Pan, R. Scott Cost, Yun Peng, Pavan Reddivari, Vishal Doshi, and Joel Sachs. Swoogle: a search and metadata engine for the semantic web. In *Proceedings of the 2004 ACM CIKM International Conference on Information and Knowledge Management, Washington, DC, USA, November 8-13, 2004*, pages 652–659, 2004.

[29] Xin Dong, Alon Halevy, and Jayant Madhavan. Reference Reconciliation in Complex Information Spaces. In *SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 85–96, New York, NY, USA, 2005. ACM Press.

[30] Dublin Core Metadata Element Set, Version 1.1, December 2006. `http://dublincore.org/documents/dces/`.

[31] Ahmed K. Elmagarmid, Panagiotis G. Ipeirotis, and Vassilios S. Verykios. Duplicate Record Detection: A Survey. *IEEE Transactions on Knowledge and Data Engineering*, 19(1):1–16, 2007.

[32] Euracert. *Euracert Accessibility Guidelines.* online `http://www.euracert.org/`.

[33] Jérôme Euzenat and Pavel Shvaiko. *Ontology matching.* Springer-Verlag, Heidelberg (DE), 2007.

[34] Ronald Fagin, Amnon Lotem, and Moni Naor. Optimal aggregation algorithms for middleware. *J. Comput. Syst. Sci.*, 66(4):614–656, 2003.

[35] Ivan P. Fellegi and Alan B. Sunter. A Theory for Record Linkage. *Journal of the American Statistical Association*, 64(328):1183–1210, 1969.

[36] Angela Fogarolli, Giuseppe Riccardi, and Marco Ronchetti. Searching information in a collection of video-lectures. In *Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications 2007*, pages 1450–1459, Vancouver, Canada, June 2007. AACE.

[37] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley, 1995.

[38] Aldo Gangemi and Valentina Presutti. A grounded ontology for identity and reference of web resources. In *i3: Identity, Identifiers, Identification. Proceedings of the WWW2007 Workshop on Entity-Centric Approaches to Information and Knowledge Management on the Web, Banff, Canada, May 8, 2007.*, 2007.

[39] Hector Garcia-Molina. Pair-wise entity resolution: overview and challenges. In Philip S. Yu, Vassilis J. Tsotras, Edward A. Fox, and Bing Liu, editors, *Proceedings of the 2006 ACM CIKM International Conference on Information and Knowledge Management, Arlington, Virginia, USA, November 6-11, 2006*, page 1. ACM, 2006.

[40] Chiara Ghidini and Fausto Giunchiglia. Local models semantics, or contextual reasoning=locality+compatibility. *Artif. Intell.*, 127(2):221–259, 2001.

[41] Ramanathan V. Guha and Rob McCool. TAP: a Semantic Web Platform. *Computer Networks*, 42(5):557–577, 2003.

[42] Joseph Hassell, Boanerges Aleman-Meza, and Ismailcem Budak Arpinar. Ontology-Driven Automatic Entity Disambiguation in Unstructured Text. In Isabel F. Cruz, Stefan Decker, Dean Allemang, Chris Preist, Daniel Schwabe, Peter Mika, Michael Uschold, and Lora Aroyo, editors, *International Semantic Web Conference*, volume 4273 of *Lecture Notes in Computer Science*, pages 44–57. Springer, 2006.

[43] Patrick Hayes. *RDF Semantics*, February 2004. http://www.w3.org/TR/rdf-mt/.

[44] Mauricio A. Hernández and Salvatore J. Stolfo. Real-world Data is Dirty: Data Cleansing and The Merge/Purge Problem. *Data Min. Knowl. Discov.*, 2(1):9–37, 1998.

[45] Mauricio A. Hernández and Salvatore J. Stolfo. Real-world Data is Dirty: Data Cleansing and The Merge/Purge Problem. *Data Min. Knowl. Discov.*, 2(1):9–37, 1998.

[46] Aidan Hogan, Andreas Harth, and Stefan Decker. Performing object consolidation on the semantic web data graph. In *i3: Identity, Identifiers, Identification. Proceedings of the WWW2007 Workshop on Entity-Centric Approaches to Information and Knowledge Management on the Web, Banff, Canada, May 8, 2007.*, 2007.

[47] Aidan Hogan, Andreas Harth, Jürgen Umbrich, and Stefan Decker. Towards a Scalable Search and Query Engine for the Web. In *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*, pages 1301–1302, 2007.

[48] IBM. *IBM DB2 Glossary*. `http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp`.

[49] Afraz Jaffri, Hugh Glaser, and Ian Millard. Uri identity management for semantic web data integration and linkage. In *3rd International Workshop On Scalable Semantic Web Knowledge Base Systems*. Springer, 2007.

[50] Greg Janée and James Frew. The adept digital library architecture. In *JCDL '02: Proceedings of the 2nd ACM/IEEE-CS joint conference on Digital libraries*, pages 342–350, New York, NY, USA, 2002. ACM Press.

[51] Dmitri V. Kalashnikov and Sharad Mehrotra. Domain-independent Data Cleaning via Analysis of Entity-relationship Graph. *ACM Transactions on Database Systems (ACM TODS)*, 31(2):716–767, June 2006.

[52] Dmitri V. Kalashnikov, Sharad Mehrotra, and Zhaoqi Chen. Exploiting Relationships for Domain-independent Data Cleaning. In *SIAM International Conference on Data Mining (SIAM SDM)*, Newport Beach, CA, USA, April 21–23 2005.

[53] Sascha Kriewel, Claus-Peter Klas, Sven Frankmlle, and Norbert Fuhr. A Framework for Supporting Common Search Strategies in DAFFODIL. In A. Rauber, C. Christodoulakis, and A M. Tjoa, editors, *Research and Advanced Technology for Digital Libraries. Proc. European Conference on Digital Libraries (ECDL 2005)*, pages 527–528. Springer, 2005.

[54] Saul Kripke. *Naming and Necessity.* Basil Blackwell, Boston, 1980.

[55] M. Kudo and S. Hada. XML Document Security based on Provisional Authorisation. In *In Proc. of the 7th ACM Conference on Computer and Communications Security*, page 8796, 2000.

[56] George Lakoff. *Women, Fire, and Dangerous Things.* University of Chicago Press, Chicago, IL., 1987.

[57] Douglas B. Lenat. Cyc: A large-scale investment in knowledge infrastructure. *Commun. ACM*, 38(11):32–38, 1995.

[58] Pierre Lévy. IEML, Finalités et structure fondamentale. In Jean-Michel Penalva, editor, *Intelligence Collective, Rencontres 2006, Presses de lécole des mines de Paris*, pages 117–136, May 2006.

[59] S. J. Liebowitz and Stephen E. Margolis. Network Externalities. In *The New Palgrave's Dictionary of Economics and the Law*. MacMillan, 1998.

[60] Javier Lopez, Antonio Mana, and Mariemma Inmaculada Yagüe del Valle. Xml-based distributed access control system. In *E-Commerce and Web Technologies, Third International Conference, EC-Web 2002, Aix-en-Provence, France, September 2-6, 2002, Proceedings*, pages 203–213, 2002.

[61] Javier Lopez, Antonio Mana, Ernesto Pimentel, José M. Troya, and Mariemma Inmaculada Yagüe del Valle. Access control infrastructure for digital objects. In *Information and Communications Security, 4th International Conference, ICICS 2002, Singapore, December 9-12, 2002, Proceedings*, pages 399–410, 2002.

[62] Antonio Mana, Javier Lopez, Juan J. Ortega, Ernesto Pimentel, and José M. Troya. A framework for secure execution of software. *Int. J. Inf. Sec.*, 3(2):99–112, 2004.

[63] Kieran McDonald and Alan F. Smeaton. A Comparison of Score, Rank and Probability-Based Fusion Methods for Video Shot Retrieval. In Wee Kheng Leow, Michael S. Lew, Tat-Seng Chua, Wei-Ying Ma, Lekha Chaisorn, and Erwin M. Bakker, editors, *CIVR*, volume 3568 of *Lecture Notes in Computer Science*, pages 61–70. Springer, 2005.

[64] Alvaro E. Monge and Charles Elkan. An Efficient Domain-Independent Algorithm for Detecting Approximately Duplicate Database Records. In *DMKD*, pages 0–, 1997.

[65] NewsML Version 1.2 Functional Specification, October 2003. http://www.newsml.org/dl.php?fn=NewsML/1.2/specification/NewsML_1.2.dtd.

[66] Ian Niles and Adam Pease. Towards a standard upper ontology. In *FOIS*, pages 2–9, 2001.

[67] NITF 3.4 XSD Schema, October 2006. http://www.nitf.org/IPTC/NITF/3.4/specification/schema/nitf-3-4.xsd.

[68] Natalya F. Noy. Semantic Integration: a Survey of Ontology-based Approaches. *SIGMOD Rec.*, 33(4):65–70, 2004.

[69] Tim O'Reilly. *The Social Network Operating System*, October 2007. online http://radar.oreilly.com/archives/2007/10/social_network_operating_system.html.

[70] Patrick Pantel, Andrew Philpot, and Eduard H. Hovy. Matching and Integration across Heterogeneous Data Sources. In *Proceedings of the 7th Annual International Conference on Digital Government Research, DG.O 2006, San Diego, California, USA, May 21-24, 2006*, pages 438–439, 2006.

[71] C. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[72] P.F. Patel-Schneider, P. Hayes, and I. Horrocks. Web Ontology Language (OWL) Abstract Syntax and Semantics. Technical report, W3C, February 2003. http://www.w3.org/TR/owl-semantics/.

[73] Luigi Pirandello. *One, None and a Hundred Thousand*. E. P. Dutton & Co., Inc., New York, 1st edition, 1933. Translated from the Italian by Samual Putnam.

[74] Willard Van Orman Quine. *Set Theory and Its Logic*. Harvard University Press, revised edition, 1969.

[75] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2007. ISBN 3-900051-07-0.

[76] Erhard Rahm and Philip A. Bernstein. A Survey of Approaches to Automatic Schema Matching. *VLDB Journal: Very Large Data Bases*, 10(4):334–350, 2001.

[77] Luciano Serafini, Heiner Stuckenschmidt, and Holger Wache. A Formal Investigation of Mapping Languages for Terminological Knowledge. In *BNAIC 2005 - Proceedings of the Seventeenth Belgium-Netherlands Conference on Artificial Intelligence, Brussels, Belgium, October 17-18, 2005*, pages 379–380, 2005.

[78] Keith Shafer, Stuart Weibel, Erik Jul, and Jon Fausey. Introduction to persistent uniform resource locators. published online., 1996. `http://purl.oclc.org/docs/inet96.html`.

[79] M. Sipser. *Introduction to the Theory of Computation*. PWS Publishing, 1997.

[80] Gleb Skobeltsyn, Toan Luu, Ivana Podnar Zarko, Martin Rajman, and Karl Aberer. Web text retrieval with a p2p query-driven index. In *SIGIR*, pages 679–686, 2007.

[81] Luis Snchez-Fernndez, Norberto Fernndez-Garca, Ansgar Bernardi, Lars Zapf, Anselmo Peas, and Manuel Fuentes. An experience with semantic web technologies in the news domain. In *Workshop Semantic Web Case Studies and Best Practices for eBusiness*, 2005.

[82] Heiko Stoermer, Paolo Bouquet, Ignazio Palmisano, and Domenico Redavid. A Context-based Architecture for RDF Knowledge Bases: Approach, Implementation and Preliminary Results. In Massimo

Marchiori, Jeff Z. Pan, and Christian de Sainte Marie, editors, *Web Reasoning and Rule Systems, First International Conference, RR 2007, Innsbruck , Austria, June 7-8, 2007, Proceedings*, volume 4524 of *Lecture Notes in Computer Science*, pages 209–218. Springer Berlin/Heidelberg, June 2007.

[83] Heiko Stoermer, Ignazio Palmisano, and Domenico Redavid. Achieving Scalability and Expressivity in an RDF Knowledge Base by Implementing Contexts. In *Proceedings of SWAP 2006, the 3rd Italian Semantic Web Workshop, Pisa, Italy, December 18-20, 2006. CEUR Workshop Proceedings, ISSN 1613-0073, online http://ceur-ws.org/Vol-201/14.pdf*, December 2006.

[84] Heiko Stoermer, Ignazio Palmisano, Domenico Redavid, Luigi Iannone, Paolo Bouquet, and Giovanni Semeraro. Contextualization of an RDF Knowledge Base in the VIKEF Project. In Shigeo Sugimoto, Jane Hunter, Andreas Rauber, and Atsuyuki Morishima, editors, *Digital Libraries: Achievements, Challenges and Opportunities, 9th International Conference on Asian Digital Libraries, ICADL 2006, Kyoto, Japan, November 27-30, 2006, Proceedings*, volume 4312 of *Lecture Notes in Computer Science*, pages 101–110. Springer, November 2006.

[85] P. F. Strawson. *Entity and Identity And Other Essays.* Cambridge University Press, 1997.

[86] SUN Microsystems, Inc. *Core J2EE Patterns – Business Delegate*, 2002. `http://java.sun.com/blueprints/corej2eepatterns/Patterns/BusinessDelegate.html`.

[87] Sun Microsystems, Inc. *Core J2EE Patterns – Data Access Object*, 2002. `http://java.sun.com/blueprints/corej2eepatterns/Patterns/DataAccessObject.html`.

[88] SUN Microsystems, Inc. *Core J2EE Patterns – Transfer Object*, 2002. `http://java.sun.com/blueprints/corej2eepatterns/Patterns/TransferObject.html`.

[89] Andrei Tamilin. *Distributed Ontological Reasoning: Theory, Algorithms, and Applications.* PhD thesis, University of Trento, Italy, 2007.

[90] Martin Theobald, Gerhard Weikum, and Ralf Schenkel. Top-k query evaluation with probabilistic guarantees. In *VLDB*, pages 648–659, 2004.

[91] U.S. General Services Administration. *Section 508 Accessibility Guidelines.* online `http://www.section508.gov`.

[92] VIKEF Consortium. VIKEF R&D Highlight - Semantic Infusion for Content Digestion. published online, 2007. `ftp://ftp.cordis.europa.eu/pub/ist/docs/kct/vikef-poster2b-a3_en.pdf`.

[93] W3C. *Web Content Accessibility Guidelines.* online `http://www.w3.org/TR/WCAG10-HTML-TECHS`.

[94] W3C. XQuery 1.0: An XML Query Language. Published online., January 2007. `http://www.w3.org/TR/xquery/`.

[95] Stuart L. Weibl and Traugott Koch. The Dublin Core Metadata Initiative - Mission, Current Activities, and Future Directions. *D-Lib Magazine*, 6(12), 2000.

[96] William E. Winkler. The State of Record Linkage and Current Research Problems. Technical report, Statistical Research Division, U.S. Census Bureau, Washington, DC, 1999.

[97] Ian H. Witten and David Bainbridge. *How to Build a Digital Library.* Elsevier Science Inc., New York, NY, USA, 2002.

[98] Mariemma Inmaculada Yagüe, María del Mar Gallardo, and Antonio Mana. Semantic access control model: A formal specification. In *Computer Security - ESORICS 2005, 10th European Symposium on Research in Computer Security, Milan, Italy, September 12-14, 2005, Proceedings*, pages 24–43, 2005.

[99] Mariemma Inmaculada Yagüe, Antonio Mana, and Javier Lopez. A metadata-based access control model for web services. *Internet Research: Electronic Networking Applications and Policy*, 15:99–116(18), 2005.

[100] Mariemma Inmaculada Yagüe, Antonio Mana, and Francisco Sanchez. Semantic interoperability of authorizations. In *Proceedings of the 2nd International Workshop on Security In Information Systems, WOSIS 2004*, pages 269–278, 2004.

[101] Marcia Lei Zeng and Lois Mai Chan. Metadata Interoperability and Standardization A Study of Methodology Part II Achieving (Interoperability at the Record and Repository Levels). *D-Lib Magazine*, 12(6), 2006.

[102] Xuan Zhou, Julien Gaugaz, Wolf-Tilo Balke, and Wolfgang Nejdl. Query Relaxation using Malleable Schemas. In *SIGMOD '07: Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 545–556, New York, NY, USA, 2007. ACM Press.

Appendix A

XML Schemas of API Data Structures

A.1 AnnotatedQuery

Listing A.1: The AnnotatedQuery XML Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
  xmlns:okkam="http://www.okkam.org/schemas/AnnotatedQuery"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.okkam.org/schemas/AnnotatedQuery"
  elementFormDefault="qualified"
  attributeFormDefault="qualified"
  version="0.6">
  <xs:element name="AnnotatedQuery">
    <xs:annotation>
      <xs:documentation>
        AnnotatedQuery is the root object of the tree that is used to represent a matching query posed by a client application.
        I contains obligatory and optional components that are used by the server to compose query strategies for finding good matches.
      </xs:documentation>
    </xs:annotation>
    <xs:complexType>
```

```
      <xs:sequence>
        <xs:element name="QueryString" type="xs:string">
          <xs:annotation>
            <xs:documentation>Simply the copy of the query
               string. Required, in case to be generated by the
               client application.</xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element name="QueryContext" type="okkam:Context"
           minOccurs="0">
          <xs:annotation>
            <xs:documentation>Optional. The context in which the
               query was posed. We have to see which parameters
               we think are useful.</xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element name="QueryMetadata" type="okkam:Metadata"
           minOccurs="0">
          <xs:annotation>
            <xs:documentation>Optional. What we can say about
               the query in general.</xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element name="QueryAnnotation"
           type="okkam:Annotation">
          <xs:annotation>
            <xs:documentation>token-by-token annotation. this is
               NOT optional, in case the client must perform a
               tokenization of the query and at least provide a
               list of values.</xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element name="QueryExpansion"
           type="okkam:ExpansionHints" minOccurs="0">
          <xs:annotation>
            <xs:documentation>Optional. Hints to the server for
               potential query expansion.</xs:documentation>
          </xs:annotation>
```

```
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <!--
*
* the Context type
*
-->
  <xs:complexType name="Context">
    <xs:annotation>
      <xs:documentation> First draft specification of what we
         want to say about the "context" in which a query was
         posed.</xs:documentation>
    </xs:annotation>
    <xs:all>
      <xs:element name="lang" type="xs:language" minOccurs="0"/>
      <xs:element name="location" type="okkam:Location"
         minOccurs="0"/>
      <xs:element name="device" type="okkam:Device"
         minOccurs="0"/>
      <xs:element name="clientIdentifier" type="xs:string"
         minOccurs="0">
        <xs:annotation>
          <xs:documentation>A string identifying the client
             software (protege plugin, web client,
             etc.)</xs:documentation>
        </xs:annotation>
      </xs:element>
    </xs:all>
  </xs:complexType>
  <!--
*
* the Device type
*
-->
  <xs:complexType name="Device">
    <xs:annotation>
```

```
      <xs:documentation> To be defined: the type of device the user is working on.</xs:documentation>
    </xs:annotation>
    <xs:all>
      <xs:element name="name" type="xs:string"/>
    </xs:all>
  </xs:complexType>
  <!--
*
* the Location type
*
-->
  <xs:complexType name="Location">
    <xs:annotation>
      <xs:documentation>To be defined. Need to check how to easily implement this.</xs:documentation>
    </xs:annotation>
    <xs:all>
      <xs:element name="name" type="xs:string"/>
    </xs:all>
    <xs:attribute ref="okkam:OkId" use="optional"/>
  </xs:complexType>
  <!--
*
* the Metadata type
*
-->
  <xs:complexType name="Metadata">
    <xs:annotation>
      <xs:documentation> First draft specification, what we can say about the query in general</xs:documentation>
    </xs:annotation>
    <xs:all>
      <xs:element name="limit" type="xs:int" minOccurs="0">
        <xs:annotation>
          <xs:documentation>The maximum number of desired results from the server.</xs:documentation>
        </xs:annotation>
```

```
      </xs:element>
      <xs:element name="typeHint" type="xs:string" minOccurs="0">
        <xs:annotation>
          <xs:documentation>A hint about the type of entity we are looking for.</xs:documentation>
        </xs:annotation>
      </xs:element>
    </xs:all>
  </xs:complexType>
<!--
*
* the Relevance enum type
*
-->
<!-- the old solution
  <xs:simpleType name="Relevance">
    <xs:restriction base="xs:NMTOKEN">
      <xs:enumeration value="low"/>
      <xs:enumeration value="medium"/>
      <xs:enumeration value="high"/>
    </xs:restriction>
  </xs:simpleType>
 -->

  <xs:simpleType name="Relevance">
        <xs:restriction base="xs:integer">
          <xs:minInclusive value="1" />
          <xs:maxInclusive value="100" />
        </xs:restriction>
  </xs:simpleType>



<!--
*
* the Annotation type and its components (tokens)
*
-->
```

```
<xs:complexType name="Annotation">
  <xs:annotation>
    <xs:documentation>
          Sequence of tokens of the query with individual
          annotations. Tokens are obligatory.
        </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="Token" type="okkam:QueryToken"
        maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="QueryToken">
  <xs:annotation>
    <xs:documentation> Representation of a single query
        token</xs:documentation>
  </xs:annotation>
  <xs:all>
    <xs:element name="label" type="xs:string" minOccurs="0">
      <xs:annotation>
        <xs:documentation>The label of a token, if
            available.</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="value" type="xs:string">
      <xs:annotation>
        <xs:documentation>The value part of a
            token.</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="typeHint" type="xs:string" minOccurs="0">
      <xs:annotation>
        <xs:documentation>a string giving a hint about the
            type the token.</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="relevanceHint" type="okkam:Relevance"
        minOccurs="0">
```

```
        <xs:annotation>
          <xs:documentation>a choice of low-med-high about the
             relevance of this token in the
             query.</xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="namespaceHint" type="xs:string"
         minOccurs="0">
        <xs:annotation>
          <xs:documentation>hint about a potentially
             "well-known" namespace, e.g. FOAF</xs:documentation>
        </xs:annotation>
      </xs:element>
    </xs:all>
  </xs:complexType>
  <!--
*
* the ExpansionHints type and its components
*
-->
  <xs:complexType name="ExpansionHints">
    <xs:annotation>
      <xs:documentation>a list of hints. if the ExpansionHints
         element is used, at least one hint must be given.
         </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="Hint" type="okkam:ExpansionHint"
         maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="ExpansionHint">
    <xs:annotation>
      <xs:documentation>name-value pairs that the client
         proposes to the server for potential query expansion.
         </xs:documentation>
    </xs:annotation>
    <xs:all>
```

```
      <xs:element name="label" type="xs:string" minOccurs="0"/>
      <xs:element name="value" type="xs:string"/>
      <xs:element name="typeHint" type="xs:string"
         minOccurs="0"/>
    </xs:all>
  </xs:complexType>
  <!--
*
* the OKKAM ID global attribute
*
-->
  <xs:attribute name="OkId" type="xs:anyURI">
    <xs:annotation>
      <xs:documentation> The OKKAM ID of an
         element</xs:documentation>
    </xs:annotation>
  </xs:attribute>
</xs:schema>
```

A.2 OkkamURIResult

Listing A.2: The OkkamURIResult XML Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
   elementFormDefault="qualified"
   attributeFormDefault="unqualified">
  <xs:complexType name="Confidences">
    <xs:sequence>
      <xs:element name="Confidence" type="xs:float"
         minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="OkkamURIResult">
    <xs:annotation>
      <xs:documentation>Return value for search
         query.</xs:documentation>
    </xs:annotation>
```

```
    <xs:complexType>
      <xs:all>
        <xs:element name="Result" type="Uris" minOccurs="0">
          <xs:annotation>
            <xs:documentation>List of top-k
               URIs.</xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element name="Confidences" type="Confidences">
          <xs:annotation>
            <xs:documentation>List of confidences, one for each
               URI in Result, same order.</xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element name="Message" type="xs:string"
           minOccurs="0">
          <xs:annotation>
            <xs:documentation>Server message as String, for
               display to user.</xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element name="Code" type="xs:int" minOccurs="0">
          <xs:annotation>
            <xs:documentation>
# 0: OK, data attached.
# 1: Internal OkkamCore error caused by JAXBException.
# 2: Internal OkkamCore error with the database (SQLException).
# 3: No Query Specified. The input query was empty.
# 4: Processing OK, but no results were found.</xs:documentation>
          </xs:annotation>
        </xs:element>
      </xs:all>
    </xs:complexType>
  </xs:element>
  <xs:complexType name="Uris">
    <xs:annotation>
      <xs:documentation>List of String</xs:documentation>
    </xs:annotation>
```

```
    <xs:sequence>
      <xs:element name="Uri" type="xs:string" minOccurs="0"
         maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

A.3 EntityProfile

Listing A.3: The EntityProfile XML Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
   elementFormDefault="qualified"
   attributeFormDefault="unqualified">
  <xs:element name="EntityProfile">
    <xs:annotation>
      <xs:documentation>Comment describing your root
         element</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:all>
        <xs:element name="Labels">
          <xs:complexType>
            <xs:complexContent>
              <xs:extension base="LabelsType"/>
            </xs:complexContent>
          </xs:complexType>
        </xs:element>
        <xs:element name="References" type="ReferencesType"
           minOccurs="0"/>
        <xs:element name="AssertionsOfIdentity"
           type="AssertionsOfIdentityType" minOccurs="0"/>
        <xs:element name="AlternativeIdentifiers"
           type="AlternativeIdentifiersType" minOccurs="0"/>
        <xs:element name="OkkamURI" type="xs:anyURI"/>
        <xs:element name="PreferredIdentifier" type="xs:string"
           minOccurs="0"/>
```

```
        <xs:element name="WordnetIdentifier" type="xs:string"
            minOccurs="0"/>
      </xs:all>
    </xs:complexType>
  </xs:element>
  <xs:complexType name="LabelsType">
    <xs:annotation>
      <xs:documentation>List of labels</xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="Label" type="LabelType"
          maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="ReferencesType">
    <xs:annotation>
      <xs:documentation>List of references</xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="Reference" type="ReferenceType"
          minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="AssertionsOfIdentityType">
    <xs:annotation>
      <xs:documentation>List of URI</xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="okkamuri" type="xs:anyURI" minOccurs="0"
          maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="AlternativeIdentifiersType">
    <xs:annotation>
      <xs:documentation>List of String</xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="Identifier" type="xs:string"
```

```
        minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="LabelType">
    <xs:annotation>
      <xs:documentation>Representation of a name/value pair in
        the profile</xs:documentation>
    </xs:annotation>
    <xs:all>
      <xs:element name="prefix" type="xs:string"/>
      <xs:element name="value" type="xs:string"/>
    </xs:all>
  </xs:complexType>
  <xs:complexType name="ReferenceType">
    <xs:annotation>
      <xs:documentation>A reference</xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="type" type="xs:string"/>
      <xs:element name="value" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

This document and the work described in it was (almost) entirely made with Open Source Software:

- LaTeX(http://www.latex-project.org)
- TeXnicCenter (http://texniccenter.org)
- JabRef (http://jabref.sourceforge.net)
- Miktex (http://miktex.org)
- Ghostscript (http://pages.cs.wisc.edu/~ghost/)
- R (http://www.r-project.org and [75])
- JavaNCSS (http://www.kclee.de/clemens/java/javancss)
- The GNU Image Manipulation Program (GIMP) (http://www.gimp.org)
- OpenOffice (http://www.openoffice.org)
- The SUN Java Developer's Kit (http://java.sun.com)
- The Eclipse IDE (http://www.eclipse.org)
- JBoss (http://www.jboss.org)
- Apache Tomcat and numerous other tools and libraries of the Apache Project (http://www.apache.org)
- Subversion (http://subversion.tigris.org)
- GNU/Linux (http://www.gnu.org)

Consequently, the source code of the OKKAM prototype as described in Chapter 5 has been released as Open Source under the Mozilla Public License 1.1 (http://www.mozilla.org/MPL/MPL-1.1.html),