Master's Thesis

Creating Scientific Social Webs

carried out at the

Faculdade de Engenharia da
Universidade do Porto
Portugal

under the guidance of
Prof. Jaime Villate

by

José Pedro Macedo Alves Ferreira

Geneva, 11th April 2008 _____

Contents

List of Figures

Abstract

The "arrival" of the Semantic Web has been for long announced by researchers and enthusiasts. The original vision, imagined by the creator of the World Wide Web, Tim Berners-Lee, portrays a world where the creative power of human beings and the large processing power of machines unite into a hybrid social network of human and artificial agents. So far, nothing that resembles this vision that some qualify as a "technological utopia" has been reached, but the Semantic Web community cannot be blamed for the lack of activity. During the last couple of years, several ontologies have been developed, and standards have been set, bringing the Semantic Web to a level of stability that has never been experienced before.

One of the "overlays" that has always been present in the "traditional web" was that of "networks of researchers", or "scientific networks". Though highly interleaved in the Web of documents, these communities have used the WWW since its beginnings, as a way to share publications, and other relevant information. Nowadays, with the emergence of the Semantic Web, these networks of researchers have a unique opportunity to establish stronger connections, by using their own information for their benefit.

This document provides an overview over the Semantic Web, the technologies that make possible its implementation, and the different repositories of information that it provides to its users. After this introduction, the focus is placed on the application of Semantic Web technologies to the construction of "Semantic Scientific Networks", or "Scientific Webs". Several different ontologies and data sources (connected to this subject) are presented, as well as possible ways to articulate them.

An experimental framework for making data from Object Oriented Databases available as RDF, called OURSE, is introduced as well. OURSE is used as a tool to make information about scientific conferences, present in the Indico conference management system (developed by CERN), available as RDF. Finally, the idea of a "mashup" platform, called Scio, targeted at integrating all the information concerning these "networks" is presented, as well as some design issues that would arise in a possible implementation.

Resumo

A chegada da Web Semântica tem sido anunciada desde há muito por investigadores e entusiastas. A visão original, imaginada pelo próprio criador da World Wide Web, Tim Berners-Lee, retrata um mundo onde a capacidade criativa to ser humano e o amplo poder de processamento das máquinas se fundem numa rede social híbrida, constituída por agentes artificiais e humanos. Até ao momento, nada que se pareça com esta visão que alguns classificam de "utopia tecnológica" foi ainda alcançado, mas a comunidade de entusiastas da Semantic Web não peca por passividade. Ao longo dos últimos dois anos, várias ontologias foram desenvolvidas, e novos *standards* foram criados, elevando a Semantic Web a um patamar de estabilidade que nunca havia sido alcançado.

Uma das "camadas" que desde sempre estiveram presentes na "web tradicional" foi a das "redes de investigadores", ou "redes científicas". Encontrando-se altamente entrelaçadas na Web de documentos, estas comunidades utilizam-na desde os seus primórdios, como forma de partilhar publicações, e outra informação relevante. Nos dias de hoje, com o surgimento da Semantic Web, estas redes de investigadores têm uma oportunidade única de estabelecer ligações mais fortes, utilizando a informação que produzem para seu próprio benefício.

Este documento fornece uma visão geral sobre a Web Semântica, as tecnologias que tornam possível a sua implementação, e os diferentes repositórios de informação que esta fornece aos seus utilizadores. Depois de uma introdução, o foco é apontado para a aplicação da Semantic Web à construção de "Redes Semânticas Científicas", ou "Webs Científicas". Várias ontologias distintas ligadas ao tema são apresentadas, assim como possíveis formas de as articular.

Uma *framework* experimental, denominada OURSE, concebida para traduzir dados de Bases de Dados Orientadas por Objectos para RDF, é também apresentada. A *framework* OURSE é utilizada como ferramenta para disponibilizar informação sobre conferências científicas, armazenada no sistema de gestão de conferências Indico (desenvolvido no CERN), na forma de RDF. Finalmente, uma plataforma "mashup", denominada Scio, e orientada para a integração da informação que descreve tais "redes científicas", é exposta, assim como alguns problemas que surgiriam aquando da sua implementação.

Acknowledgements

I would like to thank everyone that in some way contributed for the writing of these pages. Since I cannot enumerate such an amount of people, I will just focus on the most obvious contributions:

The people from the rdflib project, for providing this wonderful piece of software, and helping me out with some technical questions;

The creator of Oort, Niklas Lindström, for his interest on the concepts behind OURSE, and his availability to discuss them with me;

The Semantic Web Interest Group of W3C, through its IRC channel and mailing list, for their prompt help in clearing out any question that I would have about the Semantic Web. It is great being able to share our points of view with some of the people that actually created concepts that this document describes. It is hard to find such a dynamic community, that everyday has some new thing to surprise us with;

Sir Tim Berners-Lee, who has been already included in the previous point, but deserves this mention for his role as the "great architect" of the SW, and his inspiration, through his book "Weaving the Web";

My colleagues at CERN, for their help with Indico-related matters, and for their support in my everyday life: Thomas Baron, José Benito Gonzalez and David Bourillot especially.

"We hypostasize information into objects. Rearrangement of objects is change in the content of the information; the message has changed. This is a language which we have lost the ability to read. We ourselves are a part of this language; changes in us are changes in the content of the information. We ourselves are information-rich; information enters us, is processed and is then projected outwards once more, now in an altered form. We are not aware that we are doing this, that in fact this is all we are doing."

Philip K. Dick, "VALIS"

1 Introduction

The Semantic Web is currently one of the most expected technological phenomenons. It started as the idea of Tim Berners-Lee, the inventor of the World Wide Web (and Director of the W3C - World Wide Web Consortium), of an extension of the "classic web" that would allow at last the representation and exchange of knowledge, in a universal and structured way, thus enabling machines to participate in the construction of such an information network at a deeper level. Such a web should be flexible enough, in a way that even the "languages" that would be used for communication could be described using logical constructs, and extended by the means of universal mechanisms: the concept of ontology appears as a way to define how knowledge from a particular field can be expressed. There would have to exist ontologies to describe everything, from animal species to people, relationships, and even feelings, so that this "new web" (or rather an extension of the WWW, as Berners-Lee himself prefers to define it) would have at least the same expressive power as the "old" one.
Berners-Lee chose the RDF model as the cornerstone of his Semantic Vision. The *Resource Description Framework* is an information-modelling tool, that can be represented through different notations and formats. It defines the most rudimentary concepts, the building blocks of a Semantic Web of Linked Data, that more complex structures, such as ontologies, base on. RDF represents things in a very simple way, through triples with a subject, a predicate and an object. These triples can be used to state things like "I like peanuts", "I live in France", but also (through composition mechanisms) more complex constructions like "I know someone whose name is Bob, living in a country where people speak french". But the Web of Knowledge doesn't limit itself to statements: developers all over the world are currently working on machine reasoning mechanisms, that perform operations of logical inference over ontologies, in an attempt to achieve the most basic processes of logical thinking. The borders of computer science, information science, philosophy and logic blur more and more, the deeper the research on the Semantic Web goes. Suddenly, the web is not anymore about fragments of linked text, but rather "raw information": ontologies, descriptions, rules, and everything else that models human knowledge.
In spite of the optimistic vision of Berners-Lee, and the vast Semantic Web Community, not everyone is sure that such an ambitious project will strive. Some consider it an utopia, others believe that it will fail, and there are, as well, those who think that it will simply create a "semantic ghost" of the current web, with no practical use. In fact, the community is well aware

of the problems that such a revolutionary concept introduces in terms of privacy, trust, and even economics. The idea of linked information, available to everyone, with no barriers or borders, confuses many minds, and provokes concern in some others.
This document will try to provide a look into the Semantic Web, from the perspective of what has been a specific, semi-isolated layer of the "normal web", from its beginning: Research Networks. The WWW was primarily created, at CERN, as a way for scientists to preserve their information, allowing them to search and share documents (and keep them persistent) without having to print and index them in folders. Through the years, researchers of Computer Science and other areas of knowledge have adopted the web as the primordial means for storing and sharing their publications, organizing events, and establishing communication among interest groups. The Semantic Web can be seen as an opportunity to improve the quality of the services that are provided to the scientific community, through the construction of structures that will be labeled as "Semantic Web Research Networks", "Scientific Webs","Social Research Networks", or any other designation that implies a network of people that participate in research, and interact using the (Semantic) Web.
That said, this research document has three main objectives:

> Introducing the Semantic Web, the RDF model, and the technologies that currently support them;
>
> Presenting the Semantic Web and the RDF model as a viable and efficient solution for the construction of a worldwide scientific web, through the implementation of a specific architecture and the utilization of existing ontologies;
>
> Presenting solutions for the problem of dealing with existing repositories of data that need to be made available as RDF for such an effort to succeed. At this point, the Indico[1] platform, and the ZODB database, employed by the former as its primordial data store, will be used as a case-study.

1.1 Organization of this thesis

This document is organized into six sections. The first section provides an introduction to the subject of the Semantic Web, and related technologies:

[1] http://www.cern.ch/indico/

the RDF model, ontology languages and the most widespread ontologies, as well as querying languages. Then, in section 2, the problem of Scientific Webs is introduced, as well as the different "pieces of the puzzle" that are part of it. Section 3 provides an insight into the state of art in Semantic Web technology, more specifically in the availability of "legacy storage" as RDF, and the presence of information about scientific networks in the Semantic Web. The following section presents a solution, based on the OURSE framework, and the utilization of existing ontologies. Then, Section 5 provides an evaluation of the work that was performed, and some ideas for future work. Finally, the last section is devoted to summarizing the outcomes and conclusions of the document, and the research work that supported it.

1.2 The Semantic Web

> *"I have a dream for the Web... and it has two parts. In the first part, the Web becomes a much more powerful means of collaboration between people. [...] In the second part of the dream, collaborations extend to computers. Machines become capable of analyzing all the data on the Web – the content, links, and transactions between people and computers. A 'Semantic Web', which should make this possible, has yet to emerge, but when it does, the day-to-day mechanisms of trade, bureaucracy and our daily lives will be handled by machines talking to machines. The 'intelligent agents' people have touted for ages will finally materialize."*
> (Tim Berners-Lee) [10]

Tim Berners-Lee, creator of the World Wide Web, once imagined a world where the creative power of human beings and the large processing power of machines would unite into a hybrid social networks of human and artificial agents. Qualified by some as a "technological utopia", and seen by many others as the next step in the evolution of the Web, the Semantic Web is definitely one of the phenomenons in computer science and information technology that has generated the most substantial expectations in researchers, entrepreneurs and some everyday Web users. From the "epiphany" of Berners-Lee, a large-scale project, supported by the World Wide Web Consortium (W3C), was put in motion, involving academics and enterprises, as well as some hobbyists. Several working groups for the Semantic Web exist in the W3C, taking gradual steps towards the creation of new standards and the solution of design problems. The idea that is at the core of the Semantic Web is the transformation of the Web into a universal medium for data, in-

formation and knowledge exchange, that not only humans, but machines as well, can profit from. This implies the definition of data models that allow information to be specified in machine-understandable ways, without losing expressive power, and keeping it readable for humans. The answer to these constraints is the Resource Description Framework, one of the cornerstones of the Semantic Web.
The architecture of the Semantic Web can be described as a stack of "layers" that pile up, in what is informally called the "Semantic Web Layer Cake"(fig. 1). This "cake" has the concept of Universal Resource Identifier (URI) at its base, as some kind of "id card" that each resource in the Semantic Web possesses. This is the base where RDF builds on, as well as XML, that shares some of its concepts with the RDF model. Above RDF, query, ontology and rule languages are constructed, as basic tools that give support to the layers of logic and proof (that involve reasoning and artificial intelligence), and, finally, a layer of "trust", that will be responsible for keeping the Semantic Web cohesive, damping the effects of manipulation, and introducing some hierarchy in terms of "authority" of data sources.

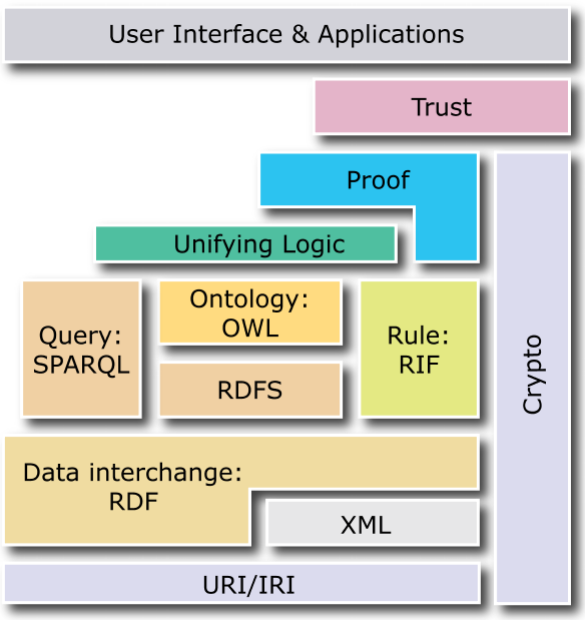

Figure 1: The "Semantic Web Layer Cake"

1.3 The Resource Description Framework (RDF)

[46] defines RDF as being "a language for representing information about resources in the World Wide Web". Although it can be used as a language for description of characteristics of web resources (i.e. creation date or authorship information about a web page), it allows as well the description of things that "can be identified on the Web, even when they cannot be directly retrieved on the Web" [46]. This last characteristic turns RDF into a valuable tool in describing not only web resources, but physical things and locations, people, and even abstract concepts, such as languages and tastes. Following the philosophy of the Semantic Web (§1.2), RDF was conceived to be understood by machines, working as a means to exchange information about resources.

Identifying Resources As a web-targeted language, RDF identifies concepts using URIs (Universal Resource Identifiers) ([60, §2]) and URI references ([39, §6.4]). A URI specifies in an unambiguous form the means of retrieving a specific resource, indicating mechanisms/protocols necessary to its obtention. As an example, `http://www.fe.up.pt/` specifies the root ("/") of the HTTP server present at the address `www.fe.up.pt`. This last URI belongs to a specific category of identifiers known as URLs (Universal Resource Locator), that aim at specifying *where* the resource is, and are widely used, mostly in the World Wide Web. The other category that falls inside the scope of URIs is that of URNs (Universal Resource Name). A URN indicates *what* the resource is, rather than *where*. An example of such a name would be `urn:idcard:1278923`, that identifies a national citizen, by its identification number, in the `idcard` namespace. The most widely used way to address semantic web resources is still through the use of URIs, not only for historical reasons, but also for the fact that the resources often exist as "physical" entities, somewhere in the web.
The concept of "URI reference" may be seen as a generalization of URIs, in order to encompass non-absolute paths[56]. Some examples are provided, to better illustrate the differences among the different types of URI references:

> `http://www.fe.up.pt/` - a valid, absolute URI, identifying the resource at the root of the `www.fe.up.pt` location (in this case, a WWW server);
>
> `urn:FEUP:020509059` - a URI (in this specific case, a URN), that refers to the resource 020509059 in the `FEUP` namespace. In this case, it could

be interpreted as a student identification number of the engineering school with the same name;

`http://www.fe.up.pt/si/` - a valid, absolute URI, identifying the resource at the root of the `si` hierarchy, in the same server;

`http://www.fe.up.pt/si/#menu` - an absolute URI, this one specifying a fragment of the resource that was previously mentioned, called `menu`;

`/si/` - a relative URI, that identifies the resource located at the root of the `si` hierarchy, that is itself located at the root of the current location. Obviously, this kind of schema has no validity if not included in the context of a specific location;

`../../si` - once again, a relative URI, this one using "path segments" as a way to go up in the name hierarchy;

`#menu` - a relative URI that references a fragment of the current resource;

It is worthy to note that these addressing schemes are not necessarily bound to a specific interpretation. If in `http://www.fe.up.pt/si/#menu`, the "menu" fragment can be seen as an (X)HTML anchor, and `si` as a directory, this is not always the case. The URI schema provides a way of placing resources inside hierarchies, even if these have not a "physical" existence (or even if they don't exist at all). As an example, `http://www.w3.org/People/Berners-Lee/card#i` specifies a fragment of a resource, hierarchized as `www.w3.org/People/Berners-Lee/card` : "a card belonging to someone called "Berners-Lee"" is just one possible interpretation of this URI. This URI resolves to an actual representation, in RDF, of the "profile" of Tim Berners-Lee. However, it could be used in the RDF model, even if it didn't exist: for instance, `http://wonderland.com/animals/Cheshire_Cat` does not resolve to any accessible physical resource (at the time of the writing of this document). In spite of its ethereal nature, this resource can be mentioned in every possible RDF graph[2].

[2]Even if the mentioned examples have used the `http:` prefix, as a way to reinforce the idea that an RDF resource does not need to be resolvable, the same applies for each possible scheme, implemented (`ftp`, `irc`...) or not (`teleport`, `warp portal`...)

Relating Concepts The advantage of RDF resides in the fact that it allows the specification of relations between resources. It does so through the use of "triples", tuples in the form *(subject, predicate, object)*, that describe things in a way that resembles object-oriented languages. For instance, the triple `(Alice, love, Bob)` could be interpreted as representing the statement "Alice loves Bob". In an object-oriented fashion, we could state that `Alice.love = Bob`. However, RDF goes beyond the traditional OO paradigm, and allows a number of extra freedoms, that increase its expressiveness[50, §2.2].
A group of RDF triples can be seen as a DLG (Direct Labelled Graph), in which the connected nodes are the *subject* and the *predicate*, and the *verb* works as the "label". The previous example is represented in figure 2. However, it is not hard to evolve from this to more complex relationships, that can be perfectly expressed using this model (see figure 3).

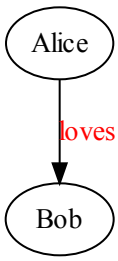

Figure 2: "Alice loves Bob" - represented in the form of a DLG

The binary nature of the relations between nodes in the RDF model brings some limitations with it. For instance, in the graph of figure 3, we might want to specify that Bob knew Alice through Desmond. This is not clearly stated by the current representation. Thus, a solution that approximates the model of figure 4 would be needed. Here, an extra node is used as an abstract entity ("Acquaintance") that serves the sole purpose of specifying a "proxy" (`through`) in the process.

Types of Nodes Until now, graph nodes have been used to represent "entities", more specifically, people. As it was already seen in this document, in the RDF model, these resources are represented through the use of URIs.

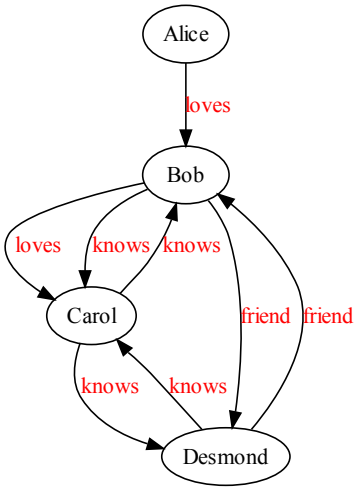

Figure 3: "Alice loves Bob, but Bob loves Carol, whom he happens to know through his friend Desmond"

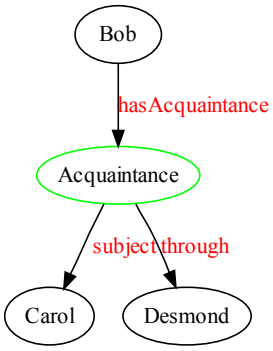

Figure 4: "Bob knows Carol through Desmond." This representation clearly indicates this fact, contrarily to what happened in the previous example.

Our fictitious friends could be easily translated to this paradigm:

`http://imaginarypeople.com/People/Alice`

`http://imaginarypeople.com/People/Bob`

`http://imaginarypeople.com/People/Carol`

`http://imaginarypeople.com/People/Desmond`

Other options would include `http://imaginarypeople.com/People#Alice` or `http://imaginarypeople.com/People/Alice#me`, or even `http://imaginarypeople.com/People/Alice/#me`. All of them are valid URIs, and all of them are different. In the same way, the relations that we established among them can be represented by:

`http://imaginarypeople.com/Relationships#loves`

`http://imaginarypeople.com/Relationships#knows`

Once again, several representations would be possible. The relations involved in our "acquaintance model" are no exception to this:

`http://imaginarypeople.com/Relationships#hasAcquaintance`

`http://imaginarypeople.com/Relationships#through`

`http://imaginarypeople.com/Relationships#subject`

However, for the information in figure 4 to be totally expressed in the model, we must be able to state that one of the nodes belongs to the "class" `Acquaintance`. Classes can also be represented in RDF, much in the same way that other resources are:

`http://imaginarypeople.com/Relationships#Acquaintance`

RDF expresses the notion of "instance of a class" through the `http://www.w3.org/1999/02/22-rdf-syntax-ns#type` property. Thus, the triple (`http://imaginarypeople.com/Acquaintances/1`, `http://www.w3.org/1999/02/22-rdf-syntax-ns#type`, `http://imaginarypeople.com/Relationships#Acquaintance`) states that the mentioned resource (`http://imaginarypeople.com/Acquaintances/1`) belongs to the "Acquaintance" class.

Namespaces The need to write the full URL of a resource in order to refer to it has caused the introduction of "namespace abbreviations" as a way to improve human readability of RDF descriptions, and reduce the overall "byte size" of the documents themselves. In our example, every person belongs to the `http://imaginarypeople.com/People/` namespace, and every relation is contained in `http://imaginarypeople.com/Relationships#`. If we define the abbreviations `people` and `relationships` for the two namespaces, respectively, things become much more readable:

`people:Alice`

`relationships:loves`

(`people:Alice`, `relationships:loves`, `people:Bob`)

Every binding between an abbreviation, or "prefix", and a namespace is only valid inside a specific context. Normally, a document that describes an RDF graph includes namespace bindings in its header. The major RDF notation formats include constructs that allow the document creator to clearly state those bindings[3].

The N3 Notation Using the example of figure 3, one can construct a textual representation of the RDF triples that compose the graph.

Listing 1 A representation of the RDF triples, using the N3 syntax

```
people:Alice relationships:loves people:Bob .

people:Bob relationships:loves people:Carol ;
           relationships:knows people:Carol ;
           relationships:knows people:Desmond .

people:Carol relationships:knows people:Bob ;
             relationships:knows people:Desmond .

people:Desmond relationships:knows people:Bob ;
               relationships:knows people:Carol .
```

This example is written as a piece of N3 code. N3 (Notation3) [7] is a notation for RDF serialization, designed as an alternative to the not very

[3] N-Triples[5, §3], a subset of N3/Turtle, is clearly an exception, since it was conceived as a minimalist language.

human-friendly RDF/XML[3] format (currently the most used in the Semantic Web). N3 includes, in addition to the constructs that allow the specification of RDF graphs, support for variables and quantifiers, in order to express logic-based rules, and some other constructs that allow a greater expressive power. In the context of this document, a subset of N3, Turtle ("Terse RDF Triple Language")[4], will be used, since it provides all the basic elements for expressing RDF triples in graphs. It is worthy to notice the way in which the triples are specified, with statements about the same subject separated by semicolon, and terminated by a period. The same effect could be achieved using only simple `subject predicate object` statements.
The example of listing 1 wouldn't, by itself, constitute a valid N3/Turtle graph serialization, due to the lack of bindings between prefixes and namespaces. Thus, the addition of `prefix` directives would be needed (listing 2).

Listing 2 The missing header for our N3/Turtle example

```
@prefix relationships: <http://imaginarypeople.com/Relationships#> .
@prefix people: <http://imaginarypeople.com/People/> .
```

Joining the examples from listings 2 and 3 (exactly in this order), one obtains a valid N3/Turtle file, that specifies a graph with 3 nodes and 8 arcs (triples). Representing the graph from figure 4 would be equally simple:

Listing 3 The graph from figure 4, represented as N3/Turtle. The `a` relation is a shorthand for `rdf:type`.

```
@prefix relationships: <http://imaginarypeople.com/Relationships#> .
@prefix people: <http://imaginarypeople.com/People/> .

people:Bob relationships:hasAcquaintance
                            <http://imaginarypeople.com/Acquaintances/1>.

<http://imaginarypeople.com/Acquaintances/1>
                            a relationships:Acquaintance;
                            relationships:through people:Desmond;
                            relationships:subject people:Carol.
```

One extra node and arc are added, in order to represent the fact that the node, marked in green in the figure, belongs to the class "Acquaintance".

Blank Nodes One other concept that RDF introduces is that of a "blank node" or `BNode`[46, §3.2]. A blank node doesn't have a URI, and is not

addressable from external graphs. In the example of fig. 4/listing 3, `http://imaginarypeople.com/Acquaintances/1` is clearly a resource that is not useful outside the context it is declared in. We might say that "Bob has an acquaintance, whose subject is Carol, and acquired through Desmond", but addressing the "acquaintance" itself has no use at all: only its properties are important. `<http://imaginarypeople.com/Acquaintances/1>` would be, said so, a good candidate for replacement with a `BNode`. Blank nodes are represented in N3/Turtle using the `_` prefix (i.e. `_:acquaintance1`). However, one can use a special syntax when there's no need to refer to the `BNode` from more than one place in the graph. So, our example would turn into:

Listing 4 The graph from figure 4, represented as N3/Turtle, this time using a blank node for the "Acquaintance".

```
@prefix relationships: <http://imaginarypeople.com/Relationships#> .
@prefix people: <http://imaginarypeople.com/People/> .

people:Bob relationships:hasAcquaintance
                        [ a relationships:Acquaintance;
                          relationships:through people:Desmond;
                          relationships:subject people:Carol ].
```

The graph in listing 4 is much more readable than that of listing 3, but some information is implicit, namely the fact that the square brackets "return" a blank node, and all the statements inside them refer to this node.

Literals Until now, all the nodes from the example graphs were either URI references to concrete resources, or blank nodes. However, it is not rarely that one has to represent statements like "Bob is 23 years old" or even "Bob's name is Robert Smith". Actually, the use of people's names as part of the resource URIs is totally arbitrary: instead of `people:Bob`, `people:h342jad7` could be used. Being so, the resource URI is no more than a unique identifier, and, even if it follows a logic structure, it is not expected to convey any other information.
RDF allows the specification of "literals"[39, §6.5], strings that should be interpreted as actual values, instead of references. These literals can only be used as triple objects, and can be either "plain" or "typed". Plain literals are represented as strings, and have no interpretation associated with them: they're simply self-denoted. They can include, optionally, a language tag, useful for internationalization purposes.

Typed literals include a "datatype URI", the URI of the datatype that should be taken in account when interpreting the value. For instance, the string "30" for the property "weight" of a person has different values, depending on the numbering system that is being used (decimal, octal, hexadecimal...?), and even the unit type that the property is measured in (Kg, lb, ounce...?)[4]. The range of datatypes that can be used includes (but is not limited to) the XSD built-in datatypes[42, §3]. In N3/Turtle, the ^^ marker is used to separate a value from its datatype. Listing 5 shows some examples of possible literals.

Listing 5 Some examples of typed and untyped literals. Note how `xsd` is used as a prefix for the XSD namespace, and the different languages for the job description.

```
@prefix profile: <http://imaginarypeople.com/Profile#> .
@prefix people: <http://imaginarypeople.com/People/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

people:Bob profile:weight "75.0"^^<http://www.w3.org/2001/XMLSchema#float>;
           profile:age "25"^^<http://www.w3.org/2001/XMLSchema#int>;
           profile:name "Robert Smith";
           profile:job "Hairdresser"@en;
           profile:job "Coiffeur"@fr;
           profile:job "Cabeleireiro"@pt;
           profile:birthDate "1982-12-01"^^xsd:date.
```

Some other constructs are available in the RDF model, and the N3/Turtle languages for RDF serialization; however, they won't be described in this document. The reader is advised to consult [7] and [4] for an in-depth analysis of N3 and Turtle, respectively.

Other Notation Formats Up to now, only the N3/Turtle format has been used in this document. As it was already mentioned, N3 was built with readability in mind, since it provides a plain text, clear syntax description of an RDF graph. However, the most widely adopted format in the Semantic Web is still RDF/XML[3]. RDF/XML was the first RDF serialization format to emerge, encoding graphs using XML notation. It is not as intuitive as the N3/Turtle/N-Triples syntax, due to its XML-based nature, and, for that reason, it won't be used in an extensive way in the course of this document.

[4] In fact, the question of unit types is normally addressed using an extra blank node and property, as described in [46][§4.4], but the example is used as a way to reinforce the idea of context-sensitiveness.

However, it remains as the most widely used format in the Semantic Web, and definitely deserves a reference. Listing 6 shows how the previous example would look like in RDF/XML.

Listing 6 The same example, this time in RDF/XML.

```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF
  xmlns:profile='http://imaginarypeople.com/Profile#'
  xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
>
  <rdf:Description rdf:about="http://imaginarypeople.com/People/Bob">
   <profile:weight
    rdf:datatype="http://www.w3.org/2001/XMLSchema#float">75.0</profile:weight>
   <profile:age rdf:datatype="http://www.w3.org/2001/XMLSchema#int">25</profile:age>
   <profile:job xml:lang="en">Hairdresser</profile:job>
   <profile:job xml:lang="fr">Coiffeur</profile:job>
   <profile:job xml:lang="pt">Cabeleireiro</profile:job>
   <profile:name>Robert Smith</profile:name>
   <profile:birthDate
    rdf:datatype="http://www.w3.org/2001/XMLSchema#date">1982-12-01</profile:birthDate>
  </rdf:Description>
</rdf:RDF>
```

As it can be seen, some limitations exist, like that of being unable to use prefixes inside XML tags. For that and other reasons, RDF/XML loses to N3 on the field of simplicity.
Some other RDF serialization formats are gaining momentum at the time of the writing of this document. Maybe one of the best examples is RDF/JSON[19], that consists in encoding RDF graphs using the JSON (JavaScript Object Notation)[20] format. JSON is a subset of ECMAScript[24], and can be interpreted as Python code, as well, only with slight changes. So, it is a good alternative to XML as a serialization format, in the context of web applications, since it is much lighter, and easy to parse. RDF/JSON is still under development, but several Semantic Web libraries and applications already include support for this format, due to its simplicity in both generation and parsing.
RDFa[1] is another serialization format under development, with the support of the W3C, that aims to embed RDF data inside XHTML documents, without the need to create separate descriptions for the resources described in the web page. The objective is to take advantage of the fact that most web pages contain an inherent structure, and this structure can be translated

Listing 7 The same example, this time in RDF/JSON.

```
{
  "http:\/\/imaginarypeople.com\/People\/Bob": {
    "http:\/\/imaginarypeople.com\/Profile#weight": [
      {
        "value": "75.0",
        "type": "literal",
        "datatype": "http:\/\/www.w3.org\/2001\/XMLSchema#float"
      }
    ],
    "http:\/\/imaginarypeople.com\/Profile#age": [
      {
        "value": "25",
        "type": "literal",
        "datatype": "http:\/\/www.w3.org\/2001\/XMLSchema#int"
      }
    ],
    "http:\/\/imaginarypeople.com\/Profile#name": [
      {
        "value": "Robert Smith",
        "type": "literal"
      }
    ],
    "http:\/\/imaginarypeople.com\/Profile#job": [
      {
        "value": "Hairdresser",
        "type": "literal",
        "lang": "en"
      },
      {
        "value": "Coiffeur",
        "type": "literal",
        "lang": "fr"
      },
      {
        "value": "Cabeleireiro",
        "type": "literal",
        "lang": "pt"
      }
    ],
    "http:\/\/imaginarypeople.com\/Profile#birthDate": [
      {
        "value": "1982-12-01",
        "type": "literal",
        "datatype": "http:\/\/www.w3.org\/2001\/XMLSchema#date."
      }
    ]
  }
}
```

to the RDF model, though the use of annotations. This approach has great similarities with the one advocated by the Microformats[37] movement, but, instead of relying on specific semantic formats, weaves the more general RDF model with XHTML code. Since the work developed in the context of this document does not concern web page annotation, RDFa won't be explored in detail.

1.3.1 Ontologies

In the example graphs provided up to now, both the namespace `http://imaginarypeople.com/Relationships#` and `http://imaginarypeople.com/Profile#` have been used to group concepts that share a common field of application: for instance, the "Relationships" vocabulary includes concepts like the predicates "knows" and "loves", and the class "Acquaintance"; all these concepts share the fact that they target relationships among people, and provide specific constructs to address the description of such relationships.
According to Wikipedia[61]:

> *"An ontology is a representation of a set of concepts within a domain and the relationships between those concepts. It is used to reason about the properties of that domain, and may be used to define the domain."*

In a more practical sense:

> *"Artificial-intelligence and Web researchers have co-opted the term for their own jargon, and for them an ontology is a document or file that formally defines the relations among terms. The most typical kind of ontology for the Web has a taxonomy and a set of inference rules."* [6]

The "taxonomy" defines categories of objects, or "classes", and the relations among them (properties). The inference rules go further, by allowing the ontology designer to specify the building blocks that can be used for reasoning over such an ontology. The purpose of the work described by this document is to make data available as RDF; being so, it will concentrate on the taxonomic aspect of ontologies, and leave the reasoning part for future research.
The "Relationships" vocabulary, defined in the namespace `http://imaginarypeople.com/Relationships#` can be classified as an ontology for description of relationships, for the reasons that were mentioned in the previous paragraph.

However, as expected, it is not very useful for practical means, as there are already better ontologies for this kind of knowledge representation.

FOAF One of the most widespread ontologies in the Semantic Web is the FOAF[5] ("Friend Of A Friend") vocabulary. FOAF was developed by Dan Brickley and Libby Miller in 2000[13], and is currently the *de facto* standard for the specification of profile information. The specification of FOAF is defined at `http://xmlns.com/foaf/0.1/`, and this is, as well, the root of the FOAF namespace. Let's consider the example graph in listing 8, an excerpt of the author's FOAF profile[6]. Apart from the use of the comma operator in order to avoid repetition of the `foaf:interest` property, nothing new is presented in the example. The only difference is that the FOAF ontology really exists and includes a wide range of properties for a `foaf:Person`: gender, names, interests, websites, and even a "depiction".

Listing 8 An excerpt from the author's FOAF profile.

```
@prefix foaf:<http://xmlns.com/foaf/0.1/> .

  <http://zarquon.biz/~mahound/mahound.rdf#me> a foaf:Person ;
foaf:gender "Male" ;
foaf:name "Jose Pedro Ferreira" ;
foaf:givenname "Jose Pedro" ;
foaf:family_name "Ferreira" ;
foaf:interest "Artificial Intelligence","Artificial Life","Cinema","Literature",
                      "Music","Philosophy","Poetry","Programming","Technology" ;
foaf:depiction <http://www.zarquon.biz/~mahound/me_zurich_small.jpg> ;
foaf:homepage <http://mahound.zarquon.biz> ;
foaf:weblog <http://weblog.zarquon.biz> ;
foaf:workplaceHomepage <http://www.cern.ch> ;
foaf:workInfoHomepage <http://cdsware.cern.ch/indico> ;
foaf:schoolHomepage <http://www.fe.up.pt> .
```

However, FOAF owes its name to the possibility of linking people through "friendship" relationships. This is where the `foaf:knows`[13, #term_knows] predicate comes into play. This property establishes a mapping between two `foaf:Person` instances, and is interpreted as "subject knows object". As the FOAF specification document says, this "interaction relationship" doesn't have a specific degree of "strength": some people may interpret it as "having

[5] `http://www.foaf-project.org`

[6] `http://zarquon.biz/~mahound/mahound.rdf`

talked with X", and others as "being X's friend". For more specific "degrees of friendship", the "Relationship" ontology[35] extends FOAF, in order to allow such a differentiation.
Listing 9 exemplifies possible usages of the `foaf:knows` property. Both a specific resource URI and a blank node are used, both for the same end. The first statement can be read as "the person described by `http://zarquon.biz/~mahound/mahound.rdf#me` knows the one defined at `http://imaginarypeople.com/People/Bob`", whilst the second one can be interpreted as "the person described by `http://zarquon.biz/~mahound/mahound.rdf#me` knows another person whose name is John Doe". The latter is normally used in situation where the "object" has no known description available in RDF.

Listing 9 A possible excerpt from the author's FOAF profile, expressing the `foaf:knows` relationship.

```
<http://zarquon.biz/~mahound/mahound.rdf#me>
        a foaf:Person ;
foaf:knows <http://imaginarypeople.com/People/Bob>,
                    [ a foaf:Person ;
                      foaf:name "John Doe" ] .
```

But the real power of the RDF model resides in the ability to describe resources using a combination of different ontologies, thus enabling the reutilization of taxonomies and rules, and providing a virtually unlimited descriptive power. Listing 10 shows another excerpt from the author's FOAF profile, this time expressing the "based near" property, that specifies a geographical place where the subject is usually located. This "place" is described with the help of the WGS84 vocabulary. WGS84 is a standard for geodetic coordinates[62], while the WGS84 vocabulary[12] defines classes and properties that allow its description as RDF. One might question the usefulness of this data, as part of a personal profile document: latitude and longitude are not exactly the kind of information that one wants to provide in his/her profile, since, while extremely detailed, it has no immediate use for an human being. However the power of the web as a service platform makes it possible to translate this information into different kinds of representations. In figure 5, the author's foaf profile is shown, as rendered by the Zitgist Semantic Web browser[7], revealing a small map that locates the subject in a

[7] `http://dataviewer.zitgist.com/`

visual way. In listing 11, the result of a query to the GeoNames[8] database is shown, revealing the closest "named place" to the latitude/longitude coordinates specified by the same profile. Both representations are far more useful for a human that might be interested to consult the subject's profile. However, by specifying the latitude/longitude parameters and leaving their resolution to external agents, no information is lost, and its use by future, more complex, applications is protected.

Listing 10 An excerpt from the author's FOAF profile, expressing the `foaf:based_near` property.

```
@prefix geo: <http://www.w3.org/2003/01/geo/wgs84_pos#> .

<http://zarquon.biz/~mahound/mahound.rdf#me>
        a foaf:Person ;
        foaf:based_near [ a geo:Point ;
                    geo:lat "46.23198" ;
                    geo:long "6.04499" ] .
```

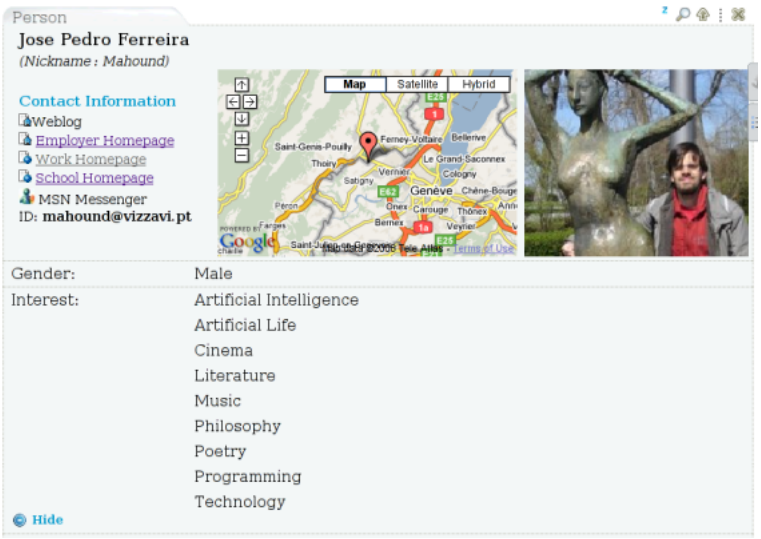

Figure 5: Zitgist rendering the author's FOAF profile

[8] `http://www.geonames.org/`

Listing 11 The result of a call to the GeoNames XML API, using the latitude/longitude as parameters.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<geonames>
  <geoname>
    <name>Cern</name>
    <lat>46.2333333</lat>
    <lng>6.05</lng>
    <geonameId>2661235</geonameId>
    <countryCode>CH</countryCode>
    <countryName>Switzerland</countryName>
    <fcl>S</fcl>
    <fcode>STNB</fcode>
    <distance>0.4137</distance>
  </geoname>
</geonames>
```

Other Ontologies Many other ontologies exist, for the most diverse uses. The process of ontology creation is open to anyone, and the only obstacle that a designer faces is that of convincing people to actually use his/her creation. Some famous ontologies include:

Dublin Core [45] – An RDF version of the Dublin Core[9] ontology for metadata about information resources. It is one of the basic ontologies, providing information about title, description, authorship, publishing information and other data about resources in general.

RDF Calendar [10] – An implementation of the iCalendar[22] standard in RDF: events, locations, dates and all the associated information.

RDF vCard [11] – An implementation of the vCard[21] standard in RDF. vCard is used for specification of information usually contained in business cards.

SIOC [12] (pronounced as "shock") – "Semantically-Interlinked Online Communities Project" – is a project that aims to interconnect communities such as blogs, bulletin boards, Usenet groups and mailing-lists. It

[9] http://dublincore.org/
[10] http://www.w3.org/TR/rdfcal/
[11] http://www.w3.org/TR/vcard-rdf
[12] http://sioc-project.org/

defines an ontology that allows the specification of elements such as "forum posts", "user groups" and "sites", and the connections between them.

DOAP [13] – "Description of a Project" – a vocabulary that describes open source projects. It allows specification of information related to the maintainers of the project, releases, licenses, version control repositories, besides all the profile information (description, operating systems, programming languages...) associated with a software project. The DOAPSpace Project[14] provides a translation service that makes meta-data on SourceForge[15] projects available as DOAP[16].

DOAC [17] – "Description of a Career" – an ontology that allows the specification of the professional capabilities of an individual, much like in a CV/Resume. It was designed to be compatible with the Europass[18] CV model. Language skills, education, work experience and computer skills are some examples of concepts addressed by this ontology.

SKOS [19] – "Simple Knowledge Organization System" – SKOS locates itself very close to the the concept of "meta-ontology": it defines concepts that can be interrelated and grouped, much like in a thesaurus or a taxonomy. Contrarily to an ontology, it doesn't define any axioms or facts, but rather establishes relations of hierarchy between concepts. However, some research has been happening on the problem of translating SKOS to Semantic Web ontologies[32].

MO [20] – "Music Ontology" – this ontology provides three levels of detail for the description of musical works. It can represent artists, tracks, albums and storage supports (digital formats and physical supports), as well as composers, genres, lyrics, scores and performers. The "deeper" level of detail allows musical event decomposition (i.e. "At 3:27 of «My Funny Valentine» from Chet Baker - «The Last Great Concert», Chet Baker starts singing"). This ontology is very extensive, providing a wide range of possible applications for the future of digital music.

[13] `http://usefulinc.com/doap/`
[14] `http://doapspace.org/`
[15] `http://www.sourceforge.net/`
[16] `http://doapspace.org/sourceforge/`
[17] `http://ramonantonio.net/doac/`
[18] `http://europass.cedefop.europa.eu/`
[19] `http://www.dbpedia.org`
[20] `http://musicontology.com/`

Ontology Languages The ontologies that have been presented in this document have been provided as abstract definitions of taxonomies and rules. The only provided "specifications" of these ontologies have been the different documents that describe each class and property, and relate them with "real world" entities. However, ontologies themselves are supposed to be serialized in some way, so that machines are capable of interpreting and reasoning over them. Only by knowing the domain and range of a property (i.e. `foaf:knows` requires two `foaf:Persons`), and the class hierarchy (`foaf:Person` is a subclass of `foaf:Agent`) a machine will be able to decide if a concrete RDF graph is valid or not, and, in the affirmative case, answer to queries like "is this person a `foaf:Agent`?". That said, being able to write down, in a universal language, all that is known about an ontology proved to be an essential step not only in the process of graph validation, but also in opening the way for reasoning mechanisms to operate.
The first approaches to the problem of describing web ontologies came through SHOE and XOL. The former was an HTML-based language, whilst the latter would later evolve into OIL[25]. OIL was the first ontology language that was built over the RDF/RDFS schemas. The RDF Schema language (RDFS) provides the basic building blocks of ontologies, such as Classes, and relations that can be established between classes (inheritance) or between a class and a property (range and domain, much like in mathematical functions).

OWL OWL (Web Ontology Language)[58] was made a W3C recommendation in 2004, replacing DAML+OIL as the primary language for ontologies. It is largely a revision DAML+OIL, using the RDFS language, as well, as a basis. OWL is made available in three "flavors":

- OWL Lite - Supports taxonomies and simple constraints. It was introduced so that it could be easily implemented (due to its limited amount of features), and targeted mainly existing classification hierarchies that needed to be translated;
- OWL DL - Provides full expressiveness, but within the limits of Description Logics. OWL DL guarantees computational completeness and decidability. It is perhaps the most interesting of the three idioms, for its machine-friendliness;
- OWL Full - Enables full expressiveness, with no boundaries except those of OWL itself;

OWL lets the ontology creators define classes, properties and individuals (instances of classes). In addition to that, constraints can be added to properties, in terms of cardinality and possible values. Properties can be as well specified to hold qualities that will make reasoning possible (i.e. property "hasChild" can be declared the inverse of property "hasParent")[54, §3.3].
Ontology languages won't be explored in further detail through this document. The logic that underlies them has, for itself, enough to say for several other thesis. The reader should consult [58] and [54] for further details.

1.3.2 Query Languages

RDF parsing is not enough for satisfying the needs of complex applications. Searching for "all the acquaintance of a `foaf:Person`" or all the occurrences of any other property seems like an easy operation to perform: it is a matter of cycling through all the graph's triples and comparing values. However, getting a list of "names of Bob's friends", or even "names of people that were born in Porto" would be slightly more complex operations. As for the latter, one would have to search for the resource that would represent a city with the name "Porto", and then, for every person possible, see if there was an arc "born in" connecting them. In the affirmative case, one would have to extract the name associated with that "person resource" (this would be, of course, assuming that the "born in" arcs would connect people to cities, and not vice-versa). As the complexity of the queries increases, so does the amount of code required to execute them. This is clearly the kind of work that an interpreted RDF querying language would do, without the need for writing extensive amounts of code, and keeping the focus on data, instead of database structures.
According to [28], there are three main "families" of RDF Querying languages[28, §4]:

Relational Query Languages – i.e. SPARQL[53] and RQL[36] – Much based on relational databases, and SQL syntax. SPARQL is now a W3C recommendation, in addition to being the *de facto* standard for the Semantic Web. SPARQL incorporates itself elements from RQL, but is mainly based on the SquisQL and RDQL languages;

Reactive Query Languages – Algae[52] – This family of languages is closely related to the previous one, but allows the specification of ECA (Event-Condition-Action) rules;

Navigational Access Query Languages (or Path-based) – Versa[49] – This family of languages relies on query through access to the RDF

"hierarchy", in a way that resembles functional programming languages. Versa adopted the philosophy of XPath[23] as a way to access graph elements.

SPARQL After some years without an "official" language for RDF querying, SPARQL was finally recommended by W3C, in January 2008. In the words the inventor of the World Wide Web and Semantic Web Architect Tim Berners-Lee, "trying to use the Semantic Web without SPARQL is like trying to use a relational database without SQL"[59]. In fact, SPARQL (and its less used counterparts) substantially improved the quality of life of the Semantic Web developers, allowing them to query RDF graphs in the same way they would query a regular database, and opening the way for complex applications.
A typical SPARQL query resembles a SQL query, in the sense that it follows the `SELECT ... FROM ... WHERE ...` structure. Since the `FROM` part is used only in specific situations[53], one can say that the most common kind of SPARQL query has the form:

`SELECT` *`[variables]`* `WHERE` *`[constraints]`*.

The "variables" list contains the variables that will be returned in the end (bound to some value, for each row), like in SQL. The "constraints" part contains a list of triples, encoded in an N3-like format, that can include references to variables. For instance, in order to obtain, from the DBpedia [21] data set, all the people that were born in Porto, one would do as in listing 12.

Listing 12 SPARQL query that gets all the names of people that were born in a city whose name is "Porto", from DBpedia.

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX p: <http://dbpedia.org/property/>
SELECT ?name WHERE {
  ?city rdfs:label "Porto"@pt.
  ?person p:birthPlace ?city;
          rdfs:label ?name.
  }
```

A segment of the results of the query from listing 12, when applied over DBpedia's SPARQL endpoint[22], is represented in figure 6.

[21] see §1.4
[22] `http://www.dbpedia.org/sparql/`

```
Almeida Garrett
Eduardo Souto de Moura
Francisco Sá Carneiro
Jorge Costa
ジョルジュ・コスタ"
José Manuel Barroso
José Manuel Durão Barroso
Дурау Баррозу, Жозе Мануэл
ジョゼ・マヌエル・ドゥラン・バローゾ
Durão Barroso
若泽·曼努埃尔·巴罗佐
João Vieira Pinto
João Pinto
Jorge Nuno Pinto da Costa
Manoel de Oliveira
Оливейра, Мануэл де
マノエル・デ・オリヴェイラ
Nuno Gomes
```

Figure 6: Some query results. Each name appears once per "translation". Duplicate entries have been manually removed.

Like N3/Turtle, SPARQL allows the specification of namespace prefixes, using the `PREFIX` keyword.
In figure 7, the constraints inside the `WHERE` block are displayed, as well as one of the sub-graphs that match them.

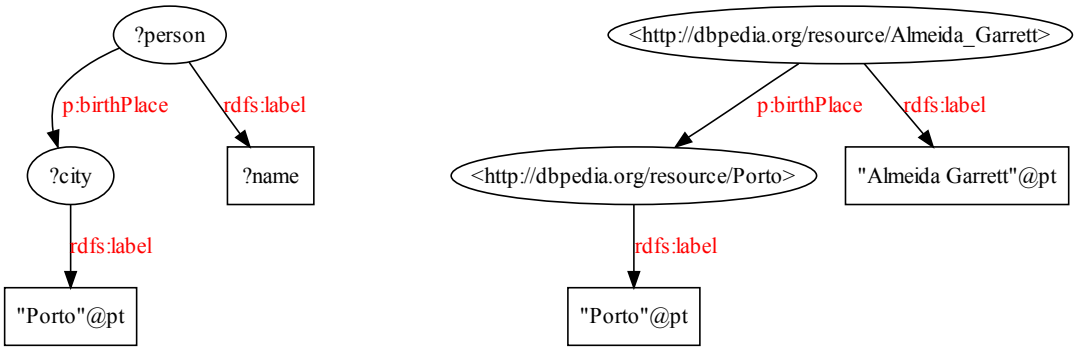

Figure 7: On the left side, the constraint from the SPARQL query represented as a graph. On the right side, a possible match for the pattern.

SPARQL queries are supposed not only to be applied over "local" triplestores,

but to be transmitted through the web, as a way to query remote repositories (i.e. the previous example was only possible due to this "remote querying" method). The "SPARQL Protocol for RDF"[17] defines ways to do so using both plain HTTP and SOAP. For its easiness of use (and implementation), the HTTP interface is the most adopted by service providers. It basically requires that the SPARQL query be properly URL-encoded, and passed in an HTTP POST/GET parameter called "query". Some other parameters, related to named graphs, can be provided, but these can be consulted in detail in [17]. A remote service that accepts SPARQL queries and returns their results is normally called a "SPARQL Endpoint". A SPARQL endpoint usually has a URI, like `http://dbpedia.org/sparql` (the endpoint for DBpedia), that works as the request handler. For instance, a request to the URL:

```
http://dbpedia.org/sparql?query=SELECT%20?s%20?p%20?
o%20WHERE%20\protect\T1\textbraceleft%20?s%20?p%20?o.%20\
protect\T1\textbraceright%20LIMIT%2010
```

will execute the query:

```
SELECT ?s ?p ?o WHERE  ?s ?p ?o.    LIMIT 10
```

returning ten triples from the graph (the first ten that the triple store picks). SPARQL results can be made available in several different formats. The standard is to make them available in the SPARQL Query Results XML Format[14], a W3C recommendation that was issued together with the language specification and the protocol. However, as always, an alternative JSON-based format[18] exists, that is being widely adopted by content producers, in spite of not being (yet) a recommendation.

1.3.3 Describing Graphs

Sometimes it is useful to provide a description of a resource that is a graph itself. An example is provided by listing 13: the graph that describes the author's profile is used as just another resource, and a relation of authorship is declared, through `dc:creator`, with the resource that represents him as the object.
Listing 14 shows another way to do this using the N3 notation. It is important to notice that Turtle does not support this "graph quoting" feature, as it goes beyond the scope of the RDF model. The technique consists in creating an "unnamed graph", by using the curly brackets operators, and

Listing 13 An N3/Turtle-serialized graph that describes the graph that contains the author's FOAF profile.

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .
<http://zarquon.biz/~mahound/mahound.rdf>
          dc:creator <http://zarquon.biz/~mahound/mahound.rdf#me>.
```

quoting it as if it were a regular resource. This is called a "formula" in N3. One possible use for this kind of constructs, that Berners-Lee advocates in [7], is the classification of information in terms, for instance, of truth value:

```
{<http://zarquon.biz/~mahound/mahound.rdf#me>
    foaf:knows [ foaf:name "Kevin Bacon" ].}
                                  a n3:falsehood.
```

means that the author does not know anyone whose name is "Kevin Bacon" (and, therefore, doesn't know the famous actor with the same name). This capability of N3 has not been widely adopted, as it goes into "unstable territory", outside the boundaries of the RDF specification. Independently from that, some other approaches to the revocation of RDF information are being developed[47].
There is a group of people inside the Semantic Web Interest Group of the W3C, working on the topic of named graphs[23]. A notation language called TriG[11], an extension of Turtle, is being developed, allowing the definition of such graphs in an intuitive way, that resembles the syntax for anonymous graphs for N3.

Listing 14 An N3-serialized graph and contained sub-graph.

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix foaf:<http://xmlns.com/foaf/0.1/> .
{
<http://zarquon.biz/~mahound/mahound.rdf#me>
        foaf:mbox <mailto:pedro.ferreira@fe.up.pt>.
} dc:creator <http://zarquon.biz/~mahound/mahound.rdf#me>.
```

1.4 Linked Data

The vision of a Semantic Web requires that resources do not exist in isolation, but rather link to other resources in a useful way. It is easy to establish

[23] http://www.w3.org/2004/03/trix/

a parallel with the "traditional" web: links and anchors become properties and URI references, - an HTML link to a favorite music band's web page becomes an "arc" between a FOAF profile and a resource describing the same band, through the use of a proper ontology. There's a clear difference here, introduced by the machine-understandable nature of the RDF model: a link from a web page cannot be seen out of its context - a crawler will hardly understand the difference between "this band is very bad, don't listen to them" or "this band is one of my favorites". And this is only one of the problems, since the crawler will probably never be aware that the resource relates to a band, and that this band has a particular name, and belongs to a music style, and so on. This is just one of the reasons why the linking potential of the Semantic Web is so big when compared to its traditional counterpart: the dissipation of knowledge is minimized, since the impact of important problems like that of natural language processing is reduced. Knowledge connects naturally, without the need for computationally expensive indexing processes, or even (in some situations) human intervention.
Tim Berners-Lee has defined four basic principles that should guide the process of content creation both in the "Traditional" and Semantic Web[9]:

1. Using URIs as names for thing – something that is encouraged by the RDF model;

2. Using HTTP URIs, so that it is possible to look them up – not enforced by RDF, but a good practice;

3. Providing useful information, when someone looks up an URL – an obvious requirement, but not always applied. [9] refers to the example of major Social Networks, that do not export their information in standard form;

4. Including links to other URIs, to facilitate discovery – Links are a measure of a page's value in the WWW[24], and the Semantic Web is no exception: as [9] says: "The value of your own information is very much a function of what it links to, as well as the inherent value of the information within the web page."

A linked Semantic Web will be, as well, a web of integrated services, where data from different sources can be connected, and shown to the user in a more friendly/pleasant way. There are already some applications, out in the web, performing this kind of operation. The concept of a "mashup" is already

[24]This measure is applied, for instance, by the Google PageRank algorithm[15]

widespread through the web, and even if the employed technologies are not, most of the time, based on RDF, the concepts are very approximate. Here are some examples of "mashup applications":

Fundrace 2008 [25] – A mashup, created for the US Presidential Campaign 2008, that uses Google Maps and data from fund donors, in order to display the geographical localization of each donor, as well as the destination and monetary value of his/her donation.

FriendFeed [26] – Allows the user to keep track of the actions performed by his/her friends in different social networking sites.

Yahoo! Pipes [27] – Not a "mashup", but rather a "mashup building tool". *Yahoo! Pipes* allows a user to combine different data sources, and produce a feed that outputs "mashed up" data. Several data sources are supported: i.e. flickr, Google APIs, CSV files or RSS feeds.

The main difference between a "Web 2.0 mashup" and a Linked Data application is the API question. While, in an RDF-based model, information would be available through the use of different ontologies, but always organized according to the RDF specification, in a "Web 2.0 mashup", different APIs have to be used, in order to communicate with the different data sources. Each service has its own API, that may involve different networking protocols and data formats. There is still (at the time of the writing of this document) no common approach to the problem of service interoperability, from the main players in the field of web data, though some efforts are in course[28].

Linking Open Data The "Open Data" movement is a phenomenon that has grown during the last decades. Scientific progress has taken the western civilization to a point where some ideas that were taken for granted by the average citizen ask for a deeper look:

> Genetics are unfolding a whole new world of knowledge about the human body, that some believe that has to be claimed by the human race as collective property;

[25] `http://fundrace.huffingtonpost.com/`
[26] `http://friendfeed.com/`
[27] `http://pipes.yahoo.com/`
[28] In the field of Social Networks, Google's OpenSocial is one of them – `http://code.google.com/apis/opensocial/`

As the number of cameras multiplies, mapping the earth from the sky, and replicating the number of private photos available on-line, the questions of ownership and privacy rise in complexity;

There's a rising pressure for the results of projects that are funded by the government and associated institutes to be freely accessible: after all, everybody is supposed to benefit from the money that comes from taxes;

The amount of initiatives that aim to produce knowledge that is, from the beginning, aimed to be freely available, is currently visible on the web: Wikipedia[29], ibiblio[30], Project Gutenberg[31] and Project Mutopia[32] are just a few examples;

In addition to the naturally free repositories of information, some for-profit services are providing as well some of their data for free on the web, either as part of their business strategy, or because it is a requirement for such a service to work. An example of the former is the Magnatune[33] music label, that provides complete metadata about all the artists, the albums they've produced, and the respective tracks, as well as low-quality versions of the audio (clearly a "try before buying" business model); the flickr[34] photo sharing service is an example of the latter, providing both free and paid accounts (for extra storage), but making the content available for the general public (as it is expected from a photo sharing service). Both services provide APIs that enable external software to harvest metadata, or even buy content (as it is the case with Magnatune).
The Linking Open Data Initiative (LOD) stems from the Semantic Web Education and Outreach (SWEO) Interest Group[35] (from the W3C), and has the main objective of making all the existing open data sets available as Linked Data. That implies providing them as RDF, and linking them properly. On October 2007, the LOD data sets summed an impressive amount of over two billion triples, interlinked by three million RDF links[36]. The data

[29] http://www.wikipedia.org/
[30] http://www.ibiblio.org/
[31] http://www.gutenberg.org/
[32] http://www.mutopiaproject.org/
[33] http://www.magnatune.com/
[34] http://www.flickr.com
[35] http://www.w3.org/2001/sw/sweo/
[36] See http://esw.w3.org/topic/SweoIG/TaskForces/CommunityProjects/LinkingOpenData

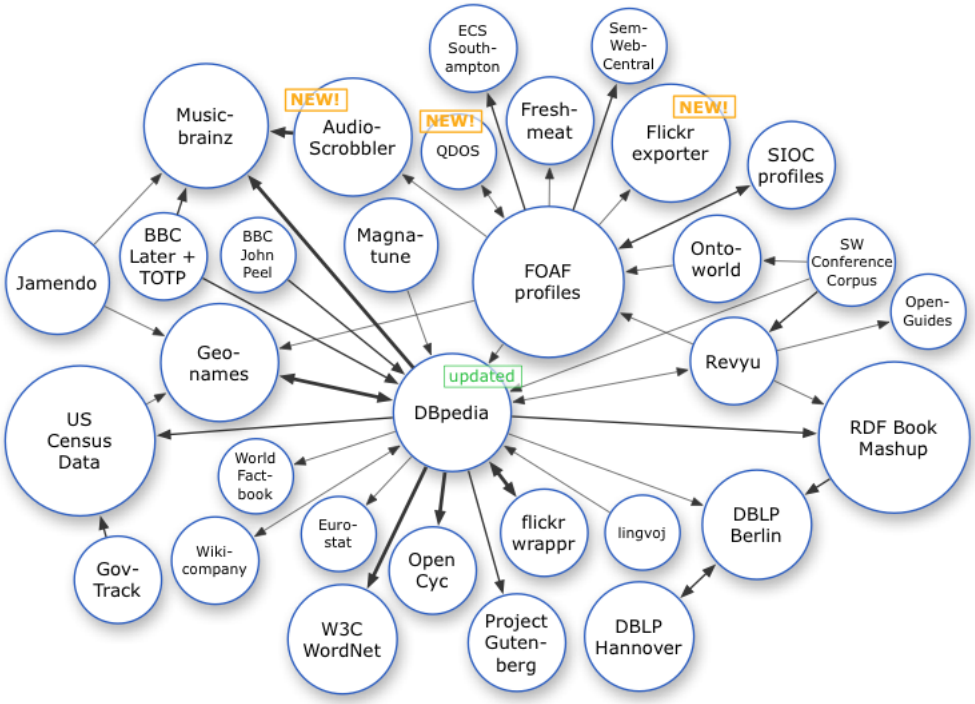

Figure 8: The "Linking Open Data Cloud", as of March 2008.

sources that integrate this project, as well as their interconnections, can be seen in fig. 8. Here are some examples:

DBpedia [37] – An RDF version of Wikipedia, maintained by developers from the University of Leipzig[38], the Freie Universität Berlin[39], and from OpenLink Software[40]. Both access through URI references and and SPARQL endpoint are provided.

DBTune [41] – A service, maintained by Yves Raymond[42] (a PhD Student from the Centre for Digital Music, Queen Mary, University of London), that provides (among others) RDF interfaces for: the Magnatune label

[37] `http://www.dbpedia.org`
[38] `http://www.uni-leipzig.de`
[39] `http://www.fu-berlin.de/en/`
[40] `http://www.openlinksw.com/`
[41] `http://dbtune.org`
[42] `http://moustaki.org`

metadata repository (SPARQL Endpoint available), profiles from the MySpace social network and metadata from the last.fm[43] service, that stores a log of the songs that were listened by a particular user.

GeoNames [44] – GeoNames provides an ontology for the description of geographical locations, and a huge data set with RDF resources for each town, region or country known by the service.

RDF Book Mashup [45] – This service provides an RDF interface that joins information from both the Amazon and Google book databases, providing resources that represent authors and publications, as well as information on pricing.

lingvoj [46] – A repository that contains RDF descriptions of 507 languages (at the moment of the writing of this article).

The RDF data generated from non-RDF sources is created through different processes. While DBpedia, for instance, relies on the direct conversion of Wikipedia data from a database replica, DBTune relies mostly on "HTML scraping"[47] for its services. The RDF Book Mashup stands at an intermediate level, converting data already available from the Amazon and Google Base APIs, provided as XML.

Smushing "Smushing" is a term that entered the Semantic Web jargon, and that basically means the process of merging data, based on unique identifiers. It is definitely an essential concept for the integration of data from different sources. For a FOAF profile, several fields can be a unique identifier[48]: the e-mail address, the e-mail SHA1 sum, or even social network accounts. On the other hand, in the absence of such a unique identifier, a set of non-unique identifiers can be used instead (with a non-null probability of error, of course). Smushing can be performed at several degrees:

- Explicit equality - The `owl:sameAs`[58, §3.2] property can be used to identify a resource as being identical to another one. This is the best case scenario, where resources are clearly identified as being the same.

[43] `http://last.fm`
[44] `http://www.geonames.org/ontology/`
[45] `http://sites.wiwiss.fu-berlin.de/suhl/bizer/bookmashup/`
[46] `http://www.lingvoj.org`
[47] "scraping" - extracting data from the output, in this case, from an HTML page.
[48] For more on FOAF and identity, see `http://rdfweb.org/mt/foaflog/archives/000039.html`

No real "smushing" is needed in this case, if the triplestore is "smart" enough to process `owl:sameAs` relationships;

- Inverse-functional properties - There are properties that uniquely identify an object: these are Inverse Functional Properties[58, §3.3]. A property p is an IFP if considering triplestore T, and its subset Tp (all the triples that contain p as a predicate) it accomplishes:

$$\forall_{o \in obj(T_p)} \exists^1_{s \in subj(T_p)} : < s, p, o > \in T \quad (1)$$

(note: $subj(A)$ means the set of triple subjects from triplestore A, and $obj(A)$ the set of triple objects from triplestore A.)

In other words, two different resources cannot have the same value for an IFP (or else the triplestore is inconsistent). Some examples of IFPs are:

  - Mailboxes and their SHA1 hashes - since most people don't want to release their e-mail addresses in plain, readable text, it is common to encrypt them using a digest algorithm, thus keeping their uniqueness (and comparability), but scrambling their meaning;
  - Web Accounts (social networks et al.);
  - Homepage addresses;
  - URIs of RDF resources - a resource has only one URI, except when `owl:sameAs` is used to state otherwise;

Some other properties will certainly not work:

  - Names;
  - Addresses;
  - Phone numbers;
  - DNA checksums[49] - Identical twins share the same DNA;

- Heuristic Smushing - When no IFPs are available, criteria based on heuristics have to be used, in order to try to estimate a value that quantifies the certainty that two resources are the same. There's some work going on the field of the definition of "rough" ontologies, with "approximate" concepts, and [38] illustrates its application to instance unification;

[49] DNA checksums are certainly science-fiction, but the FOAF specification mentions them as a "joke", and an example of a possible identifier[13, #term_dnaChecksum]

"Smushing" is an important operation for the Semantic Web of Linked Data, since it allows data from different sources to be integrated in an healthy way, preserving identity and avoiding replicas that could cause confusion.

2 Problem Description

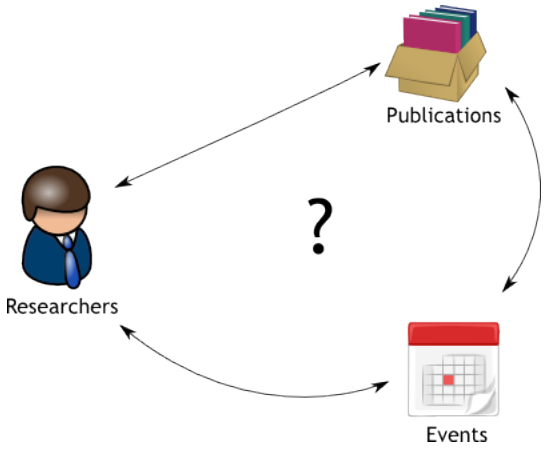

Figure 9: The three main "players" in Scientific Social Networks.

From the beginning, this document placed its focus on:

- Searching for a solution that would allow the construction of a world-wide Scientific-oriented Social Web, over the Semantic Web, through the implementation of a specific architecture;
- Searching for solutions for the problem of data availability - most data are still not available as RDF, and cannot be incorporated in such a network;

The solution for the first problem is highly dependent on the developments that occur on the second one: there's no way anyone can aspire to unify such diverse information, without good means of transforming it into a common model (in this case, RDF). Figure 9 shows a simple representation of the problem of "Social Research Networks": Researchers, Events and Publications are the vertexes of this "triangle", and they have to be connected somehow. The definition of these three "players" as the primordial entities in this problem doesn't come by chance:

- Researchers - Users are essential for any Social Network - if there are no users, there's no "social";
- Events - Events have become essential in the panorama of Web-based Social Networks, independently of their focus: almost every large-scale

social network provides the ability to create events and invite other users to attend them. For the Scientific Community, this has particular importance, since conferences, workshops, symposiums and seminars consume a considerable amount of time from the researchers, and, in the past, acted as the single point of social interchange between them, and a way to stay up-to-date with the last developments in the community;

- Publications - Every Web-based Social Network involves content, being it messages, forum posts, movie reviews or pictures. In the case of the Scientific Community, publications have always been the standard for information sharing: scientific papers are published in journals and conference/workshop/symposium proceedings, integrating a worldwide-distributed library. Publications are as well a social instrument in the Scientific Community, since multiple researchers can collaborate in the writing of a paper (this topic will be expanded later in the text).

The main questions that arise are:

- How to effectively connect all these "dots" using the Semantic Web, and knowing beforehand that they're spread through different places, and represented in different formats?
- How to organize the flow of information in such a way that knowledge doesn't stay dispersed, but remains linked and consistent, and, above all, easy to browse by the end-user?
- How to provide all the information (in RDF) that such a an ambitious construction would require?

In order to find a solution to this questions, one must start by understanding how Research Networks work, and how they integrate with the "current web".

2.1 The Challenge of Research Networks

The Scientific Community is conjectured to be highly interlinked, by nature[2][§1]. In 1969, Casper Goffman introduced the concept of "Erdös Number", honouring the famous Hungarian mathematician and graph theory guru Paul Erdös. The Erdös number measures the distance between a subject and Paul Erdös himself, using paper co-authorship as the metric. Assuming that Erdös had an Erdös number of zero ($E = 0$), and a person

that co-authored a paper with him has $E = 1$, and a person that co-authored a paper with someone that co-authored a paper with Erdös has $E = 2$, and so on... one will find out that the graph of mathematicians is so rich in its connectedness, that, in average, each researcher that has co-authored at least one paper and has a finite Erdös number (doesn't live in a "graph island") is only 4.65 steps away from Erdös[30]. This mathematical joke shows, in a simple way, how dense the graph of collaborations is. In fact, the most optimistic interpretations of the so-called "small world phenomenon", initially introduced by Milgram[57], claims that one can get to virtually anyone in the world, through acquaintance relationships, with no more that six steps. In spite of the high degree of connectedness between the members of the scientific community, the web doesn't seem to be working as it should in bringing these people together, and, above all, providing a browse-able graph of authors, publications, and associated scientific events. Some of the reasons were already discussed in §1.4, and they apply to social networks in general:

- The "big graph" is not a connected graph - users are spread through small "graph islands", and these islands do not allow connections between them;
- Contents are not available in a standard format. In the scientific field, during the last decade, the Open Archive Initiative has striven in minimizing this problem, but some problems remain;
- There's an enormous amount of replicated and inconsistent content: for instance, authors exist in different (or even the same) repositories, with slightly different names, and papers exist in different versions through different places. The same applies for general-purpose web-based social networks: many people belong to more than one social network;
- As a consequence of the previous point, an author loses control over his/her contents: it is inconceivable having to change some piece of information in ten different places, each time a modification is needed. In the context of social networks, that's even worse, because people like to keep track of the movies they see, books they read, and there's an overwhelming supply in terms of services, in this field, and no way to transfer content between them;

The lack of defined standards is not the only culprit to blame in this problem. One should not forget that knowledge is valuable, and, essentially in

general-purpose social networks, the companies that provide these services earn money by taking people to their sites. Providing a "migration"/sharing scheme would encourage users to spend more time in other social networks. There's, as well, the legal problems of information disclosure (that can be avoided if the user is allowed to set privacy options), and the financial value of contacts and statistic data. However, the open nature of the web has, for many times, proved to persist as the main driving force in its development: efforts like Wikipedia, and businesses like YouTube[50] and flickr have proved to be successful platforms for content sharing, and both provide APIs for interfacing with external applications. The FOAF Project tried to set a standard that could do the same with Social Networks (even before the most significant ones, such as Facebook[51] and Orkut[52] appeared), but was, until now, successful only to a very limited degree. Even if the "greedy" vision of "the largest social network ever" still persists in the mind of the main players, it is fair to place the question about whether this will ultimately generate a win/lose situation for competitors and the ultimate absorption of smaller social networks by the "giants", or simply take the market to a point of stagnation where users will get fed of maintaining three or four redundant profiles and simply drop them for something else simpler. It would certainly be an advantage for the Linked Data community if "commercial" social networks could keep the burden of storing such a huge amount of information, while allowing information to easily flow in and out. However, that situation seems to be still quite far away (though some experiments on common APIs start to emerge[53]).

Publications Scientific publications are distributed through the web, in different kinds of repositories:

- Institutional repositories – from universities, laboratories and publishers, like CERN's CDS[54] or ACM's digital library[55];
- Repositories on particular subjects – arXiv[56] and SPIRES[57], for instance, contain only texts in the field of mathematics and physics,

[50] `http://www.youtube.com`
[51] `http://www.facebook.com`
[52] `http://www.orkut.com`
[53] `http://code.google.com/apis/opensocial/`
[54] `http://cds.cern.ch/`
[55] `http://portal.acm.org/dl.cfm`
[56] `http://arxiv.org/`
[57] `http://www.slac.stanford.edu/spires/`

while CiteSeer[58] and DBLP[59] focus more on Computer Science;

- Nation-wide repositories (i.e. from libraries) – like the Digital Portuguese National Library[60], that allows the storage of thesis from Portuguese authors, on the most diverse subjects.

In addition to all these services, there are "aggregators" that "crawl" the repositories and index the documents, providing search capabilities over a large database of texts. An example of such an aggregator is the Google Scholar[61] service.

Events Information on Academic Events is, like Publications, spread through different repositories, of different natures, on the web. However, the "contents" of a conference are rarely centralized in a single place:

- E-mails announcing call for papers (CFPs) circulate through the main mailing lists on the different scientific subjects;
- Websites for conferences are set up, some of them using automated solutions (like CERN Indico[62]), and others, custom web pages;
- Some automated systems are used for paper submission/reviewing and publication of proceedings (i.e. EasyChair and Indico);
- The accepted papers and some other content are available either in normal web pages, conference management systems like Indico or EasyChair[63], or even some specific applications like SISSA's PoS (Proceedings of Science), in addition to the publication repositories that were already mentioned;

Researchers Researchers, as web users, are everywhere: they have e-mail accounts, personal web pages, and they're probably registered in dozens of web-based services, including social networks, conference management systems and publication repositories. Their identity is replicated over the web, bearing different names and other inconsistent information. The example of Tim Berners-Lee is clear: CiteSeer refers to him as Tim Berners-lee, T.

[58] http://citeseer.ist.psu.edu/
[59] http://dblp.uni-trier.de/
[60] http://bnd.bn.pt/
[61] http://scholar.google.com
[62] http://indico.cern.ch
[63] http://www.easychair.org/

Berners-lee and T. J. Berners-lee. It is impossible, relying only on the author's name, to read "T. Berners-lee" and distinguish "Tim Berners-lee" from an hypothetical "Tom Berners-lee". Record linkage is not a new problem, and it has been studied by many as a tool for minimizing redundant entries in data sets in the order of millions of entries[63]. However, the web was born in an era when people were already aware of the problems that come with redundancy and inconsistency - database designers just ignored the fact that unambiguous identification of authors would be useful in a near future. This problem makes life much harder for those who advocate a Semantic Web ruled by the principles of Linked Data, but some years from now, will hopefully be gone (assuming that the WWW will stick with the ideals of the Semantic Web and Linked Data).
Conventional web-based Social Networks can be seen as the main repositories of information about people. On the other hand, their closed nature (in terms of making user content available to the exterior) keeps most of them from being an useful instrument for this case. However, some (fortunate) exceptions exist:

- hi5[64], one of the biggest social networks on the Web (around 98 million users[65]) provides some of the user profile information as FOAF. The support is still very basic, but it is, without any doubt, a very important step;
- LiveJournal provides approximately the same quality of FOAF information as hi5, providing profiles for around 15 million users[66];
- MySpace, the biggest social network on the web, does not provide FOAF profiles as part of its services, but there is a third party service that does "on the fly" translation[67].

More examples of WWW Social Networks that provide user profile information as FOAF can be consulted at the W3C Semantic Web Interest Group Wiki[68].
Figure 10 shows one of the many possible perspectives on the world of Scientific Social Networks. Some of the relations that show up in the graph could be easily replaced by their inverse: for instance, one could say that

[64] http://www.hi5.com
[65] http://www.pipl.com/statistics/social-networks/size-growth/
[66] http://blogs.sun.com/bblfish/entry/15_million_foaf_files
[67] http://dbtune.org/myspace/
[68] http://esw.w3.org/topic/FoafSites

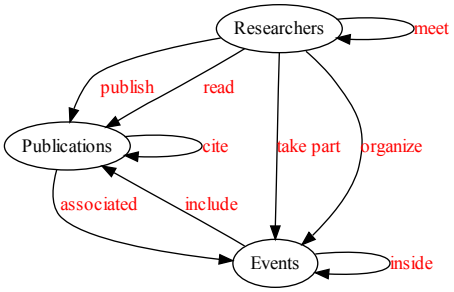

Figure 10: The "Researchers-Publications-Events graph".

"Publications are made by Researchers", instead of "Researchers publish Publications", or "Scientific Events are organized by Researchers", instead of "Researchers organize Scientific Events". However, since the "Researchers" are the purpose of the whole social network, and, consequently, the most important vertex of this "triangle", depicting them as the subjects of the relations seems much more obvious. It is good to mention, though, that the orientation of these "graph arcs" depends highly on the mechanisms of implementation. The graph of fig. 11 depicts what could be seen as an "ideal" representation of a small scientific network, in a technology-independent view. Several relations are expressed explicitly: acquaintance, authorship and participation; and others are presented in an implicit way: co-authorship is perhaps the most important of them.

2.2 Publications in the Semantic Web

Publishing information about academic papers and other publications through the Semantic Web is definitely not a new idea. In 2004, the Semantic Web Cluster from DERI Galway (Ireland) made all the information contained in DBLP available as an RDF dump[69]. The result was produced through the conversion of XML dumps from DBLP[70] to RDF/XML (using a specially developed ontology[71]), employing an XSLT stylesheet. CiteSeer followed in 2005, using a similar process. This time, the data was available from CiteSeer itself, through OAI-PMH, a protocol for Metadata Harvest-

[69] http://sw.deri.org/~aharth/2004/07/dblp/
[70] http://dblp.uni-trier.de/xml/
[71] http://sw.deri.org/~aharth/2004/07/dblp/dblp.html

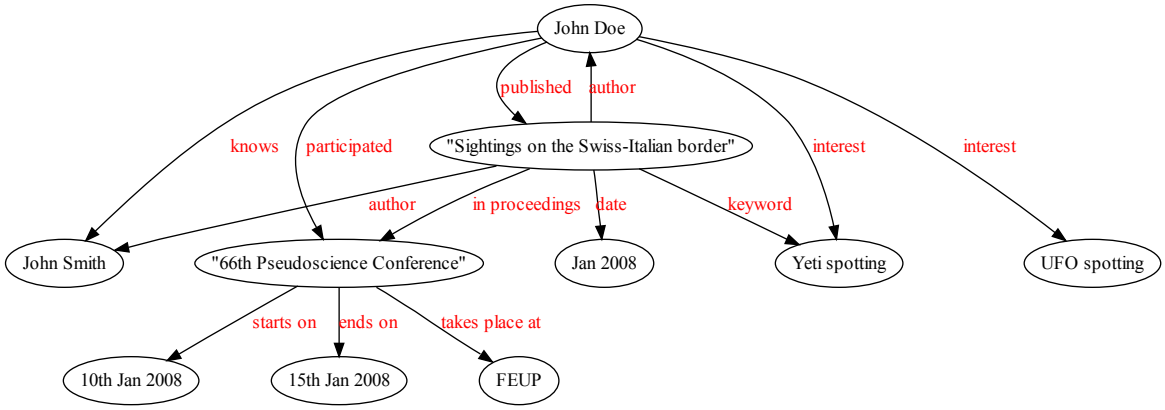

Figure 11: An excerpt of an "ideal" graph of a scientific network. The attributes relative to "John Smith" are not shown for the sake of simplicity.

ing developed by the Open Archives Initiative, that became a standard for on-line repositories. Once again, an XSLT stylesheet was used for the transformation, and the resulting RDF made use of a specifically developed ontology[72]. Later, the RDF data from both sources was uniformized, through a more general ontology for publications: the "Research Publications Ontology". This data was merged and consolidated with FOAF profiles obtained from the web, through removal of duplicates and merging of information relating to the same resource. As [33] refers, the achieved data set[73] still contained some ambiguous information and replicated instances, something natural for a "mashup" resulting from different data sources, and relying on identifiers such as "author names" and "publication titles", that are subject to a high degree of ambiguity. Even if this information is not currently available as Linked Data, this process proved that large information repositories like these can be translated into RDF, with a satisfactory result, and using simple transformations, such as those available to an XSLT stylesheet.

OAI-PMH OAI-PMH (Open Archives Initiative Protocol for Metadata Harvesting) plays a very important role in the dissemination of archive metadata through the web in general. Almost all the major document repositories

[72] `http://sw.deri.org/2005/07/CiteSeer/`
[73] `http://sw.deri.org/~aidanh/expertfinder/`

support this XML over HTTP-based protocol[74]. OAI-PMH describes each "publication" as a "record", that contains metadata that can be expressed in different XML-based formats, being the capability to serialize this information (at least using OAI Dublin Core XML[34, §5]) a requirement of the protocol[34, §2.5]. In addition to this, the records can be grouped in "sets", and filtered by date. The querying capabilities of OAI-PMH are far from being as expressive as, for instance, those of SPARQL, but are more than enough for the retrieval of single documents and/or entire collections.
OAI-PMH is not RDF-based, but its orientation towards publications, and the fact that it usually conveys Dublin Core data, makes the information encapsulated by it easily translatable to the Dublin Core Ontology for RDF. The SIMILE project, at MIT/CSAIL, has developed an "RDFizer" for OAI-PMH repositories[75], that uses an XSLT stylesheet for transforming repositories of metadata provided through this protocol in RDF triples. Example stylesheets for OAI-DC XML are included, but, to the extent of XSLT's limits, any XML-based language can be translated.

Linked Data Fortunately, in addition to several RDF dumps of publication repositories, some Linked Data on the subject is present on the Web as well. The RKB Explorer[76], a project funded by the ReSIST European Network of Excellence[77], provides numerous RDF front-ends for knowledge repositories, each one providing, according to [29], "a SPARQL endpoint, direct access to RDF data through resolvable URIs, [and] a tabulated triple browser for navigating the raw information contained". The data sources include:

- ACM Publications;
- Publications from the European CORDIS[78] Service;
- DBLP;
- CiteSeer;
- IEEE Papers;

[74] There's a list with some of them at `http://www.openarchives.org/Register/BrowseSites`
[75] `http://simile.mit.edu/wiki/OAI-PMH_RDFizer`
[76] `http://www.rkbexplorer.com/`
[77] `http://www.resist-noe.org/`
[78] `http://cordis.europa.eu/`

- Publications from several universities and companies;

For instance, in order to access the author Peter Norvig's description in the CiteSeer data set, one would browse `http://citeseer.rkbexplorer.com/id/resource-CSP229894`. Through an HTML browser, an HTML page with a tabular description of the resource is presented, but, if a user agent is configured to accept only the `application/rdf+xml` content type, RDF/XML is returned. The RKB Explorer data set is referenced in the "Linking Open Data" cloud (figure 8). According to [29], it contains more than 50 million RDF triples, available through both the SPARQL endpoint and Linked Data URIs.
Currently, RKB Explorer seems to be the greatest single repository of scientific publication metadata in the Semantic Web. It is, for this reason, a good option at providing a foundation for a globally-distributed Scientific Web of Linked Data.

2.3 Indico and the Conference Lifecycle

Indico[79] is a web-based, multi-platform conference data storage and management system, developed and maintained at CERN[80]. It allows the storage of documents and metadata related to real events. It started as an European project, but was later adopted by CERN as the substitute for the CERN Agenda. It is FLOSS, being currently distributed under the GPL license. Indico is built on the `mod_python` platform, using the Python programming language. It employs ZODB, an object-oriented DBMS, as its storage engine. Indico includes features that concern the management of different phases of the conference lifecycle:

- Call for Abstracts;
- Abstract submission;
- Book of abstracts;
- Paper Reviewing;
- Automated Registration Process;
- Timetable management - sessions and presentations can be created, and associated with multimedia content;

[79] `http://www.cern.ch/indico`
[80] `http://www.cern.ch`

- Definition of different types of sessions - such as poster sessions and breaks;
- Book of Proceedings;
- Evaluation surveys for the events;
- Allocation of spaces - Room Booking;

An Indico repository is organized as a tree of categories and events. An event is always a leaf, whilst a category can contain either categories or events (never the two simultaneously). A typical path to an event can be something like `Home > Projects > EGEE > EGEE Conferences > 3rd EGEE User Forum` (where "Home" is the "root category").
Besides its HTML-based web interface, Indico provides as well other interfaces that allow exportation of information:

- RSS - For information on upcoming events - a typical Indico RSS feed provides the list of events inside a category, with title, description, date, and a URI to the event page;
- iCal - The information provided by the iCal feeds is similar to that of the RSS ones, for categories. There's also an iCal feed available for each single event;
- OAI-PMH - For the exportation of event information and contents (papers, presentations, etc...) - Unfortunately, the amount of Dublin Core information exported by this interface is small (not much more than the RSS feed), and the rest of the data is available only through MARCXML[81] encoding;

Every user in Indico has his/her own account, being him/her a category moderator, event moderator, presenter, or even a participant on a conference. There are more than 40 known instances of Indico, spread around the world. Institutions like Fermilab (USA), IN2P3 (France) and EPFL (Switzerland) use it regularly. At a first sight, a few dozens of "installs" may not look very impressive as a number for an Open Source project, but given the fact that the "target population" of Indico (large scientific institutions) is quite small, and the amount of information that they store can be huge (around 400000 events, only at CERN), this number grows significantly in importance. The first Indico service (located at CERN) can be reached at `http://indico.cern.ch`.

[81] `http://www.loc.gov/standards/marcxml/`

Figure 12: Screen-shot of Indico's current interface.

2.4 Conclusion - "Pieces of the Puzzle"

Several "pieces of the puzzle" have been presented through the last pages, that need to fit together, in order to allow a Semantic Web-based Research Social Network to evolve:

Publication repositories Even if OAI-PMH is the standard for such metadata, publication repositories start to get connected to the Semantic Web, essentially through third-party services. The excellent service provided by the RKB Explorer is able to provide CiteSeer and DBLP (in addition to ACM and IEEE publications) as Linked Data, while some smaller databases like *openacademia* (see §3.3) start to flourish. Some other repositories that are still not available a Linked Data are possible to be connected to the "graph", through the creation of new "triplifiers" that convert other metadata formats into RDF. The "RDFizers" from SIMILE were already mentioned in this document, and they provide a valuable tool that would allow repositories like CDS to be provided as RDF. The INSPIRE[82] project promises to migrate the vast content of the SPIRES physics repository to the technology that sustains the Invenio[83] system (that is currently behind CDS), thus enabling access to it through OAI-PMH as well (which is currently not possible). The technological barriers are clearly not an issue, even if there is still a slight "babel" of ontologies for this kind of usage: Dublin

[82] `https://twiki.cern.ch/twiki/bin/view/Inspire/WebHome`
[83] `http://cdsware.cern.ch/invenio/index.html`

Core seems not to be enough, and SWRC (employed by *openacademia*) looks as a good alternative; however, large amounts of data are available though AKT (employed by RKB Explorer).

Researcher Profiles - Web-based social networks still have a long way to walk before interlinking is possible. However, FOAF and SIOC have opened the way for a new generation of semantic-based applications to build up. The web is still enjoying the golden age of the so-called "Web 2.0" or "social web", and Semantic Web projects are still too immature to be taken seriously by users - the appearance of a "killer application" would probably change the landscape, but that is yet to happen. However, many people already provide a FOAF profile in their personal homepage and blog, and many social networks already make their information available in RDF as well (as it was already mentioned). Being so, the question of user profile information can be considered a lesser problem for the completion of the "puzzle"

Scientific Events Scientific events are probably the weakest link in the chain. It is true that there are some repositories of upcoming events and CFPs. However, the level of detail of the descriptions is not satisfactory. Except for the SWC Corpus[84], the corpus of information about the European and International Semantic Web Conferences, there are practically no sources of structured information about conferences, seminars, workshops, or other scientific events. Being so, this will be the problem the rest of this document will mostly focus on solving. A particular tool that was already presented, the Indico platform, will be used as the data source for filling this gap.

[84] `http://data.semanticweb.org/`

3 Review of the state of the art

This section will try to provide an overview of the state of art on Semantic Web and related technologies. In the first half, the focus will be placed on the question of databases for the SW (triplestores), and availability of legacy data as RDF. In the second half, the state of art of tools for the Semantic Web of Research will be presented, through a description of applications that have been developed through time in order to address this particular field.

3.1 Technology for the Semantic Web

The Semantic Web aims to be platform-independent, and universally accessible. It is then normal that a wide range of tools already exists, for several programming languages and operating systems, in spite of the project's "tender age". A distributed system of such a dimension needs databases, web servers, programming APIs, and, most importantly, standard-compliant tools.

3.1.1 Triplestores

The most obvious choice for a database of a Semantic Web application is a "triplestore". A "triplestore" is a database of RDF triples (tuples composed a subject, predicate and an object), that supports URI references, blank nodes, and datatypes, as well as all the associated constructs, defined by the RDF model. A triplestore can be seen as a giant graph, in which resources link like they would "in the wild". The most natural way of exploring such a graph is through SPARQL queries, much in the same way that SQL would be the number one choice for a relational database. Triplestores often need to store graphs inside of "contexts". As an example, one can take the FOAF profiles from Alice and Bob: they can be stored in the same triplestore, but it would be good to retain an "annotation" stating that a concrete subset of those triples belongs to Alice's graph, while another concrete subset belongs to Bob's. This takes the reader to the definition of "quadruple"[8], a "triple with context", that is used in such a way that sub-graphs can be easily extracted from the global "conjunctive" graph that constitutes the triplestore. This "context" information is normally used to allow information about the origin of a triple to be specified, placing it inside a specific graph.
Triple stores often rely on existing database architecture to serialize the triple data, such as relational databases, object-oriented ones and key-based storage such as Berkeley DB. There is plenty of solutions for triple storage in the market, many of them available as Open Source. Here are some examples:

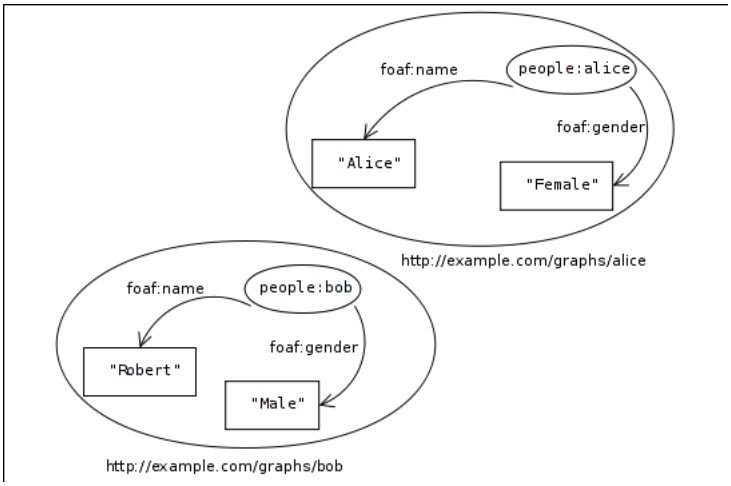

Figure 13: A triplestore, composed of two sub-graphs.

- Sesame - "Sesame is an open source framework for storage, inferencing and querying of RDF data."[85] Its triplestore engine can be deployed over relational databases (PostgreSQL, MySQL, Microsoft SQL Server, Oracle...), file systems and memory storage. It is implemented using the Java programming language;
- Jena - The Jena framework for Semantic Web applications[86] (that evolved from the HP Labs Semantic Web Programme[87]) includes memory-based and persistent storage capabilities (MySQL, Apache Derby, PostgreSQL, Oracle, Microsoft SQL Server, and others);
- OpenLink Virtuoso[88] - Virtuoso combines the functionalities of a RDBMS, ORDBMS, XML database, RDF storage and Web application server. It implements many industry standards for database access (ODBC, JDBC, OLE DB, ADO.NET and XMLA).
- 3store[89] - A MySQL-based triple store, implemented in C, and focused on performance, specialized on handling large data sets;

[85] http://www.openrdf.org/
[86] http://jena.sourceforge.net/
[87] http://www.hpl.hp.com/semweb/
[88] http://virtuoso.openlinksw.com/
[89] http://www.aktors.org/technologies/3store/

- rdflib[90] - An RDF framework for Python, that includes triple storage facilities, using, among other, MySQL, ZODB, Berkley DB and SQLite;
- Redland - Another RDF framework, written in C, with bindings for many languages (Perl, PHP, Python and Ruby), that can store triples using a MySQL, PostgreSQL, SQLite or Berkeley DB back-end.

Some other solutions exist, for other programming languages:

- ActiveRDF[91] is a Ruby framework that allows the developer to manipulate RDF structures in an object-oriented fashion. It can be used alongside the Ruby on Rails platform;
- SWI-Prolog includes an RDF/XML parser, and the `semweb` library[92], that provide a very complete W3C standard-compliant API for RDF manipulation. Some libraries for this Prolog interpreter have been developed as well, at the Centre for Digital Music, Queen Mary, University of London[93], that provide N3 parsing and reasoning support, and SPARQL endpoints on the top of Prolog knowledge bases. The declarative nature of this language, and its logic programming paradigm, make it a good fit for experimenting with reasoning on triplestores;
- "Wilbur[94] is Nokia Research Center's toolkit for programming Semantic Web applications that use RDF (as well as XML and/or DAML+OIL), written in Common Lisp" (from Wilbur's web site). The project doesn't seem, however, to be in active development, since the last release happened in 2005, and many things happened since then, in the field of Semantic Web technologies;

A large list of libraries for different programming languages can be consulted at `http://esw.w3.org/topic/SemanticWebTools`, in the W3C SWIG Wiki.

3.2 Dealing with "Legacy" Storage

It is time to return to the ground level, and think about reality, as of 2008. Robots still do not take care of the household, interplanetary tourism is still science-fiction, and twenty four years have passed without mankind falling

[90] `http://rdflib.net`
[91] `http://activerdf.org/`
[92] `http://www.swi-prolog.org/packages/semweb.html`
[93] `http://code.google.com/p/km-rdf/`
[94] `http://wilbur-rdf.sourceforge.net/`

into some kind of authoritarian mind-controlling regime (at least in the opinion of the majority). Above all, most people still store their data in relational databases. It is not one of the objectives of this document to demonstrate and anti-relational philosophy, or to advocate the replacement of this kind of storage with something different. In fact, RDBMS are very useful and currently irreplaceable in some specific cases; however, one could argue that most of the applications that employ a relational database could do the same, in a much more elegant way, with a different storage engine. The reality is, once more: RDBMS are currently the standard for non-research applications, and not even the advent of object-oriented databases seems to have changed the *status quo*[40].

3.2.1 Making data available

On November 2006, the W3C started an incubator group entitled "RDB2RDF", that, among others, aims to "specify how to generate RDF triples from one or more Relational tables without loss of information"[41]. It is perfectly clear that relational databases, as the a *de facto* "standard", store huge amounts of information that could be available through the Semantic Web, in RDF. That said, it is natural that tools that provide Relational-RDF mapping emerge as possible solutions to this problem.

D2RQ An example of such a tool is the D2RQ platform[95], and its associated D2RQ language. D2RQ addresses the "growing need for RDF applications to access the content of huge, live, non-RDF, legacy databases without having to replicate the whole database into RDF"[16]. It provides access both through Linked Data graphs and SPARQL Endpoints to data that is stored in RDBMS.
Figure 14 shows the general architecture of the D2RQ platform. The "D2R Server" provides a SPARQL endpoint to the repository, RDF output of resources (for Linked Data integration), and an HTML view (used for debugging). This server is fed by the "D2RQ Engine", that makes the actual translation from relational tables to triples, through the use of a "mapping file", specified using the "D2RQ mapping language", itself based on RDF. This "engine" provides as well the possibility to extract an "RDF triple dump" (all the triples present in the database), and interfaces for both the Jena and Sesame APIs.
The core of the transformation process lies, however, in the D2RQ mapping

[95] `http://www4.wiwiss.fu-berlin.de/bizer/d2rq/spec/`

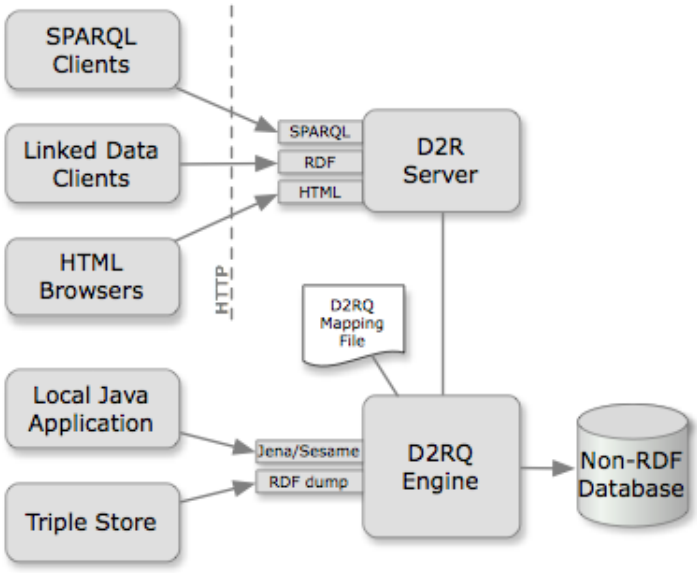

Figure 14: The architecture of the D2RQ platform (diagram taken from [16]).

language. This language is expressed through an RDF ontology, defined at `http://www.wiwiss.fu-berlin.de/suhl/bizer/D2RQ/0.1` (specified in OWL), which is also the root of its namespace. This language encompasses three fundamental concepts:

- Database - an RDF class that represents a relational database, and can have properties such as the ODBC DSN and username/password;
- Class Map - represents an RDFS/OWL-defined class which will be part of the final translation. It is associated with a URI pattern, as well, that specifies how the resulting RDF resources can be addressed. Each "Class Map" can have zero or more property bridges;
- Property Bridge - a "property bridge" establishes a mapping relationship between a column in a database table and an RDF property. The result in RDF can be either a literal or a URI reference. In the former case, both language and datatype of the literal can be specified.

A short example of an D2RQ mapping file can be seen in listing 15. This example maps the database specified in listing 16 to the FOAF ontology,

producing `foaf:Person`s with `foaf:knows` relationships. The three first definitions are pretty straightforward:

- The source database is defined, with its DSN, driver, and access credentials;
- The `map:Person` Class Map defines the target class as `foaf:Person`, and the URI pattern as `people/@@People.id@@`, meaning that, for instance, the "person" with id 1 in the "People" table can be accessed through the relative URL `people/1`;
- The `map:name` Property Bridge maps the field `name` from the `People` table to the `foaf:name` property.

The definition of `map:knows` is not so simple. The object of a `foaf:knows` relationship is a `foaf:Person` URI resource, thus the `d2rq:refersToClassMap map:Person` definition. In addition to this, the SQL JOINs between the `People` and `Acquaintances` table have to be performed, and an alias has to be used, so that the `People` table can be joined with itself, a small "hack" that is needed for the D2RQ engine to differentiate between distinct instances of `People.id`.
The resulting RDF for a possible user can be seen in listing 17. Apart from some auxiliary statements, the information about a user and his acquaintances is perfectly translated, as it can be seen.
D2RQ is being successfully used by several projects, including the already mentioned DBpedia, that is generated from "raw" SQL dumps of Wikipedia. It is seen as one of the successful examples of the capability of providing RDF data from "legacy" data sources, and will definitely play an important role in the movement towards a Semantic Web of Linked Data.

ODBMS The emergence of Object Oriented languages as the main paradigm, in the mid-90's, brought with it the question of persistence. Being able to reuse the objects is future executions of the same program would be very useful, since it would free programmers from the burden of serialization and database querying. Previously, some attempts had already been made to implement this kind of storage in object-oriented languages like Smalltalk (GemStone) and LISP (GBase). This new concept was called "Object-Oriented Database Management System" (OODBMS or ODBMS), and, in spite of its revolutionary paradigm, it didn't go very far in terms of adoption by the regular DBMS customer. The industry chose to adopt, instead, the Object-Relational Mapping (ORM) paradigm as a solution. ORM consists in

Listing 15 A D2RQ example, that maps a simple database of two tables to the RDF model.

```
@prefix d2rq: <http://www.wiwiss.fu-berlin.de/suhl/bizer/D2RQ/0.1#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix map: <file:///home/mahound/projects/thesis/tests/d2rq_test.n3#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

map:Database1 a d2rq:Database;
    d2rq:jdbcDSN "jdbc:mysql://localhost/d2rq_test";
    d2rq:jdbcDriver "com.mysql.jdbc.Driver";
    d2rq:username "test";
    d2rq:password "test" .

map:Person a d2rq:ClassMap;
    d2rq:dataStorage map:Database1;
    d2rq:class foaf:Person;
    d2rq:uriPattern "people/@@People.id@@" .

map:name a d2rq:PropertyBridge;
    d2rq:belongsToClassMap map:Person;
    d2rq:property foaf:name;
    d2rq:column "People.name";
    d2rq:datatype xsd:string .

map:knows a d2rq:PropertyBridge;
    d2rq:belongsToClassMap map:Person;
    d2rq:property foaf:knows;
    d2rq:join "People.id= Acquaintances.subj_id";
    d2rq:alias "People AS FriendlyPeople";
    d2rq:join "Acquaintances.obj_id = FriendlyPeople.id";
    d2rq:refersToClassMap map:Person .
```

Listing 16 SQL code that describes a small relational database with people and their acquaintances. No constraints are included for the sake of simplicity.

```
CREATE TABLE `People` (
  `id` int(11) NOT NULL auto_increment,
  `name` varchar(30) NOT NULL,
  PRIMARY KEY  (`id`)
);

CREATE TABLE `Acquaintances` (
  `subj_id` int(11) NOT NULL,
  `obj_id` int(11) NOT NULL,
  PRIMARY KEY  (`subj_id`,`obj_id`)
);
```

Listing 17 An N3 version of a `foaf:Person` description generated by the previous example. The RDF definition URI is different from the resource URI, and `foaf:primaryTopic` is used to indicate to the agent that reads the definition that the "primary topic" of the document is the resource stated as object. `rdfs:seeAlso` is used to link to all the `foaf:Person`s defined in the repository, using the SPARQL endpoint. The linebreak in the middle of the URI was added only for aesthetic purposes, and should be removed, would the listing be used.

```
@prefix rdfs:    <http://www.w3.org/2000/01/rdf-schema#> .
@prefix foaf:    <http://xmlns.com/foaf/0.1/> .
@prefix map:     <file:///home/mahound/projects/thesis/tests/d2rq_test.n3#> .
@prefix xsd:     <http://www.w3.org/2001/XMLSchema#> .

foaf:Person
      rdfs:seeAlso <http://localhost:8080/sparql?query=
                DESCRIBE+%3Chttp%3A%2F%2Fxmlns.com%2Ffoaf%2F0.1%2FPerson%3E> .

<http://localhost:8080/resource/people/1>
      a       foaf:Person ;
      foaf:knows <http://localhost:8080/resource/people/2> ;
      foaf:name "John Doe"^^xsd:string .

<http://localhost:8080/data/people/1>
      foaf:primaryTopic <http://localhost:8080/resource/people/1> .
```

generating an intermediate layer of code that maps objects, as represented by an OO language (C++, Java, Python...), into relational structures. There are many commercial and Open Source solutions that help in this task, by automatically generating the "glue code": Hibernate/NHibernate[96] for Java/.NET, Apache Cayenne[97] for Java, and SQLAlchemy[98] for Python are some of the best known examples. As in the end data is stored in a perfectly normal RDBMS, it is possible to use the D2RQ platform(§3.2.1) to map all the contained information to RDF.
In addition to the ORM solution, the database industry started incorporating some extensions to the SQL language with some DBMS, providing the developers with the possibility to create their custom datatypes and methods inside the database. These "hybrid" databases were called "Object-Relational Database Management Systems" (ORDBMS). It is true that none of these solutions is so elegant as the ODBMS concept, but, outside research environments, the market share was always overwhelmingly expressive: RDBMS were the choice of the industry[40].
With the advent of the Web, ODBMS started experiencing again a phase of growth, due to the need for lightweight, easy to use (really object-oriented) and not strictly performance-critical data stores. One good example is probably that of ZODB[99] (Python), from the Zope project[100], that empowers the Zope application server, and CERN Indico as well. db4o is another example, targeted at the .NET and Java platforms. The latter has gained substantial acceptance from several enterprise customers[101]. Both systems rely on the utilization of the programming language as the means to create, query, and manipulate database objects. This is seen as both an advantage and a problem, since it implies a higher dependency on programming languages, and, consequently, less portability.
An old approach that has been recently revived for its potential application to web applications is that of Document-oriented databases, that could be described as a lightweight version of ODBMS. These databases are containers of "documents" - registers that can contain an unlimited number of fields. In the case of CouchDB[102], these "documents" are defined as JSON-serialized objects, that can be created, retrieved and updated through a RESTful API,

[96] http://www.hibernate.org/
[97] http://cayenne.apache.org/
[98] http://sqlalchemy.org/
[99] https://launchpad.net/zodb
[100] http://www.zope.org/
[101] http://www.db4o.com/about/customers/
[102] http://couchdb.org/

like in a normal web server. Some other examples of such databases include Lotus Notes[103] (perhaps the oldest one), Amazon SimpleDB[104] and ThruDB[105]. SimpleDB, ThruDB and CouchDB were built with the web in mind, as the target: their simplicity and lightness perfectly fit the requirements of most web applications.

ZODB ZODB was created in the context of the Zope application server, as a persistence system for Python objects. It is distributed under an Open Source License. ZODB relies on the "overloading" of several basic Python constructs as a way to transparently provide persistent objects, and all the required mechanisms (garbage collection, caching...). In order do create a persistent Python class, one only has to extend the `Persistent` base class. Each new instance of this class will be a potential candidate to be stored in a ZODB database. A ZODB database has a root node, that behaves like a normal Python dictionary. Being so, one can add an arbitrary number of entries to it, that will contain either basic types or other dictionaries or lists. This way, ZODB provides a great degree of freedom, allowing the user to define how he/she wants to organize his/her "database tree".
Figure 15 and listing 18 show an example of a ZODB tree, that demonstrates some of the features of this database. Data can be placed inside list-like or dictionary-like structures. In the latter case, the keys can be of any valid type, including objects (in the example, it is a `datetime` object, but it could be a type defined by the user as well). Dictionaries and lists can branch out to other dictionaries and lists, and the value references of two nodes can point to the same object, thus enabling the creation of graphs.
This paradigm is highly flexible, and allows the creation of data structures much in the same way that it is done in normal object-oriented programming. Any instance of a class that extends the `Persistent` base class can be stored, and that includes possibly complex data structures, and classes with a high amount of business logic code. The execution code for the class itself is not stored, but the instance state is completely serialized. Using stored objects in a normal program becomes totally transparent for the programmer, from that moment that the object is retrieved from the database.
ZEO is the "Zope Enterprise Objects" layer, a Python API that allows the access of multiple remote clients to the same ZODB database, at the same time. ZEO makes ZODB perfectly suitable for web development, by enabling

[103] `http://www.lotus.com/notes`
[104] `http://www.amazon.com/b?ie=UTF8&node=342335011`
[105] `http://code.google.com/p/thrudb/`

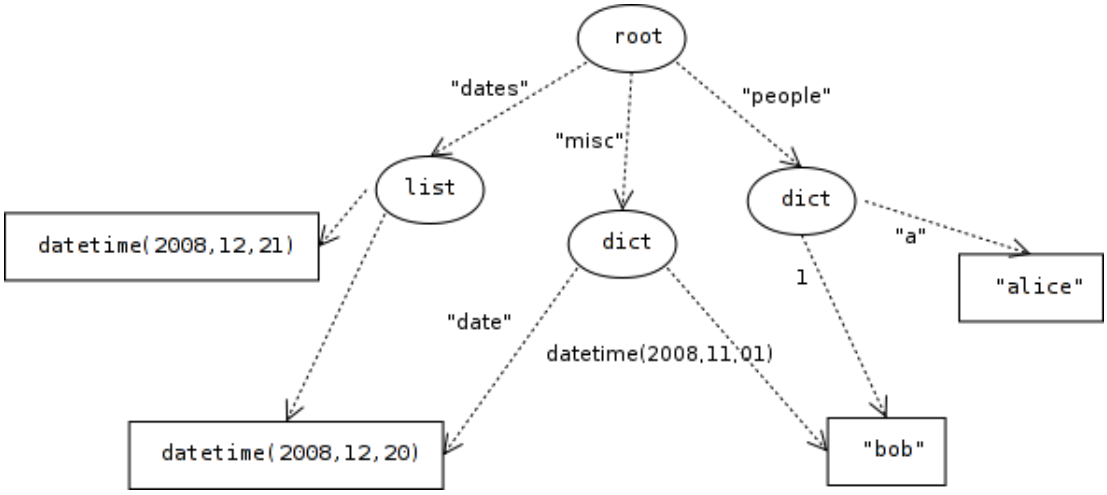

Figure 15: An example of a ZODB "tree", that is, in a correct sense, a graph.

multi-threaded access and deployment of the databases in a different machine from the web application. In spite of all these good points, ZEO/ZODB has some drawbacks too:

- The absence of a mechanism that enables complex queries - this limits in a significant degree the ability to quickly retrieve high amounts of data;
- Performance depends a lot on the implementation of the querying mechanisms - the developer has a higher degree of responsibility in this point;
- "Navigating" through the database is not so easy as listing tables - the few solutions for ZODB "tree browsing" that exist are very primary, and debugging becomes more difficult, for large applications;
- Replication is only handled by the commercial package "ZOPE Replication Services"[106]. This puts a ceiling on data integrity and scalability, for those who can only afford the open source part of ZEO/ZODB.

In spite of these limitations, ZEO/ZODB has proved to be a reliable platform for web development, through the Indico Project that, since 2002, uses it

[106] `http://www.zope.com/products/zope_replication_services.html`

Listing 18 An example of Python code that builds the graph of figure 15, given the root of the database. The resulting structure, serialized in Python code, is shown in the comments.

```
d2 = datetime(2008,12,20)
b = "bob"

root["dates"] = [datetime(2008,12,21),d2]
root["people"] = {"a":"alice", 1: b}
root["misc"] = {datetime(2008,11,01):b, "date": d2}

#{'dates': [datetime.datetime(2008, 12, 21, 0, 0),
#           datetime.datetime(2008, 12, 20, 0, 0)],
# 'misc': {'date': datetime.datetime(2008, 12, 20, 0, 0),
#          datetime.datetime(2008, 11, 1, 0, 0): 'bob'},
# 'people': {'a': 'alice', 1: 'bob'}}
```

as its main database. More information about ZEO/ZODB can be found through its documentation[107].

3.3 Scientific Networks in the Semantic Web

Flink Flink[108] is a "system for the extraction, aggregation and visualization of on-line social networks"[44]. It has been developed by Peter Mika, and it was awarded the first place of the "Semantic Web Challenge" at the International Semantic Web Conference 2004 (Hiroshima, Japan).
The Flink prototype allows the user to browse the network of Semantic Web researchers that took part in the ISWC up to 2005. It depicts the connections between researchers (acquaintance and co-authorship), through graph diagrams, with the help of a Java Applet. The user can easily go from researcher to researcher in a typical hypertext scenario, and even download the papers that were published by the author in question. Normal profile information (interests, affiliation, contacts, etc...) is presented too, as well as some statistics that describe the situation of the individual in the community (impact, connectedness, closeness, and other measures).
Flink gets its data from:

- Google - it calculates the amount of web pages in which two researchers appear associated, as well as the amount of pages that relate a par-

[107] http://wiki.zope.org/ZODB/Documentation/guide/zodb.html
[108] http://flink.semanticweb.org/

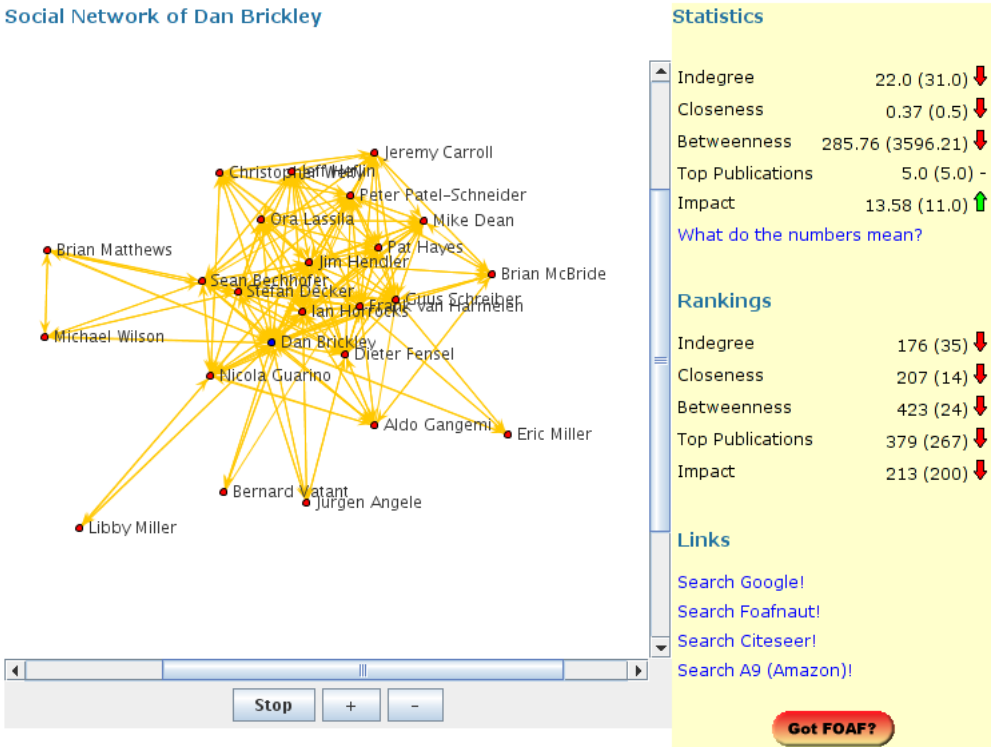

Figure 16: An example of a social graph, as generated by Flink.

ticular researcher with a specific scientific subject, and uses those as a measure;

- The Semantic Web - using the names of the researchers that took part in the ISWC, Flink crawls the Semantic Web, searching for RDF documents that provide information about them;
- E-mail - the system is capable to search for FOAF information inside a mailbox;
- Bibster[109] - Flink imports bibliographical information from this system;

For the Semantic Web "crawling", Flink uses the `rdfs:seeAlso` property[31][§5.4.1], in order to jump from graph to graph - triples that belong to RDF, FOAF,

[109] `http://bibster.semanticweb.org/`

RDFS or WGS84 namespace are stored, and the others are ignored. Then, the extracted `foaf:Person`s are matched against the ones that exist in the database, and any new information is stored. Information about the origin and date of collection of each statement is also recorded. Finally, Flink "smushes" the duplicate identities, using identity reasoning.

openacademia *openacademia*[110] is another of Peter Mika's projects, an evolution of Flink's concept, but this time using a lightweight approach. It can be seen as a semantic equivalent of CiteSeer or DBLP, utilizing BibTeXfiles as source of metadata, and exporting RDF and RSS. It uses the SWRC ontology[111] as the basis for the storage (in a Sesame triplestore) and exporting of RDF. In addition to that, it provides several views over searched results, such as a connection graph (co-authorship) similar to that made available by Flink, tag clouds, and timelines. Each query performed over *openacademia* can be expressed as an RDF URI, and thus embedded in personal sites, or consulted through feed readers (i.e. an author can make a list of his/her works available at his/her homepage).

Conclusion Flink is probably the existing application that, for the nature of its prototype, more closely resembles the idea of a Research-oriented Social Network based on the Semantic Web. Its problem resides mainly on the fact that it has not evolved in order to accommodate new data sources that started to appear all around the web. Its development seems to have stopped in 2005, when the last data set was imported. It is, however, an excellent proof of concept for a semantic web application, and, specifically, for a Semantic Web based Social Network. `openacademia` is a lighter approach to such a question, and is currently more oriented towards publications. It doesn't discard the social aspect of research networks, since it tries to identify the authors and establishing relationships among them (and the topics they explore), but it privileges publications as the main subject of the application.

[110] `http://www.openacademia.org/`
[111] `http://ontoware.org/projects/swrc/`

4 A Solution

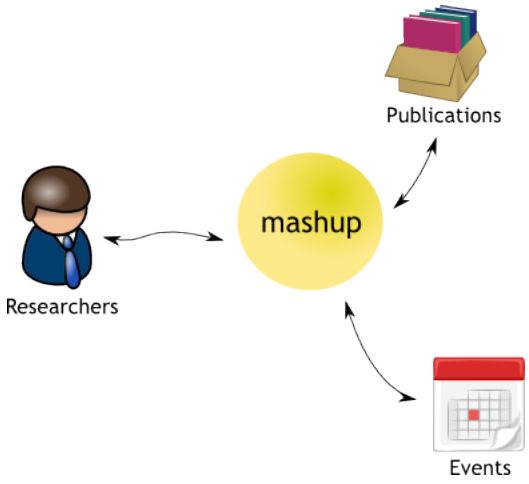

Figure 17: A possible solution: a semantic mashup

There is no doubt that the availability of RDF repositories of information about scientific events is an important issue for the construction of a global Scientific Semantic Web. Providing that both publication repositories and personal information data sources already have a substantial presence in the Semantic Web (even if some problems of uniformization of data exist), the creation of detailed information about conferences available as metadata would put the three parts in an equilibrium that would allow some work to be developed for their integration. Once this "equilibrium" was established, a process of data linking could start. This would require, though, some kind of point of crossing, an "interleaver", a "mashup" that would connect the vertexes of this triangle, by "smushing" information from different sources, and unifying identities, as a way to create a compact and heavily linked set of resources that could efficiently describe a network of researchers.

4.1 Indico as a Platform for Scientific Webs

Looking at the case of Indico once again, the huge amount of information that is stored inside the ZODB database is currently accessible through a plain HTML 4.01 based web interface, RSS and iCal feeds for event lists, and OAI-PMH for metadata about publications. Unveiling this data source to the Semantic Web would add a powerful ally to the various data sources

involved in the Linking Open Data Movement, providing Linked Data information about events, users, and eventually contents. As it was already mentioned in §2.3, Indico syndicates its information in a tree-like way, placing events inside categories, that themselves can be contained inside other categories. It would be definitely interesting to study the way in which the taxonomy that this "tree of categories" generates by going from a general repository (root) to specific categories like "Meetings from the IT-UDS-AVC section" could be used in a semantic point of view: maybe a model for such qualifications could be translatable to some taxonomy-describing language like SKOS (mentioned in §1.3.1), thus allowing intelligent agents to browse the "Indico tree" in the same way that a human does, perhaps looking for conferences related to some specific project or trying to contextualize a known event in a specific field of science. This would be definitely interesting. However, this document centers itself on the primordial question of making all these events, and the information that they link to, available to the Semantic Web community, before some higher level purposes can be aspired. For now, obtaining a description of a conference - title, summary, links to participants and contents, as well as possible sub-events (presentations, sessions) - is the priority, and an indubitably vast development. The information that is currently stored in Indico could be made available through different mechanisms:

- The most obvious one would be a direct translation, through the writing of a significant amount of code that would translate the information to RDF, either on request, or through a complete database translation mechanism (depending on such constraints as the necessity to have a "queriable" triplestore, or the need to have real time information, consistent with the database).

- One other way would be to use a mechanism that is currently being experimented with the new developments that are being made in Indico[26], that allows the translation of Object data to structured data that approximately follows the JSON data model (as a way to provide the client-side AJAX interface with representations of the business objects). Building a "bridge" between the information generated by this mechanism and the RDF model would be less expensive than the previous option, but the fact that the resulting structures are provided mainly in the form of JSON objects, with properties that are usually strings, introduces an extra complexity of dealing with the question of data types. This is enough to spoil the "elegance" of such solution, and to create undesirable lines of code.

- Yet another option would be to re-utilize the existing OAI-PMH interface, through a mechanism that would be similar to SIMILE's "RDFizers". However, mapping MARCXML to RDF would be a substantially unpleasant task, due to the enormous amount of existing MARC fields, and the resulting redundancies and replicated data. The improvement of the Dublin Core encoder would make the effort fall into the same category of the first option.

- Finally, the last considered solution would be that of directly mapping the ZODB database, that is itself an Object storage, and taking advantage of its Object-Oriented mechanisms, in order to easily map the object properties to RDF. The problem would be the need to create a way to map this information, but the benefit would come if it were possible to create such a mapping mechanism that would be ready to be used in other contexts.

The last option seemed the most profitable, since the perspective of reusing such a mechanism looked as an advantage. In fact, the amount of "glue code" that would be needed to make all this information available as Linked Data, only through "direct translation" would totally compensate for the time that would be wasted at creating a flexible, extensible and possibly standard solution . The technological part should remain intact (unless there were significant changes in the API of ZODB), while a translation model would be kept up-to-date with the latest developments of the Indico API.

4.2 OURSE

Why OURSE? OURSE is the acronym for "Object Utilizer for RDF Semantic Exporting". It was created by the author as a device for extracting of information from Object-oriented databases, and making it available as RDF Linked Data, and through a SPARQL endpoint as well. Naturally, it was heavily inspired by the D2RQ platform, but, because of its object-oriented nature, it had to cope with slightly different problems. It is the belief of the author that Object-oriented, and most probably Document-oriented databases will rise in importance among the web development scene. The need for lighter, faster and simpler data storage than a plain SQL database (that is currently used for most web applications) is clearly evident. Some examples of applications that currently use mostly relational solutions, when a different approach would be much more efficient, are:

- Blogs, that only need to store posts, comments and a slim amount of

configuration data, and still mostly base on relational databases (i.e. Wordpress[112]);

- Wikis, that are clearly document-oriented, and still many times make use of relational storage (i.e. MediaWiki[113]);
- Social networking sites, in which entities such as "users" behave in a totally object-oriented way, exchanging messages and adding content to their internal data repositories (photos, texts, etc...);
- Small projects like Pastebins[114], bookmark managers, and simple folksonomy-based applications, that store data in a simple way, and don't require anything more complex than a document-oriented storage.

The degree to which it is feasible to implement these applications using a document-based approach is itself dependent on the complexity of each specific project, and nature of the storage system that is employed. With the exception of the "Social network", all of them require a relatively simple database "schema", that should be easily manageable with solutions like CouchDB. Still, all of them are for sure implementable with recourse to an ODBMS: i.e. Plone[115] implements the features of a Wiki and a Blog at the same time, using ZODB as a basis; Indico is a powerful Content Management System and agenda application, that allows the storage of different kinds of media types, room booking management, conference website management and user messaging: once more, ZODB is used to store most of the data.

Overview OURSE should not be seen as a particular implementation of an "OO to RDF" translation mechanism. In fact, there is a prototype of a reference implementation, done in Python, that makes this translation. However, OURSE should instead be described as a mapping language for making data that is present in object repositories available as RDF. The main problem here is not finding how to translate concepts (since both models have several things in common), but to overcome technological barriers, by providing a description of "how to map" these concepts in the real world, using the data access methods that an ODBMS has to provide. Similarly to D2R, OURSE bases itself on a few concepts:

[112] `http://www.wordpress.com`
[113] `http://www.mediawiki.org/`
[114] `http://pastebin.com/`
[115] `http://plone.org/`

- Database - Roughly the same behaviour as in D2R, with the slight difference that the "username" and "password" are condensed in the DSN, in the form `driver://user:password@server:port`;
- Class Map - Like in D2R, a URI pattern is accepted, with a string that contains an URI "skeleton", and a special notation for specifying fields that should be replaced with literal values, like in a regular expression substitution (i.e. `http://example.com/people/@@id@@`). However, unlike D2R, it is not enough to specify table and column names, using a specific notation, since, for instance, a ZODB "tree" has a virtually infinite "deepness". In addition to that, most of the data articulation problems that in D2R can be solved with an "SQL JOIN" cannot be addressed in the ODBMS case, since there's no widely adopted querying language for this kind of databases. Constructs that are used to solve both problems are specified below;
- Property Mapping (Bridge) - The concept is the same as in D2R, once again with help of some extra constructs that make data extraction possible;
- Object Provider - An abstract entity, tied to a database, that provides Class Maps with the objects that will be translated. Such a provider is responsible for "digging" into the database and getting the desired object;

Going Object-Oriented It was already mentioned that the "deepness" of some object stores, compared with the "shallowness" of relational databases (that can usually go no further than the "column" detail), poses an important problem to the specification of the origin of data. In OURSE, this problem arises in two situations: when a Class Mapper needs to fetch an object from a database (through the use of an Object Provider), and when a Property Mapper needs to fetch a property value from a database object. In order to solve the latter, the notion of "getter" was introduced. In OOP jargon, a "getter" (or "Accessor") is a method that provides an interface to a usually private class attribute: this technique is used as a way to avoid direct access to class attributes. Since most database objects are likely to provide this kind of interface, OURSE defines a `MethodGetter` as an entity that applies a method over an object, in order to extract the result and use it in the transformation process. However, it provides, as well, two other "getter" types:

- `ViewGetter` - Given the name of a "view", executes its code over a database object, and returns the resulting value/object. A "view" can be theoretically defined in any programming language, and it consists solely in a function that takes the database object as parameter, and return the desired attribute. This is useful for attributes that are not directly defined in the database object, and thus have to be calculated on the fly.

- `IdentityGetter` - Like the name says, this getter returns what it takes as argument. The utility of this construct may not seem evident at the first look, but there are situations when it becomes fundamental. As an example, let's suppose that a database object from the class `Person` has to be mapped to `foaf:Person`, and the developer wants to map the acquaintances of each `Person`, using the `foaf:knows` property, that takes another `foaf:Person` as object (first line of fig. 18). Normally, the `Person` database class should have a `getAcquaintances` method (accessible though a `MethodGetter`) that would return a list of instances of `Person` (fig. 18, second line). However, it may happen the case where the class method does not return a list of `Person` instances, but instead (for some reason) directly a list of names (strings). As it can be seen through the depiction on the 3rd line of figure 18, this will result in an inconsistency, that would force the developer to present the object of `foaf:knows` as a string literal. However, if an extra level of processing, that encapsulates the string value as the `foaf:name` of a blank node is added to the equation, and an `IdentityGetter` is used to "extract the string from itself", the latter will be passed one extra level ahead, and be presented, as expected, as the value of a `foaf:name` property.

The "getters" mechanism solves the problem of retrieving properties from database objects, but not that of retrieving the database objects, themselves, from the database. This is addressed by the `ObjectProvider` construct.

Object Providers Object Providers are responsible for extracting the database objects from the stores they relate to. Since each database product has its own architecture and schema, these "object providers" are associated with a specific database. However, they aim to provide a universal way of specifying how to extract the desired information: each "database driver" implementation should be able to figure out the "physical path" to the required resources, relying on the information that the provider

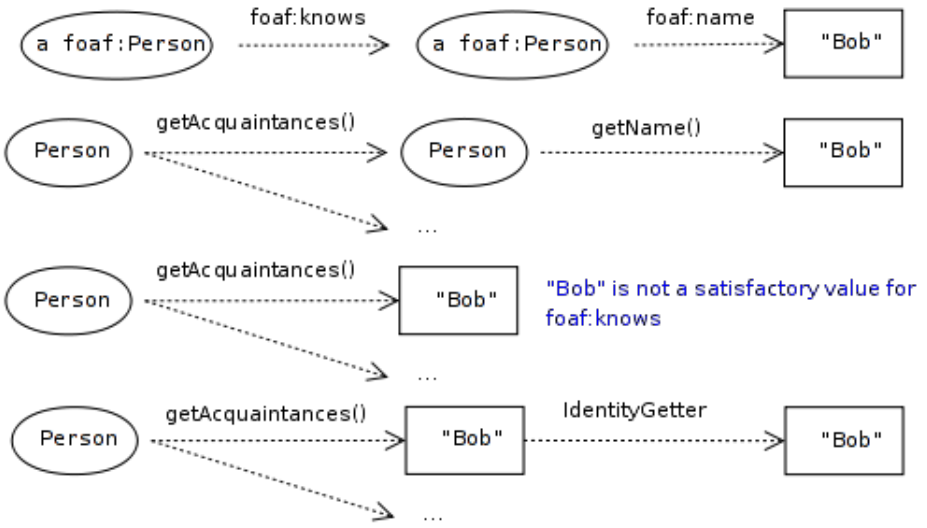

Figure 18: An illustration of the use of an `IdentityGetter`

passes. OURSE currently implements only one type of Object Provider: the `PathObjectProvider`. This particular kind is aimed at tree-based databases, like ZODB. The rationale is very simple: a tree-based ODB has a root node, and different "keys" for each branch; the objective is to "serialize" the path towards a leaf (information the provider wants to extract) in a way that resembles a filesystem directory path, or even XPath: `users/1` will point to `root['users']['1']`, in Python/ZODB syntax. Object Providers usually have to retrieve a particular "record", with a key that is passed as a parameter. In order to solve this, the previous example could be generalized as `users/@@id@@`, being `id` a parameter that is passed to the provider, in order to retrieve a particular object.

Bindings A problem that quickly arises in any implementation of a translation mechanism of this nature is that of how to generate a URI for a specific object. The scenario is simple: a client requests a URI, that is mapped into some object in the database. The object is retrieved, but one of its properties is supposed to reference another object through its URI - `foaf:knows` refers to another `foaf:Person` that, in this case, would be stored as a different object. The URI for this second object will have to be generated. This involves getting the URI pattern that is specified in the class map, and "filling the blanks" with object properties: for instance, `people/@@id@@` would have

to become `people/1`, where "1" is the person ID, contained in the database object. However, there's no way for the engine to know what property to fetch for each field. This is when the concept of binding comes into play: a binding connects a "label" or "field name" to a concrete value, by specifying a Getter that will extract the desired identifier. This way, the engine will know how to generate the URI for the object. This becomes much clearer in the example case that is presented below.

Reference Implementation The reference implementation[116] of OURSE (pyOURSE) was done using the Python programming language, and the rdflib[117] API. The visible part of this system is the OURSE Data Server, a web server that provides Linked Data RDF output and a SPARQL endpoint (fig. 19).

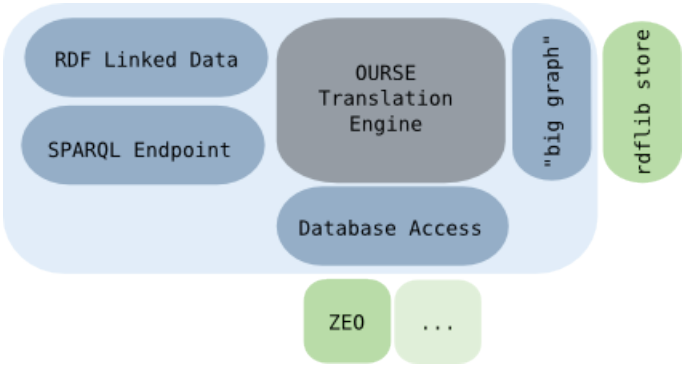

Figure 19: Overview of the OURSE Data Server

The Data Server does "on the fly" conversion of database objects to RDF, using the OURSE translation language as a means to specify this transformation. URL patterns are indexed, and each access attempt to a server resource, by a client, is mapped to the required RDF resource, that is, then generated. In addition to this, and in order to allow SPARQL queries over the data, the Data Server also supports a process of one-time conversion of a complete database to RDF, and its storage in a "big graph", a conjunctive graph with one sub-graph for each different object. Obviously, this "big graph" has to be updated regularly - this raises the problem of consistency between the two stores, that could be solved by triggering partial updates of the conjunctive graph. This is an improvement that will not be discussed in

[116] code available at `http://github.com/mahound/ourse/`
[117] `http://rdflib.net`

this document, though. The "big graph" is being stored in an rdflib triple-store, with a Sleepycat (Berkeley DB) back-end.

Example In order to help proving that OURSE actually works, a small example is mentioned, involving a small ZODB database of people, that needs to be translated to the FOAF ontology. Listing 19 shows an example Python class, that stores information about a "Person" and some attributes, such as name, email, age, and a list of acquaintances. A unique identifier is stored as well, and accessor methods ("getters", and a "setter" for the acquaintance list) are defined.

Listing 19 Python code for a `Person` class.

```
from persistent import Persistent

class Person(Persistent):
    def __init__(self, id, name, mail, age):
        self._id = id
        self._name = name
        self._mail =  mail
        self._age = age
        self._acquaintances = []

    def getId(self):
        return self._id

    def getName(self):
        return self._name

    def getEmail(self):
        return "mailto:%s" % self._mail

    def getAge(self):
        return self._age

    def getAcquaintances(self):
        return self._acquaintances

    def addAcquaintance(self, person):
        self._acquaintances.append(person)
```

Listing 20 shows Python code that generates `Person` objects for three individuals, and establishes some acquaintance relationships between them.

Then, these are stored in a Python dictionary, and finally placed at the entry with key "people", at the root of the database.

Listing 20 Excerpt of Python code that fills a small test database, based on the `Person` class.

```
root = connection.root()

alice = Person('1', "Alice", "alice@example.com", 23)
bob = Person('2', "Bob", "bob@example.com", 22)
carol = Person('3', "Carol", "carol@example.com", 24)

alice.addAcquaintance(bob)
bob.addAcquaintance(alice)
bob.addAcquaintance(carol)

root["people"] = {'1' : alice,
                  '2' : bob,
                  '3' : carol }
```

After running the script that creates and fills the database (an excerpt of which is described in listing 20), the OURSE Data Server has to be invoked (listing 21), using a proper OURSE translation language RDF file, tailored for the example database (listing 22). As it can be seen in listing 21, the amount of code that is actually needed to boot up the data server is minimal, and all the translation logic is stored in "ourse_test.n3", defined in listing 22. The Python program starts by creating a "Conjunctive Graph", an rdflib structure. Then this graph is loaded with the translation file (that, since it is in RDF, is actually a graph). After that, the storage mechanism is initialized (Sleepycat/BerkeleyDB is used, but any store supported by rdflib will work), and the data server is finally invoked, with the configuration file graph, the host it should respond to (and HTTP port), the data store, and a SPARQL Endpoint definition (in this case) passed as parameters. Both the store and SPARQL endpoint are optional parameters, and are assumed to be "I/O Memory" and "None" by default, respectively. The SPARQL Endpoint definition states a regular expression that will match relative URIs which will answer as the endpoint.

The translation file clearly echoes the main concepts involved in OURSE, that were already described before in this document. The database definition seems obvious: it is a simple DSN, without any username or password (since authentication is not being used for this example), and the database driver is "ZEO" (the only one available at the moment). Since there's only one class

Listing 21 Python code that initializes the OURSE Data Server, using the file from listing 22.

```
from rdflib import URIRef
from rdflib.Graph import ConjunctiveGraph, Graph
from rdflib.store.IOMemory import IOMemory
from rdflib.store.Sleepycat import Sleepycat

from ourse import OURSE
from ourse.sparqlserver import SPARQLEndPoint
from ourse.dataserver import OURSEDataServer

graph = ConjunctiveGraph()
graph.parse(open("ourse_test.n3","r"), format="n3")

storage = Sleepycat()
storage.open("./storage")


OURSEDataServer(graph,'localhost',8080,storage=storage,
                sparqlEP=SPARQLEndPoint(r'^\/sparql\/?$'))
```

involved, `Person`, only one Class Map is defined, with `foaf:Person` as the target class, and a relative URI pattern `/profiles/@@id@@`, meaning that i.e. `/profiles/1` will be mapped to this class, with the `id` parameter set as 1.

The `map:Person` Class Map needs an `ObjectProvider`, that extracts objects from the database that was already defined, fetching them from `root["people"][id]`, where the `id` is generated through URI matching (the `id=1` case which was mentioned just before, for instance). A "binding" is defined as well, for the reverse transformation of object data to URIs. The `getId()` method is set as the "getter" that will retrieve the "id" parameter to be used in the generated URI.

Finally, a list of Property Mappings follows: the name (a simple string), the age (an integer), and the mailbox (a URI) are simple mappings, that do not require the transformation of further objects. The list of acquaintances is translated into `foaf:knows` properties, through mapping the `getAcquaintances` "getter" to the `map:Person` Class Map. This way, the acquaintances will be processes as `foaf:Person`s as well, and all the needed transformation mechanisms will be invoked. Since `getAcquaintances` returns a list of `Person` objects, the result will be multiple instances of the

`foaf:knows` property (which is exactly what is expected).
After booting up the OURSE Data Server, using the Python script from listing 21, one can access URIs such as `http://localhost:8080/profiles/2`, using a web browser, obtaining as a result an RDF file, serialized in RDF/XML or N3, depending on the content of the "Accept" HTTP Header[27, §14.1] issued by the browser. A possible result is shown in listing 23, in N3. As it can be seen, the data and relations that were established for the database are kept in the RDF result. URIs are resolvable, and useful information is returned, in RDF. It may be said, then, that OURSE is capable of expressing this database through RDF Linked Data.
However, as it has already been mentioned before, Linked Data is not enough for large repositories of data. Though this is clearly not such a case, a SPARQL query can be performed over the data that is present in the OURSE Data Server triplestore. A particular RDF sub-graph is stored in the triplestore as soon as it is generated. That "generation process" can be triggered either programmatically (calling directly the graph serialization method from the data server), or simply by accessing it through its HTTP URI. Once a URI for a particular graph is loaded, it will be "cached" in the triplestore, and "regeneration" won't be needed anymore.
After the graphs are loaded into the triplestore, SPARQL queries can be easily performed over it. It is just a matter of requesting `http://localhost:8080/sparql`, with a mandatory `query` HTTP parameter, specifying the SPARQL query (properly encoded), and an optional `format` parameter, that can be either "json" or "xml", for the serialization of the SPARQL results in either XML or JSON (the default is currently JSON). Being so, if the query in listing 24 is performed (after being properly encoded and passed on as a POST/GET HTTP parameter), the result from listing 25 is returned (XML was chosen as the serialization format, in this particular case). The query simply gets all the pairs of individuals that know each other, and presents their names.
In spite of the apparent simplicity of this example, it is worthy to stop for a moment to think about all the operations that happen in the background, when a particular request is done. A URI is translated to a Class Map and an identification parameter, and this Class Map is bound to a specific Object Provider, that takes the burden of going to the database and extracting the object that matches the parameter. Then, this object is translated to RDF, using Property Mappings that have been defined: for Simple Property Mappings, the process is just a matter of extracting the value, using a "getter", and converting it to the required datatype, and for Complex ones, a URI reference to another object has, usually, to be generated (except in some

Listing 22 OURSE definition for the mapping of the `Person` class to `foaf:Person`.

```
@prefix :  <file://ourse.owl#> .
@prefix map: <file://ourse_test.n3#> .

# Other namespaces
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

map:Database1 a :Database;
    :DSN "zeo://localhost:9675";
    :Driver "ZEO".

map:Person a :ClassMap;
    :class foaf:Person;
    :uriPattern "/profiles/@@id@@";

    :objectProvider [ a :PathObjectProvider;
      :dataStorage map:Database1;
      :DBPath "people/@@id@@" ];

    :bind [ :getter [ a :MethodGetter;
      :method "getId" ];
    :parameter "id"  ];

    :propertyMapping [ a :SimplePropertyMapping;
            :property foaf:name;
       :getter [ a :MethodGetter;
                    :method "getName" ];
       :datatype xsd:string ];

    :propertyMapping [ a :SimplePropertyMapping;
            :property foaf:age;
       :getter [ a :MethodGetter;
                    :method "getAge" ];
       :datatype xsd:int ];

    :propertyMapping [ a :SimplePropertyMapping;
            :property foaf:mbox;
       :getter [ a :MethodGetter;
                    :method "getEmail" ];
       :datatype xsd:anyURI ];

    :propertyMapping [ a :ComplexPropertyMapping;
            :property foaf:knows;
       :getter [ a :MethodGetter;
                    :method "getAcquaintances" ];
       :targetClassMap map:Person ].
```

Listing 23 Result of an HTTP request to `http://localhost:8080/profiles/2`.

```
@prefix _3: <http://xmlns.com/foaf/0.1/>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.

 <http://localhost:8080/profiles/2> a _3:Person;
     _3:age "22"^^<http://www.w3.org/2001/XMLSchema#int>;
     _3:knows <http://localhost:8080/profiles/2>,
         <http://localhost:8080/profiles/3>;
     _3:mbox <mailto:bob@example.com>;
     _3:name "Bob".
```

Listing 24 SPARQL query that returns every pair of acquaintances (the names).

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?s ?o WHERE
{
  ?sr foaf:knows ?or;
      foaf:name ?s.
  ?or foaf:name ?o.
}
```

Listing 25 Result of the SPARQL query, as returned by the OURSE Data Server, encoded in SPARQL Query Results XML Format: (Bob, Carol), (Alice, Bob) and (Bob, Alice).

```
<?xml version="1.0" encoding="utf-8"?>
<sparql:sparql xmlns:sparql="http://www.w3.org/2005/sparql-results#"
               xmlns:xml="http://www.w3.org/XML/1998/namespace">
  <sparql:head>
    <sparql:variable name="o"/>
    <sparql:variable name="sr"/>
    <sparql:variable name="s"/>
    <sparql:variable name="or"/>
  </sparql:head>
  <sparql:results distinct="false" ordered="false">
    <sparql:result>
      <sparql:binding name="s">
<sparql:literal>Bob</sparql:literal>
      </sparql:binding>
      <sparql:binding name="o">
<sparql:literal>Carol</sparql:literal>
      </sparql:binding>
    </sparql:result>
    <sparql:result>
      <sparql:binding name="s">
<sparql:literal>Alice</sparql:literal>
      </sparql:binding>
      <sparql:binding name="o">
<sparql:literal>Bob</sparql:literal>
      </sparql:binding>
    </sparql:result>
    <sparql:result>
      <sparql:binding name="s">
<sparql:literal>Bob</sparql:literal>
      </sparql:binding>
      <sparql:binding name="o">
<sparql:literal>Alice</sparql:literal>
      </sparql:binding>
    </sparql:result>
  </sparql:results>
</sparql:sparql>
```

particular cases that involve blank nodes).

Conclusion In spite of the optimistic scenario that was presented through the last paragraphs, OURSE is still far from being stable and complete. The main problems are:

- Object Providers are still not very developed. `PathObjectProvider` can currently handle only keys that are strings, since there's no way to specify the key's datatype. More sophisticated mechanisms will have to be developed, so that a greater degree of expressiveness is attained, without the drawback of making them excessively technology-dependant. An equilibrium will be needed at this point.
- Some RDF constructs cannot still be expressed by OURSE: such an example is that of containers and collections[46, §4]. This is not a serious issue, since those are not used so often, but they will certainly be incorporated in future OURSE releases.
- Even if OURSE is designed in a way that "drivers" for other databases are easily built and plugged in, the truth is that ZODB has been, so far, the one and only "guinea pig" to be really tested. Some testing on emerging and also older object-based database systems is definitely needed.
- pyOURSE relies on rdflib, and rdflib itself is going through deep transformations. The version that is used by pyOURSE is 2.4.0, that is fairly stable for RDF parsing, serializing and manipulation, but lacking some optimization in the storage mechanisms. pyOURSE is dependant on the evolution of this project's maturity.
- With the introduction of more complex constructs, there is the chance that the RDF ontology that is employed by the OURSE translation language will become more and more complex, eventually getting to a point where the readability of specifications will be affected. An RDF-based specification model was chosen because of the example of D2RQ, that employs it in a clean and concise way. However, the differences between the two problem domains are not guaranteed to be small enough for such a danger to be out of question. This is, though, the lesser problem in the list, at least for the short term.

The reference implementation suffers from a problem that is more of a challenge than an actual problem - The reference implementation is currently

storing generated graphs in a global, conjunctive graph, so that SPARQL queries can be run over the data. However, this brings in the question of consistency between the original (ZODB) data source, and the Data Server's triplestore. This can be tackled through periodical verification cycles, that could check the database timestamps for the objects, and update them if needed. Evidently, this will only work for databases that store metadata about the objects (timestamp of last modification, etc...), and will involve an extra layer of metadata about sub-graphs, in the Data Server triplestore (i.e. statements like "sub-graph for `http://example.com/bob` was retrieved on 27/11/2007, 11:32:43AM GMT" have to be expressed). In any case, periodical updates do not solve the problem for databases that require strictly real-time information, and those cases could only be solved either through triggered updates (immersing the data server logic deep down to the level of the origin database's API), or simply by discarding the conjunctive graph, and generating only "small" graphs, on the fly, for requested URIs (of course this would disable an hypothetical SPARQL Endpoint).
It is clear that OURSE is still walking its first steps towards a possible status of "usable technology". It does not intend to be an instantaneous solution to the problem of "triplification" of legacy databases: in that regard, technologies such as D2RQ and Triplify[118] are good (and stable) approaches, with D2RQ proving to be a well performing database-oriented mechanism (enough to empower DBpedia), and Triplify a more recent, application-oriented, lightweight approach. This is because object-oriented databases are not the standard right now (as it was said many times through this document), and OURSE itself, is more of an exploration (born from the need to make Indico data available as Linked Data) than a "consumer-targeted" product. However, there are several paths that OURSE could take, that would certainly place it on the map of Semantic Web technologies:

- A possible shift towards Object-oriented scenarios that do not necessarily involve databases - there's a subset of OURSE that can be used simply as an Object-RDF mapping language, that could be employed by applications as a "standard" for this kind of transformation. Applications of the nature of Oort[119] are potential candidates.

- Adding to the previous idea the development of a "two way transform" mechanism - This means that OURSE would allow both Object-to-RDF and RDF-to-Object transforms to be specified, in a unified

[118] `http://triplify.org/`
[119] `http://oort.to/`

syntax. This is obviously not simple, since, in addition to "getters", "setters" would have to be taken in account, and possibly other object-oriented mechanisms, such as constructors. On the other hand, this would definitely be useful in many contexts, for different Semantic Web applications.

- Exploring the territory of document-based databases - Stores such as Apache CouchDB are slowly gaining visibility in the field of web content storage. Some believe that, in the future, the web will rely on these lightweight databases to make information easy to store, read and translate. Document databases can be seen as a lightweight version of object databases, and adding basic support, in OURSE, for CouchDB, would be almost trivial. That said, a device of the nature of OURSE could have great importance, "gluing together" storage mechanisms and the RDF model, in the Semantic Web.

4.3 The SWRC/SWC Ontologies

The SWRC (Semantic Web for Research Communities) ontology is an important ally in the description of a worldwide network of researchers. It comprises 53 different concepts, and 42 properties[55], modelling "key entities" in research communities, such as researchers themselves, publications, events, research topics and projects. It can be considered the "Swiss army knife" of the "Scientific Web".
There are seven basic concepts in this ontology:

- Person - A researcher, either a "Student" or an "Employee". These classes stem into different subclasses, that can be used to provide a better degree of detail on the background of the person that is described (fig. 20). There's a total of 18 subclasses of "Person", that provide a description level that goes to details like "Honorary Professor" and "Professor Emeritus".

- Organization - Represents organizations, to which researchers are associated. This includes universities, departments, institutes, research groups, associations and enterprises.

- Event - That splits into seven different types of event, from colloquia to exhibitions, lectures, meetings, seminars and workshops;

- Publication - In total, there are 22 subclasses of the "Publication" class. This allows the specification of several types of publications, like dif-

ferent types of thesis (PhD, Master and Diploma), reports, research papers, proceedings, manuals, journals, and some others. These are modelled to the image of the BibTeXstandard, widely used for bibliographical data;

- Project - That can be either a "development project" or a "research project";
- Research Topic - A subject, discipline or whatever describes something possible to be studied;
- Product - An auxiliary concept, that is usually related with Organizations and Projects, that describes some product that is being developed;

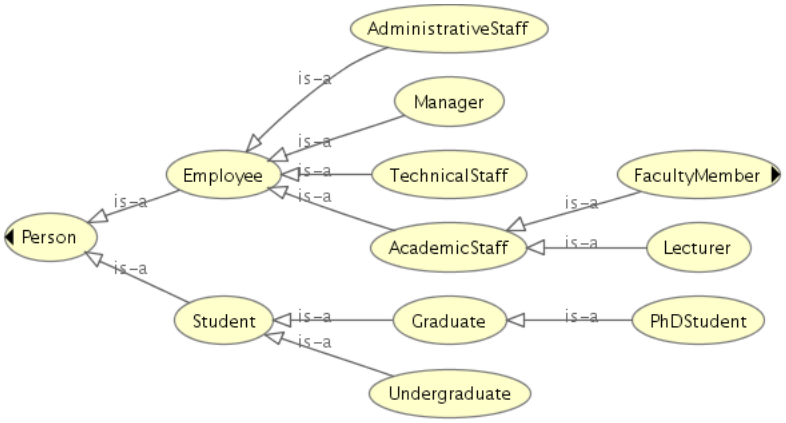

Figure 20: The sub-classes of `Person`, in the SWRC ontology. The "children" of `FacultyMember` are not shown.

All these concepts would be worthless if it weren't possible to relate them by the means of properties. Apart from the usual literal properties that describe each resource (i.e. the title of a publication or the name of a person), SWRC introduces several types of relation that can be established between the different concepts. Figure 21 shows an example of some of these properties:

- An `swrc:Person` can be related to a `swrc:Organization` using the `swrc:affiliation` property;

- The `swrc:Topic` class is referenced from several other classes, through the `swrc:isAbout` property; these include `swrc:Product`, `swrc:Project`, `swrc:Document` and `swrc:Event`, all of them concepts that usually require a subject (i.e. "a conference on Semantic Web Mining");
- The `swrc:Document` class can be referenced through the `swrc:publication`, from `swrc:Persons` and `swrc:Organizations`;
- `swrc:Events` can include an organizer/chair `swrc:organizerOrChairOf`, and be placed inside other events through `swrc:eventAt`. This is useful for placing sessions inside conferences, for instance;

This is just a small fraction of the framework provided by the SWRC ontology. [55] goes into this in more detail, and provides as well some examples of applications that are currently using this tool.

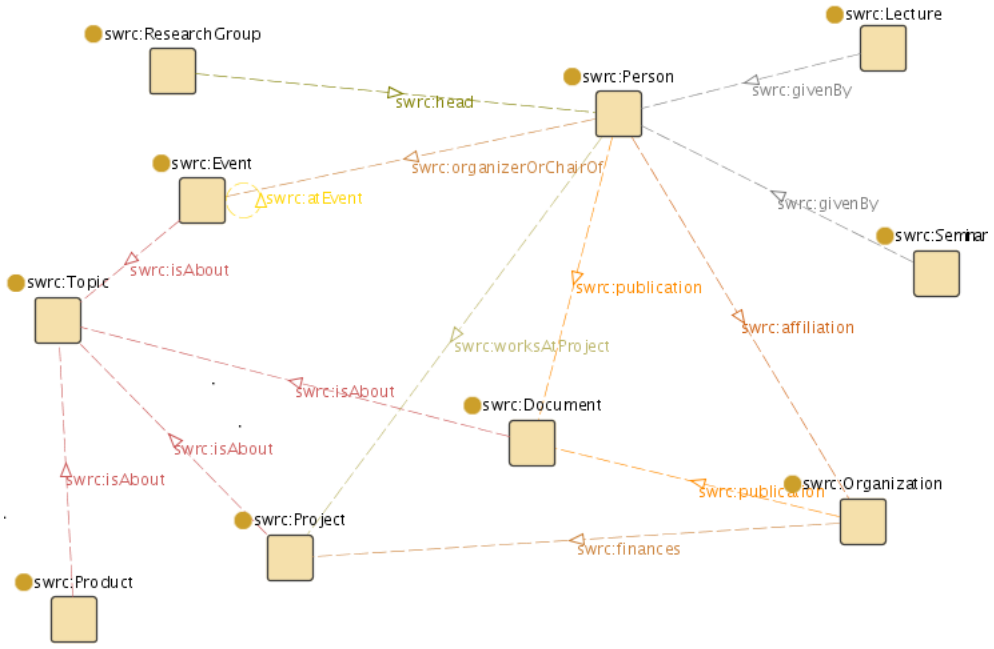

Figure 21: An example of relations between different concepts of the SWRC ontology.

In spite of being fairly complete in what relates to Research Communities, the SWRC ontology does not go into great detail when it regards to the description of one particular type of event that is primordial to the scientific

community: the Scientific Conference. For that end, the SWC ontology came into being.

4.3.1 The SWC Ontology

The SWC Ontology was developed for the series of European and International Semantic Web Conferences, based on the idea that Semantic Web researchers should start "eating their own dog food"[48], meaning that they should be the first to use the tools developed by themselves. The idea of the authors was to provide a description of all the concept and material associated with Semantic Web conferences, in RDF. In order to achieve their ends, they developed the SWC (Semantic Web Conference) ontology. Apparently, the idea of the creators of this ontology was not very different from that that is described in this document as the "core" of a Semantic Scientific Web of Researchers (the "three main players"):

> "When describing an academic conference, the three object types of greatest interest are people, events, and publications. People may take the role of paper authors, delegates at the conference, and committee members. Events may consist of talks (e.g. paper or poster presentations), conference sessions in which several papers are presented, or entire tracks. Various kinds of non-academic events also occur, such as meals, social events, and even coffee breaks. Publications can consist of full papers and poster/demo papers, plus a bound or electronic volume of the entire conference proceedings. In addition, artefacts such as sets of slides can be of great value, whilst not being formally published. Apart from this core set of data, other kinds of information, such as rooms within the conference venue, or sponsoring organizations, can be relevant." [48]

The authors of the ontology started by analyzing already existing ontologies, such as the AKT Reference Ontology[120], and the conference ontologies developed by Jen Golbeck[121] and eBiquity[122]. The lack of expressiveness that the candidates suffered from, led them to build their own, from scratch. "From scratch" is not the most adequate term, since the authors decided to reutilize the maximum amount of existing material from other ontologies, naturally adopting the classes that people were already using most. So,

[120] `http://www.aktors.org/ontology/extension`
[121] `http://www.mindswap.org/~golbeck/web/www04photo.owl`
[122] `http://ebiquity.umbc.edu/ontology/conference.owl`

the SWC ontology builds on the top of FOAF, SWRC (mainly for publications) and iCalendar ontologies. Naturally, some decisions had to be taken, concerning the concepts that overlap, among the imported ontologies: for instance, `foaf:Person` was chosen over `swrc:Person`, probably because of the widespread adoption that the former has had. The unification of data that is specified using these two classes is usually addressed through the introduction of mapping relationships between them. These mapping relationships are fundamental, so that reasoning can act over heterogeneous repositories of information[43][51].

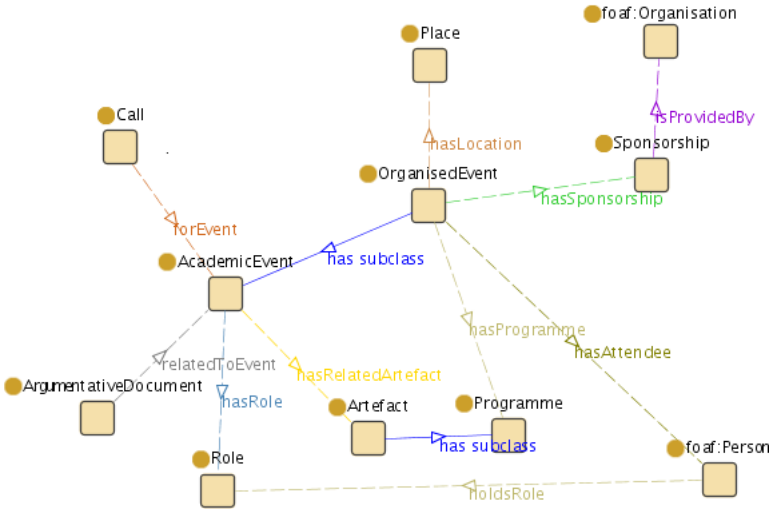

Figure 22: Some classes and properties from the SWC ontology.

SWC expands the publication types defined in SWRC with classes that target conference-specific matters (like `swc:InvitedPaper`), and define new ones, based on `foaf:Document`, that represent published resources that go beyond "traditional" articles, like slide sets, system demonstrations/descriptions, and posters. In terms of event representation, SWC has a tremendous expressive power, exemplified through fig. 23, allowing the specification of the most diverse events that happen in the lifecycle of a conference. The degree to which these classes can be "nested" one inside another (not represented in the diagram) introduces an additional level of complexity, allowing the expression of complex structures (i.e. a conference with different sessions, being that each of them contains different presentations of papers and posters, as well as panels and tutorials; included can be as well multiple social events and meals). With recourse to these mechanisms, a complete conference timetable

can be constructed.

Figure 23: The expressive power of the SWC ontology is clearly visible through the number of classes devoted to different kinds of events.

4.3.2 Describing Information from Indico

It is inevitable to notice some similarities between the metadata that can be described by SWRC/SWC, and the kind of information that Indico stores. The following table presents a synopsis of some kinds of events that a conference can have, and the support provided from both Indico and the SWC ontology.

Feature	Indico	SWC
Paper Presentations	no[123]	yes
Poster Presentations	yes	yes
Panels	no	yes
Tracks	yes	yes
Sessions	yes	yes
Tutorials	no	yes
Demonstrations	no	yes
Breaks	yes	yes
Meals	no	yes
Conference opening/closing	no	yes
Social Events	no[124]	yes

Table 1: Comparing features from Indico and SWRC

The SWC ontology exceeds Indico in terms of expressiveness in the classification of events. This is unfortunate, since no possible translation of information from the Indico database to RDF will ever use the ontology to its full potential. This is one clear example of a situation where the schema of a "legacy" database constrains the quality of the generated information.
The case of Indico was used as an experiment to find out if OURSE would perform well when applied to the particular demands of such a Scientific Web. Some harder concepts like event nesting and linking were experimented, and the result was very positive. Listings 26 and 27 show N3 outputs of information that was converted to Linked Data by OURSE. Several other resource properties could be included, but they would probably add no extra complexity to the problem.
Similarly, SPARQL queries were successfully performed over the data, using the SPARQL endpoint from the OURSE Data Server prototype. Listing 28 displays one simple query, that does a search by keyword, and listing 29 shows the result of the same query. The example consists in finding events by specifying related keywords, something that requires only a few SPARQL lines.

4.4 Scio

The center of the proposed architecture would be a "mashup" service, codenamed "Scio", the word for "knowledge" in the Esperanto language. Scio would be responsible for aggregating data that circulates in the Semantic Web, making some trivial "smushing" operations over it, and transforming

Listing 26 An example of a OURSE-generated graph of Indico information (about a conference), expressed in the SWC ontology (with the aid of iCal and DC).

```
@prefix _3: <http://www.w3.org/2002/12/cal/ical#>.
@prefix _4: <http://data.semanticweb.org/ns/swc/ontology#>.
@prefix _5: <http://purl.org/dc/elements/1.1/#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.

 <http://localhost:8080/events/3580> a _4:ConferenceEvent;
     rdfs:label "CHEP 07";
     _4:isSuperEventOf <http://localhost:8080/events/3580/sessions/13>,
         <http://localhost:8080/events/3580/sessions/14>,
         <http://localhost:8080/events/3580/sessions/15>;
     _5:keyword "CHEP";
     rdfs:comment "Computing in High Energy and Nuclear Physics";
     _3:dtend "2007-09-09T12:00:00"^^<http://www.w3.org/2001/XMLSchema#date>;
     _3:dtstart "2007-09-02T08:00:00"^^<http://www.w3.org/2001/XMLSchema#date>.
```

Listing 27 Another example of a OURSE-generated graph of Indico information, this time about a session that is contained inside the previous conference.

```
@prefix _3: <http://www.w3.org/2002/12/cal/ical#>.
@prefix _4: <http://data.semanticweb.org/ns/swc/ontology#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.

 <http://localhost:8080/events/3580/sessions/14> a _4:PaperSession;
     rdfs:label "Online Computing";
     _4:isSubEventOf <http://localhost:8080/events/3580>;
     rdfs:comment """CPU farms for high-level triggering; Farm configuration and
run control; Describing and managing configuration data and
conditions databases; Online software frameworks and tools;
online calibration procedures.""";
     _3:dtend "2007-09-06T18:00:00"^^<http://www.w3.org/2001/XMLSchema#date>;
     _3:dtstart "2007-09-03T14:00:00"^^<http://www.w3.org/2001/XMLSchema#date>.
```

Listing 28 A SPARQL request that should return the name of all the conferences associated with the "CHEP" keyword.

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dc: <http://purl.org/dc/elements/1.1/#>
SELECT ?l WHERE {
  ?conf rdfs:label ?l ;
        dc:keyword "CHEP".
}
```

Listing 29 The result of the previous query.

```
{
   "head" : {
        "vars" : [
             "l"
         ]
    },
    "results" : {
          "ordered" : false,
          "distinct" : false,
          "bindings" : [
               {
                   "l" : {"type": "literal",
                   "xml:lang" : "None",
                   "value" : "CHEP 07"}
                }
           ]
    }
}
```

it into a representation that could be easily visualized by users. The feature set was heavily inspired by web-based social networks, and by previously implemented services, like Flink, that explore the social component of the semantic web.

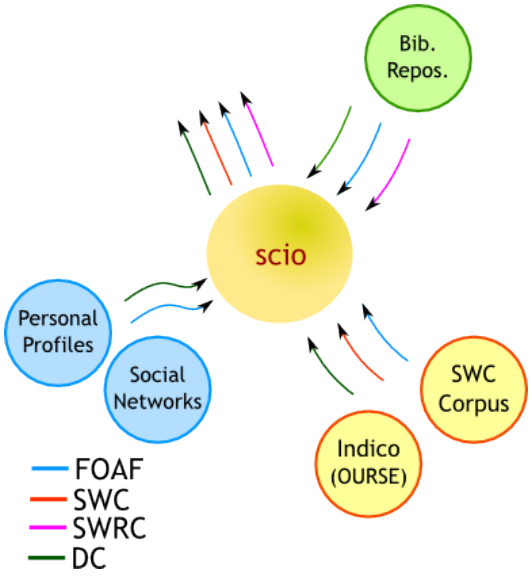

Figure 24: Scio, and its different interactions, through "ontology channels"

4.4.1 Features

Scio should provide its researcher users with:

- A profile, with:
  - Personal data: name, location, age;
  - List of publications, with the publication titles, links to co-author profiles, the event it relates to (in case it belongs to the proceedings of some event) and link to the abstract and electronic version of the text, if available;
  - List of attended events, by chronological order (and a link to the "profile" of each event);
  - A "will attend" list, where the user should be able to add events (conferences) he/she will attend;

  - List of research topics, generated from both the publications and events;
  - List of "social connections", generated from co-authorship information, and explicit input as well - a "graph plot", similar to the one employed by Flink could be used in order to provide a better user experience;

- A way of filtering out profile information that was gathered but should not be shown;
- A means to add and remove sources of profile information (i.e. FOAF profiles, other RDF specifications of publications by the author, etc...);
- Authentication - a way to guarantee that the user is really who he/she says;

On the other hand, anonymous users should be allowed as well to consult all the mentioned profile information, and browse it through hyperlinks. Some other services should be provided to unauthenticated users:

- A list of upcoming events (conferences), linking to the detailed description of each of them;
- A profile-like, detailed description of each event, including:
  - General information like title, dates and location (and "parent event", in case the event is a sub-event of some other) ;
  - A list of sub-events (sessions, presentations) that can be expanded to show the contents of each sub-event, in a tree-like way. Each sub-event should be linked to its own "event profile";
  - For presentations, links to the author profiles and electronic versions of related papers/presentations, if available;
  - A list of research topics associated with the event;

In addition to the normal HTML-based representation of all these "profiles", RDF versions should be generated as well, so that the "mashup" can be used by semantic applications.

4.4.2 Problems

A platform of this nature involves a series of complexities that are not at all easy to solve:

- The question of duplicate identities - The metadata that comes from publication repositories does not always excel for its quality. For instance, in RKB, there are different entries for Tim Berners-Lee, that use slight variations of his name. The question gets even more complex when data from other sources has to be merged as well, and no IFPs exist. The way is free for the specialists in heuristic matching algorithms to try sorting out how to solve this issue;

- A store with such an amount of triples would be hard to update - The Semantic Web of Linked Data is not supposed to be static: graphs evolve, with the addition of new properties and correction of old ones. Keeping track of these changes is not an easy task at all, and the situation gets even worse when data that is already unified and filtered has to be matched with raw, "unsmushed" and unfiltered up-to-date data. Not to mention that such an update mechanism would require storage of metadata about the consulted graphs themselves (time of last update, source, etc...);

- Implementing trust mechanisms would be a challenge - The "trust" layer that stands at the top of the "Semantic Web Layer Cake" (just below the application layer) is not yet fully implemented. Actually, there are still no standards for implementing trust mechanisms in the Semantic Web. In fact, a large discussion is taking place, at the time of the writing of this document, among the FOAF community, on possible mechanisms of authentication for FOAF "private" profiles[125]. On the other hand, the problem of revocation of information is probably the one that would create the greatest obstacles to the Scio concept: how to assert that something declared in some harvested graph is really the truth (i.e. that Bob has co-authored a paper with Alice)? This problem has already been briefly mentioned in this document, and is under careful study by the research community[47]. Some Semantic Web-based applications like DBin[126] are already taking advantage of some of these ideas.

[125] See `http://lists.foaf-project.org/pipermail/foaf-dev/2008-March/009037.html`

[126] `http://dbin.org/`

4.5 Conclusion

The concept of a "mashup" like Scio, that be capable of aggregating and linking information from different sources, is still some steps away from us. These are not steps that require strong technological progress and massive amounts of work, but instead an effort of standardization that is yet to be achieved. The main problem would be implementing authentication and trust: how to guarantee that a user is really who he/she says he/she is? How to know that all the publications he/she is referencing from his/her profile are not fake, or weren't written by someone else? How to prove that he/she was actually at an event that he/she claims to have been to? These are questions that haunt even the "old web" and the stream of social applications that have been appearing through the years. It is true that repositories like CiteSeer or DBLP will normally be trustful sources, but the advantage of the Semantic Web is precisely on linking information from different sources and extracting something useful from that.

5 Evaluation and further work

It can be said that this research project was a success in terms of the accomplishment of the objectives that were initially proposed:

- The question of Scientific Semantic Webs was studied in a broad way, through the work of several authors, and the analysis of different tools that are already available;
- The main "players" were identified, and data sources that provide semantic information for those concepts were found;
- Ontologies for describing such concepts where identified from the existing base of RDF ontologies;
- The existing knowledge base on scientific events was enriched, through the development of OURSE, that allows data that is stored using the Indico conference management system to be exported as RDF;
- OURSE has been designed with reutilization in mind, allowing its employment in different contexts and data sources - this added new value to the Semantic Web technological base, by exploring a territory that had not been explored before;

However, there's still a lot to do, on the topic Scientific Networks of Researchers, but also on some of the concepts introduced by this document (like OURSE). These points could be explored in more detail in the context of future works:

- The OURSE translation language should be expanded, in order to accommodate constructs for different databases, including document-based stores. Specifically, the *Object Providers* should be diversified;
- The OURSE Data server prototype could be expanded with a mechanism for handling source updates, refreshing its data sources periodically, and storing metadata about the harvested graphs. This should allow enough flexibility for the server to be used in applications that require different degrees of database-triplestore consistency;
- The Indico platform could be used as a base to extract more information that concerns scientific communities, like data about lectures and meetings. This is supported by the SWRC ontology, and easily translated using OURSE;

- On the other hand, some mechanisms of Indico could be revised, not only in order to allow RDF data to be extracted, but also as a way to provide a better quality of service in the web interface - as an example, the "location" of an event is currently a simple string with a place's name, but it could perfectly be a value of latitude/longitude or a reference to GeoNames, for the sake of data quality. It would only be a matter of providing a usable interface for the user (a map or an auto-completion text box). Naturally, this is something that only the Indico developers can take care of;

- Some work could be developed in using SKOS to describe the taxonomy of categories that builds up an Indico event tree - this would help locating the events in terms of the topics they refer to, thus improving the quality of search mechanisms;

One or several of these points may be of interest to anyone looking into adding some value to the work that is described in this document.

6 Summary and conclusion

It is now time to make a brief summary of all the work that was developed through (and in the context of) this document, and to draw some conclusions about the main topics that were explored.
This document tried to provide, through all these pages, an overview of the Semantic Web, and its particular application as a platform for networks of researchers to grow. Several technologies and concepts were presented, from the most basic concepts of the RDF framework, to fairly complex ontologies, and some efforts of integration between them. After introducing the RDF model, and quickly explaining the basics of notation formats, ontology and query languages, the problem of Research Networks was introduced. The main objective was to provide a "zoomed out" perspective of the different tools that the Semantic Web provides for the construction of such communities, and trying to find a way to connect the main concepts that they encompass: researchers, publications and events. The conclusion that the availability of event information in RDF was scarce was eventually reached, and the Indico platform for conference management was introduced as a possible data source that would fill this gap. In order to provide the information contained in Indico repositories as RDF, the OURSE translation language was conceived, providing a direct mapping between information contained in object databases and the RDF model. A working prototype, pyOURSE, was successfully tested with data from an Indico database, and is currently available for download on the Web. Finally, the pieces of the "jigsaw" were connected, through the proposition of a "mashup" architecture, Scio, that would unify all the data sources and provide a user-friendly (but RDF-exporting) application, that would present the Web of Researchers to the general public.

6.1 The Semantic Web

Almost one decade after Tim Berners-Lee wrote the words that opened §1.2 in his best-selling book "Weaving the Web"[10], the Semantic Web project has yet to achieve the vision of the "intelligent agents" that its architect has conceived. However, the project seems to be acquiring progressively more momentum, with new technologies and tools appearing every day, and a set of standards that is gradually getting sound. From RDF/XML to N3, from DAML+OIL to OWL, and from the Dublin Core to SWRC and FOAF, the Semantic Web has grown in complexity through the times, and proved to be similar to an evolutionary ecosystem, where new standards evolve from

hybridization (i.e. the case of the query languages), and the community of developers selects the ones that should be kept or discarded. Developing for the Semantic Web is, first of all, a social experience, that requires the consultation of different sources, and a certain degree of interaction with other developers. The agility that this collective development process possesses has such a magnitude, that it would be possible that some of the technologies and standards that are referred through this document would get obsolete a few months after its publication. On the other hand, this agility can create some good surprises, and it is not seldom that a developer bumps into a solution to his problem, made by someone else, just a few days after the problem appears. This active network of contributors gets synchronized through the instruments of the social web (blogs, wikis, social networks...), but as well through older technologies (IRC and mailing lists). The amount of knowledge that is still to be created around the Semantic Web is so vast, that universities and other research institutions take the chance to explore this recently-discovered territory, fueling the community with "brain mass". The low cost of Semantic Web research (that usually requires no more than people and computers) encourages companies to invest on this emerging area, expecting a future return in the form of profit. Giants like Yahoo!, Oracle, HP and Sun are currently investing resources on the Semantic Web, and a new generation of SW-oriented companies is slowly forming. The truth is that the vision of a world-wide graph of structured knowledge still doesn't hit the user's retina with the same impact as the developer's. Neither web users nor most of the web developers are yet convinced that the Semantic Web is such a "big deal" that they should start producing RDF. The need for a "killer app" for the semantic web, an application that takes advantage of its full potential, and displays real, user-translated benefits that cannot be achieved with older technology, is a premise for such a revolution to happen. The rules of the game are know (and fair): technology is only useful if it translates to benefits. It is the responsibility of Semantic Web Researchers and Developers to make sure that such a "messianic prophecy" is fulfilled. Some skeptics believe that it won't take long before the Semantic Web suffers from the same problems as its older counterpart: spam and "useless content" seem to be impossible to control, and the highly interconnected nature of the Semantic Web seems to be a potentially good target. Besides that, the problem doesn't seem to be only the good faith of content producers. As Cory Doctorow remarks in an essay that was peculiarly called "Metacrap: Putting the torch to seven straw-men of the meta-utopia"[127], there are prob-

[127] Available at `http://www.well.com/~doctorow/metacrap.htm`

lems that are inherently human, such as the inability to describe things in a non-biased way; and these problems can compromise what he calls the "meta-utopia", an "Utopia" of meta-data. The defenders of the Semantic Web point to the second layer (from the top) of "the cake" as the solution to these problems: "trust" will always be taken in account in a Semantic Web of Linked Data, and this "trust layer" will make sure that users can filter out the noise. There are, as well, the implications of a possible step (taken by the Web as a whole) towards the Semantic Web Vision, in terms of the evolution of society: concerns over censorship, anonymity, and financial matters still remain, and Berners-Lee is the first to admit the uncertainty about the consequences of the project's implementation:

> *"There could be spontaneous order or instability: Society could crash, much as the stock market crashed in 1987 because of automatic trading by computer. [...] To ensure stability, any complex electronic system needs a damping mechanism to introduce delay, to prevent it from oscillating too wildly. [...] We may be able to build [damping mechanisms] into the Semantic Web of cooperating computers - but will we be able to build them into the web of cooperating people?"* [10]

The truth is that every major development in human communications brought with it a certain amount of uncertainty, and the fear of the consequences. The Semantic Web is no exception, with its own strengths and weaknesses.
If one looks towards the future of computer science, the approach of Berners-Lee can be interpreted as an assumption that the realization of a "Hard" Artificial Intelligence is still far away (or simply impossible), and humans should, then, make information "understandable" by artificial agents, by decreasing the level of ambiguity and noise that currently circulates on the web, and providing properly structured data. It is indeed true that the progresses in artificial natural language understanding are far away from reaching the science-fiction vision of computers that answer to requests that are issued by a human operator, by crawling the web of documents and finding a proper answer: ambiguity, contradiction, trust... there are so many problems (even excluding the question of natural language) that keep machines away from the answers, that it is valid to think that the solution is perhaps to provide them with something they have a better probability to be able to deal with.

6.2 OURSE

OURSE proved not only to be an interesting tool for the "semantification" of object-based repositories, but provided as well some ideas for the future: a possible universal format for specifying the mapping between the OO and RDF paradigms is just one of them. In fact, some existing tools, like Oort and ActiveRDF, already provide a way to translate RDF graphs to objects. The introduction of a common "language" for specifying how this transformation (and the inverse one) should be done could bring a degree of uniformization to this matter. However, these are just ideas that will need to survive a process of maturation before deserving serious consideration. Concerning the translation language, the implementation of pyOURSE proved that the former is indeed usable and expressive enough for the task of making the Indico database available as RDF. This is not, though, by any means, a guarantee that the translation language is expressive enough for more advanced usage. The author is perfectly conscious that the problem that served as the case-study for OURSE (the case of Indico) may not be the most complex of its kind. OURSE has been developed more as a proof of concept than as a final solution, expected to be quickly adopted by everyone. The author is totally aware that Open Source projects like Indico have too many concerns for now, much more important than adapting themselves to a promised Semantic Web that still hasn't proved to deserve such a priority. On the other hand, that time will hopefully come, and, by then, these projects will have several tools at their disposal, in order to generate the required triples. Maybe one of these will be OURSE, or a more advanced solution, in some way based on it. The turning point will eventually arrive when the organizations, developers and users that are involved with large information repositories realize that the potential of information that they store is tremendous, and that releasing all this data it in a way that the Semantic Web community can reuse will benefit everybody, since new, innovating applications can be constructed. The only guarantee that can be stated now is that development on OURSE will, in the short term, go on, and new database systems (object-oriented and document-oriented as well) will be targeted.

6.3 The "Web of Science"

It was never an objective that an absolute and specific solution would be reached for the problem of Researcher Networks. In fact, if such a solution were possible to reach with such a limited amount of resources, it would have already been realized by now. The truth is that the Semantic Web is still

growing and maturing, and not all the layers from the famous "cake" are complete. In spite of this, some ingenious partial solutions to this problem have been mentioned by this document: Flink is one of them, and probably the one that resembles most the functionality that the proposed Scio "mashup" would perform. Other projects, like *openacademia*, provide less of a "unified perspective" over the problem, but nonetheless are successful in improving the quality of web applications for research, through the employment of semantic technologies. Even if there is still no "killer application" like a global Social Network of scientists based on Linked Data, triplestores start to grow in size and in number. At the same time, ontologies for the description of such information are now maturing, and specifications like those of FOAF, SWRC and SWC ontologies are gradually getting solid. The community of Semantic Web Researchers and Developers never fell into the temptation of setting "standard ontologies" and other "dogmas". In fact, the history of Semantic Web development has always been based on an agile process of continuous feedback, and a strategy of letting diversification happen, and letting the "environment" select which technologies would pass to the "next phase" and which ones would be abandoned. Like in every science, the ideas that survive are those that seem logical and coherent with reality. That's what is emerging from the development of this new concept of web: a "Web Science" that doesn't study any particular kind of subjects, but rather the vast amount of information that is contained by their interactions.

Glossary

(Screen) Scraping A process that involves retrieving information from the output of an application, as shown to a human user. 1

Hypertext Basically text that can contain "links" to other documents. The largest hypertext system in existence today is the World Wide Web. 1

iCalendar A standard for calendar data exchange, widely known as "iCal". 1

Mashup In the context of the web, a "mashup" is an application that combines data from different sources, providing a unified service based on those data. 1

REST Representational State Transfer - An architectural style for web services, based on URIs and stateless protocols. 1

RSS Family of Webfeed formats, used to publish frequently updated content. 1

SHA1 (sum) Cryptographic hash function that is normally used to compute a representation (digest) of a data sequence. 1

SOAP A protocol for exchanging XML messages about HTTP services, over a network. 1

vCard A file format standard for electronic business cards. 1

XPath XML Path Language - A language for addressing nodes inside XML documents. 1

Acronyms

(O)ODBMS Object-oriented Database Management System. 1

AJAX Asynchronous JavaScript and XML. 1

API Application Programming Interface. 1

CERN *Organisation Européenne pour la Recherche Nucléaire* - European Organization for Nuclear Research. 1

CFP Call For Papers. 1

DBMS Database Management System. 1

DLG Directed Labelled Graph. 1

FLOSS Free, Libre and Open Source Software. 1

FOAF Friend of a Friend. 1

GPL GNU General Public License. 1

HTML Hypertext Markup Language. 1

HTTP Hypertext Transfer Protocol. 1

IRC Internet Relay Chat. 1

JSON JavaScript Object Notation. 1

LOD Linking Open Data. 1

N3 Notation3 RDF serialization notation. 1

ODBC Open Database Connectivity. 1

OO Object-Oriented. 1

OOP Object-Oriented Programming. 1

ORDBMS Object-Relational Database Management System. 1

ORM Object-Relational Mapping. 1

OURSE Object Utilizer for RDF Semantic Exporting. 1

OWL Web Ontology Language. 1

RDBMS Relational Database Management System. 1

RDF Resource Description Framework. 1

RSS Universal Resource Locator. 1

SPIRES Stanford Physics Information Retrieval System. 1

SQL Structured Query Language. 1

SW Semantic Web. 1

URI Universal Resource Identifier. 1

URL Universal Resource Locator. 1

URN Universal Resource Name. 1

W3C World Wide Web Consortium. 1

WGS World Geodetic System. 1

WWW World Wide Web. 1

XHTML Extensible Hypertext Markup Language. 1

XML Extensible Markup Language. 1

XSLT Extensible Stylesheet Language Transformations. 1

ZODB Zope Object Database. 1

References

[1] Ben Adida and Mark Birbeck. RDFa Primer. W3C working draft, W3C, March 2008. http://www.w3.org/TR/2008/WD-xhtml-rdfa-primer-20080317/.

[2] A.-L. Barabasi, H. Jeong, Z. Neda, E. Ravasz, A. Schubert, and T. Vicsek. Evolution of the social network of scientific collaborations. *Physica A: Statistical Mechanics and its Applications*, 311(3-4):590–614, August 2002.

[3] Dave Beckett. RDF/XML syntax specification (revised). W3C recommendation, W3C, February 2004. http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/.

[4] Dave Beckett. Turtle - terse RDF triple language. Available at http://www.dajobe.org/2004/01/turtle/, 2007.

[5] Dave Beckett and Jan Grant. RDF test cases. W3C recommendation, W3C, February 2004. http://www.w3.org/TR/2004/REC-rdf-testcases-20040210/.

[6] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, 284(5):34–43, May 2001.

[7] Tim Berners-Lee. Notation 3. Desing issue, W3C, 1998. http://www.w3.org/DesignIssues/Notation3.html.

[8] Tim Berners-Lee. Rdf anonymous nodes and quantification. Desing issue, W3C, 1998. http://www.w3.org/DesignIssues/Anonymous.html.

[9] Tim Berners-Lee. Linked data. Desing issue, W3C, 2006. http://www.w3.org/DesignIssues/LinkedData.html.

[10] Tim Berners-Lee and Mark Fischetti. *Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web.* Harper Collins Publishers, New York, 1999.

[11] Chris Bizer and Richard Cyganiak. The TriG Syntax. Specification, Freie Universität Berlin, July 2007. http://www4.wiwiss.fu-berlin.de/bizer/TriG/.

[12] Dan Brickley. Basic Geo (WGS84 lat/long) Vocabulary. W3C - Semantic Web Interest Group - Informal Specification, W3C, February 2006. http://www.w3.org/2003/01/geo/.

[13] Dan Brickley and Libby Miller. FOAF Vocabulary Specification. Namespace Document 2 Nov 2007, FOAF Project, 2007. `http://xmlns.com/foaf/0.1/`.

[14] Jeen Broekstra and Dave Beckett. SPARQL query results XML format. W3C recommendation, W3C, January 2008. http://www.w3.org/TR/2008/REC-rdf-sparql-XMLres-20080115/.

[15] Kurt Bryan and Tanya Leise. The $25,000,000,000 Eigenvector: The Linear Algebra behind Google. *SIAM Rev.*, 48(3):569–581, 2006.

[16] Jörg Garbers Chris Bizer, Richard Cyganiak and Oliver Maresch. The D2RQ Platform v0.5.1 - Treating Non-RDF Relational Databases as Virtual RDF Graphs. Specification, Freie Universität Berlin, October 2007. `http://www4.wiwiss.fu-berlin.de/bizer/d2rq/spec/`.

[17] Kendall Grant Clark, Lee Feigenbaum, and Elias Torres. SPARQL protocol for RDF. W3C recommendation, W3C, January 2008. http://www.w3.org/TR/2008/REC-rdf-sparql-protocol-20080115/.

[18] Kendall Grant Clark, Elias Torres, and Lee Feigenbaum. Serializing SPARQL query results in JSON. W3C note, W3C, June 2007. http://www.w3.org/TR/2007/NOTE-rdf-sparql-json-res-20070618/.

[19] N2 Wiki Contributors. RDF JSON Specification - N2 Wiki, March 2008. `http://n2.talis.com/mediawiki/index.php?title=RDF_JSON_Specification&oldid=1040`.

[20] Douglas Crockford. RFC4627: JavaScript Object Notation. RFC, Internet Society, 2006. `http://www.apps.ietf.org/rfc/rfc4627.html`.

[21] F. Dawson and T. Howes. RFC2426: vCard MIME Directory Profile. RFC, Internet Society, 1998. `http://www.apps.ietf.org/rfc/rfc2426.html`.

[22] F. Dawson and D. Stenerson. RFC2445: Internet Calendaring and Scheduling Core Object Specification (iCalendar). RFC, Internet Society, 1998. `http://www.apps.ietf.org/rfc/rfc2445.html`.

[23] Steven DeRose and James Clark. XML path language (XPath) version 1.0. W3C recommendation, W3C, November 1999. http://www.w3.org/TR/1999/REC-xpath-19991116.

[24] ECMA. ECMAScript language specification, December 1999. ECMA Standard 262, 3rd Edition.

[25] Dieter Fensel, F. Van Harmelen, Ian Horrocks, Deborah L. McGuinness, and P. F. Patel-Schneider. Oil: an ontology infrastructure for the semantic web. *Intelligent Systems*, 16(2):38–45, 2001.

[26] Jose Pedro Ferreira. Improving the Indico Framework at the European Organization for Nuclear Research - Internship Report - LEIC 2006/2007 (FEUP). Technical Report CERN-IT-Note-2007-030, CERN, Geneva, Aug 2007.

[27] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1 (RFC 2616). RFC, Internet Society, 1999. available at `http://www.ietf.org/rfc/rfc2616.txt`.

[28] Tim Furche, Benedikt Linse, François Bry, Dimitris Plexousakis, and Georg Gottlob. RDF Querying: Language Constructs and Evaluation Methods Compared. In Pedro Barahona, François Bry, Enrico Franconi, Nicola Henze, and Ulrike Sattler, editors, *Reasoning Web*, volume 4126 of *Lecture Notes in Computer Science*, pages 1–52. Springer, 2006.

[29] Hugh Glaser and Ian C. Millard. RKB Explorer: Application and Infrastructure. In Jennifer Golbeck, Peter Mika, editor, *Proceedings of the Semantic Web Challenge 2007, Busan, Korea*, November 2007.

[30] Jerry Grossman. Facts about Erdös Numbers and the Collaboration Graph, December 2006. Available on-line at `http://www.oakland.edu/enp/trivia.html`.

[31] Ramanathan V. Guha and Dan Brickley. RDF vocabulary description language 1.0: RDF schema. W3C recommendation, W3C, February 2004. http://www.w3.org/TR/2004/REC-rdf-schema-20040210/.

[32] Martin Hepp and Jos de Bruijn. GenTax: A Generic Methodology for Deriving OWL and RDF-S Ontologies from Hierarchical Classifications, Thesauri, and Inconsistent Taxonomies. In Enrico Franconi, Michael Kifer, and Wolfgang May, editors, *ESWC*, volume 4519 of *Lecture Notes in Computer Science*, pages 129–144. Springer, 2007.

[33] Aidan Hogan and Andreas Harth. The ExpertFinder Corpus 2007 for the Benchmarking and Development of ExpertFinding Systems. In *Pro-*

ceedings of the 1st International ExpertFinder Workshop, Berlin, January 2007.

[34] Open Archives Initiative. The open archives initiative protocol for metadata harvesting, 2004.

[35] Eric Vitiello Jr. and Ian Davis. Relationship: A vocabulary for describing relationships between people, 2004. `http://purl.org/vocab/relationship`.

[36] Gregory Karvounarakis, Sofia Alexaki, Vassilis Christophides, Dimitris Plexousakis, and Michel Scholl. RQL: A Declarative Query Language for RDF. In *WWW '02: Proceedings of the 11th international conference on World Wide Web*, pages 592–603, New York, NY, USA, 2002. ACM Press.

[37] Rohit Khare. Microformats: The next (small) thing on the semantic web? *IEEE Internet Computing*, 10(1):68–75, 2006.

[38] Michel C.A. Klein, Peter Mika, and Stefan Schlobach. Approximate instance unification using roughowl: Querying with similarity in openacademia, 2007. Available at `http://www.cs.vu.nl/~mcaklein/papers/rowl-openacedemia.pdf`.

[39] Graham Klyne and Jeremy J. Carroll. Resource description framework (RDF): Concepts and abstract syntax. W3C recommendation, W3C, February 2004. `http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/`.

[40] Neal Leavitt. Whatever Happened to Object-Oriented Databases? *IEEE Computer*, 33(8):16–19, 2000.

[41] Ashok Malhotra. W3C RDB2RDF Incubator Group. Available at `http://www.w3.org/2005/Incubator/rdb2rdf/`, March 2008.

[42] Ashok Malhotra and Paul V. Biron. XML schema part 2: Datatypes. W3C recommendation, W3C, May 2001. `http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/`.

[43] P. Mika, M. Klein, and R. Serban. Semantics-based publication management using RSS and FOAF. In *Proceedings, Semantic Wiki 2006 (Submitted)*, 2006.

[44] Peter Mika. Flink: Semantic Web Technology for the Extraction and Analysis of Social Networks. *Journal of Web Semantics*, 3(2):211–223, 2005.

[45] Pete Johnston Mikael Nilsson, Andy Powell and Ambjörn Naeve. Expressing Dublin Core metadata using the Resource Description Framework (RDF). DCMI Recommendation, DCMI, January 2008. `http://dublincore.org/documents/dc-rdf/`.

[46] Eric Miller and Frank Manola. RDF primer. W3C recommendation, W3C, February 2004. `http://www.w3.org/TR/2004/REC-rdf-primer-20040210/`.

[47] Christian Morbidoni, Axel Polleres, and Giovanni Tummarello. Who the FOAF knows Alice? RDF Revocation in DBin 2.0. In Giovanni Semeraro, Eugenio Di Sciascio, Christian Morbidoni, and Heiko Stoermer, editors, *4th Italian Semantic Web Workshop SEMANTIC WEB APPLICATIONS AND PERSPECTIVES (SWAP 2007)*, volume 314 of *CEUR-WS.org*, Bari, Italy, December 2007. CEUR.

[48] Knud Möller, Tom Heath, Siegfried Handschuh, and John Domingue. Recipes for semantic web dog food - the eswc and iswc metadata projects. In Karl Aberer, Key-Sun Choi, Natasha Noy, Dean Allemang, Kyung-Il Lee, Lyndon J B Nixon, Jennifer Golbeck, Peter Mika, Diana Maynard, Guus Schreiber, and Philippe Cudré-Mauroux, editors, *Proceedings of the 6th International Semantic Web Conference and 2nd Asian Semantic Web Conference (ISWC/ASWC2007), Busan, South Korea*, volume 4825 of *LNCS*, pages 795–808, Berlin, Heidelberg, November 2007. Springer Verlag.

[49] Uche Ogbuji. Versa: Path-Based RDF Query Language. *XML.com*, July 2005. `http://www.xml.com/pub/a/2005/07/20/versa.html`.

[50] E. Oren and R. Delbru. ActiveRDF: Object-oriented RDF in Ruby. In *Proceedings of the ESWC Workshop on Scripting for the Semantic Web*, 2006.

[51] Zhengxiang Pan, Abir Qasem, and Jeff Heflin. An investigation into the feasibility of the semantic web. July 2006.

[52] Eric Prud'hommeaux. Algae RDF Query Language. Language description, W3C, 2004. `http://www.w3.org/2004/05/06-Algae/`.

[53] Andy Seaborne and Eric Prud'hommeaux. SPARQL Query Language for RDF. W3C Recommendation, W3C, January 2008. `http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/`.

[54] Michael K. Smith, Deborah L. McGuinness, and Chris Welty. OWL web ontology language guide. W3C recommendation, W3C, February 2004. http://www.w3.org/TR/2004/REC-owl-guide-20040210/.

[55] York Sure, Stephan Bloehdorn, Peter Haase, Jens Hartmann, and Daniel Oberle. The SWRC ontology - Semantic Web for research communities. In Carlos Bento, Amilcar Cardoso, and Gael Dias, editors, *Progress in Artificial Intelligence — Proceedings of the 12th Portuguese Conference on Artificial Intelligence (EPIA 2005), December 5-8, 2005, Covilhã, Portugal*, volume 3803 of *Lecture Notes in Computer Science*, pages 218–231. Springer, Berlin–Heidelberg, Germany, DEC 2005.

[56] Roy T. Fielding Tim Berners-Lee and Larry Masinter. RFC3986: Uniform Resource Identifier (URI): Generic Syntax. RFC, Internet Society, 2005. `http://www.apps.ietf.org/rfc/rfc3986.html`.

[57] Jeffrey Travers and Stanley Milgram. An Experimental Study of the Small World Problem. *Sociometry*, 32(4):425–443, 1969.

[58] Frank van Harmelen and Deborah L. McGuinness. OWL Web Ontology Language Overview. W3C recommendation, W3C, Feb 2004. http://www.w3.org/TR/2004/REC-owl-features-20040210/.

[59] W3C. W3C Opens Data on the Web with SPARQL - powerful technology for querying distributed and diverse data, January 2008. Press Release, available at `http://www.w3.org/2007/12/sparql-pressrelease`.

[60] Norman Walsh and Ian Jacobs. Architecture of the world wide web, volume one. W3C recommendation, W3C, December 2004. `http://www.w3.org/TR/2004/REC-webarch-20041215/`.

[61] Wikipedia. Ontology (information science) — wikipedia, the free encyclopedia, 2008. [Online; accessed 22-March-2008].

[62] Wikipedia. World geodetic system — wikipedia, the free encyclopedia, 2008. [Online; accessed 23-March-2008].

[63] William E. Winkler. The state of record linkage and current research problems. Technical Report Statistical Research Report Series RR99/04, U.S. Bureau of the Census, Washington, D.C., 1999.