Trust on the Semantic Web

Submitted in fulfilment
of the requirements of the degree
Master of Science
of Rhodes University

Russell Andrew Cloran

August 7, 2006

Abstract

The Semantic Web is a vision to create a "web of knowledge"; an extension of the Web as we know it which will create an information space which will be usable by machines in very rich ways. The technologies which make up the Semantic Web allow machines to reason across information gathered from the Web, presenting only relevant results and inferences to the user.

Users of the Web in its current form assess the credibility of the information they gather in a number of different ways. If processing happens without the user being able to check the source and credibility of each piece of information used in the processing, the user must be able to trust that the machine has used trustworthy information at each step of the processing. The machine should therefore be able to automatically assess the credibility of each piece of information it gathers from the Web.

A case study on advanced checks for website credibility is presented, and the site presented in the case presented is found to be credible, despite failing many of the checks which are presented.

A website with a backend based on RDF technologies is constructed. A better understanding of RDF technologies and good knowledge of the RAP and Redland RDF application frameworks is gained. The second aim of constructing the website was to gather information to be used for testing various trust metrics. The website did not gain widespread support, and therefore not enough data was gathered for this. Techniques for presenting RDF data to users were also developed during website development, and these are discussed.

Experiences in gathering RDF data are presented next. A scutter was successfully developed, and the data smushed to create a database where uniquely identifiable objects were linked, even where gathered from different sources.

Finally, the use of digital signature as a means of linking an author and content produced by that author is presented. RDF/XML canonicalisation is discussed in the provision of ideal cryptographic checking of RDF graphs, rather than simply checking at the document level. The notion of canonicalisation on the semantic, structural and syntactic levels is proposed. A combination of an existing canonicalisation algorithm and a restricted RDF/XML dialect is presented as a solution to the RDF/XML canonicalisation problem.

We conclude that a trusted Semantic Web is possible, with buy in from publishing and consuming parties.

Acknowledgments

Thanks are due to those who have enabled my studies. This work would not have been possible without funding, and so deep gratitude is owed to Deutscher Akademischer Austausch Dienst (DAAD), the Centre of Excellence at Rhodes University, Rhodes University, and my family.

The Computer Science Department at Rhodes University has gained my respect and gratitude for their hard work and positive attitude at all times and in all aspects. Thank you to everyone in the department.

This work has been a result of input from many directions, not to mention the strong shoulders on which I've been privileged to stand. My thanks to those who have enabled my work, through their great contributions to freely available knowledge and work. This includes both the Semantic Web community and the open source community. Special thanks to those involved in the projects which have directly benefited this work, especially the FreeBSD, Apache, PHP and Redland projects.

All my peers in the computer science department also deserve my thanks. Special thanks are due to my friends, who have supported and encouraged me when I've needed it. Darb, Dave, David, Dominic, Guy, Ingrid, Yusuf, thanks. Special thanks to Jonathan and Hannah for the sharp eyes and useful comments.

My final thanks go to my research supervisor, Barry Irwin. Thanks for your encouragement, guidance and the occasional threat! Thanks for allowing me the freedom to explore the boundaries, and for letting me know the limits when I've gone too far. Thank you also for your friendship and hospitality throughout the process.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Currently, the World Wide Web (WWW, or Web) [21, 27] is an information space of interlinked documents, created mainly using HTML or XHTML. The Web is an evolution of the ideas and vision of Tim Berners-Lee, who envisioned this interlinked information space in the late 1980's [27], and brought it to fruition at CERN. The Web enjoyed almost instant popularity among the hypertext community [27], and the growth in popularity of the Internet as a whole can be linked with the growth of the Web – making the Web one of the Internet's killer applications.

The information on the Web is designed to be formatted for display by computers, and consumed by people. Search engines create searchable indexes of the information with little real understanding of the underlying content. On the other hand, it could easily be argued that the Web would be of much less utility if it were not for search engines such as AltaVista and Webcrawler in the past, and Google most recently. Whilst some search engines have been successful in exploiting some of the information which is implicit in the Web, as well as the metadata and semi-structured data within the documents forming the Web, these require vast resources and still allow for only limited specific queries[1]. An example of a service on the Web which uses some structure found in Web documents to enable specific queries is Google's Glossary feature [82], which allows a user to search for definitions of words or terms, as shown in Figure 1.1.

Generally, the metadata that does exist is often in a loosely structured format, with fragmented dialects and no simple way of gleaning a meaningful understanding of the metadata. Specific queries require either a good understanding of the information within the document and/or its metadata by a search engine. This understanding can be gained either by creating software

[1] Specific queries are queries which ask for a specific piece of information, such as "dentists in the Grahamstown area who have an open time slot in the next two days"

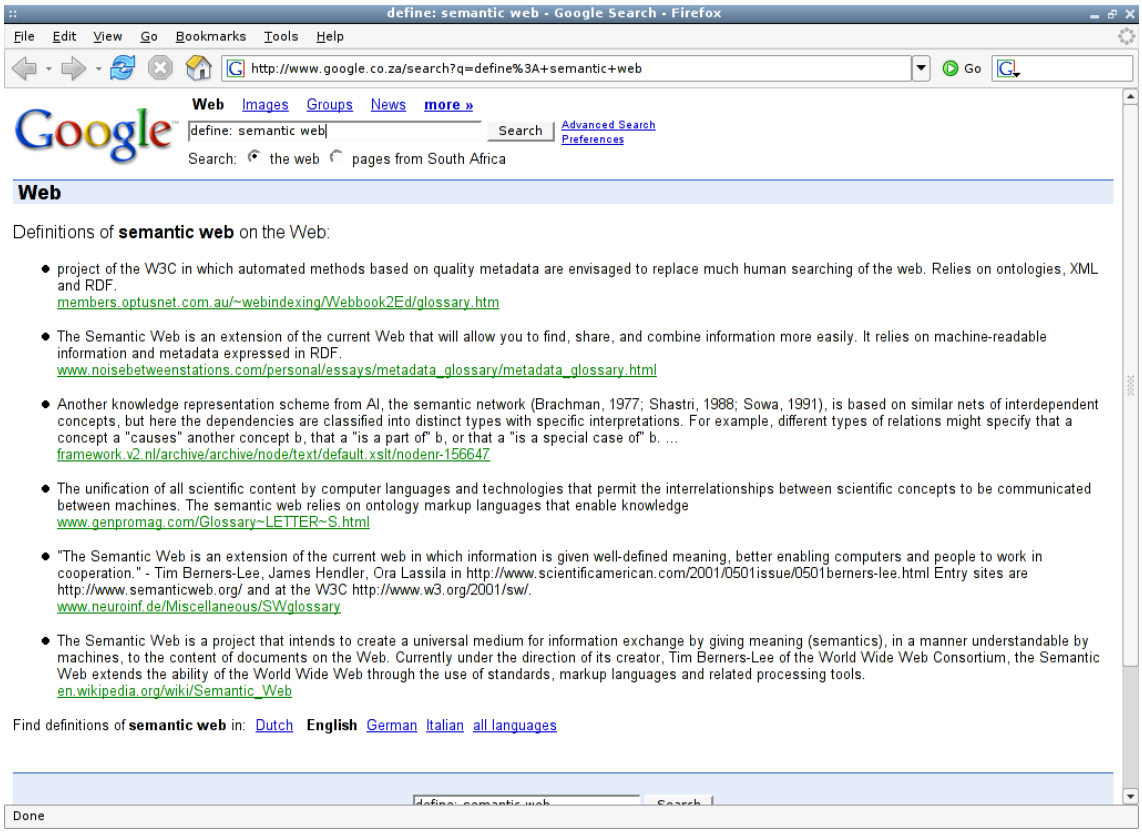

Figure 1.1: An example of using document structure to provide specific search, Google's Glossary feature for the term "Semantic Web"

which can understand the language and formatting of the document (such as Google has done to create its Glossary feature) and its metadata, or by creating a machine understandable language which is easy for machines to process.

The Semantic Web can be thought of as the next generation of the Web – an information space which can be well understood by computers, allowing computers to work better for their users [28]. This ”web of knowledge” has the potential to change the way people think about the Web, an already powerful phenomenon, and the way we interact with our computers. The Semantic Web differs from “Web 2.0”, an idea which has recently become popular, and promotes the usage of the Web as a platform, fully utilising the strengths of Web technologies. Web 2.0 is discussed further in Section 2.2. The move will be towards a scenario where everyday applications, such as calendaring applications, will become increasingly web-aware, processing information for the user, and acting on that information where possible. The Resource Description Framework (RDF) [89, 102, 114], was originally designed as a framework for representing simple metadata about resources on the Web, such as a document's author, title and publication date. RDF's simple structure is, however, also applicable as a machine understandable language when it is put together with RDFS [36] and OWL [16], which allow the semantically rich description of concepts. This suitability to representing information in a machine understandable manner has lead it to become the framework (and language, in RDF/XML [18]) which will help to realise the vision of the Semantic Web, or web of knowledge.

In order for users to accept a technology which allows many of their software applications to perform searches and tasks automatically to help with day to day activities, it is important that they are able to trust the information presented to them by the software they use. Thus, as a computer acts on the information which it has obtained from the Web, it is important that the accuracy of the information can be validated automatically, preferably without user intervention. Without this trust the user could be presented with incorrect information which may not be noticed or may be difficult to diagnose.

The concept of trust is difficult to describe. In Chapter 3, we show that the trusted and the trusting parties must be clearly defined, as well as the domain of trust. On a high level it can be described as the belief that a process will complete correctly. Research into the area of trust and its applicability in various computing systems is growing, with some research towards trust, trust networks, and their applicability to the Semantic Web.

1.1 Problem statement

If a machine understandable information space is to be created and used, its users must be able to trust the information which their user agents ("browsers") present to them. This is only possible if the user agent is able to determine the credibility of the information which it is to use for processing. This credibility assessment may be based on a number of factors, including the trustworthiness or credibility of the creator, be it a person, organisation or automated system.

The problem that this work addresses is that of automatically determining the trust which should be placed in information gathered from the Semantic Web.

There are two factors which greatly affect the ability of an agent to successfully gauge the credibility of a piece of information, based on its creator. Firstly, a link between the creator and the information must be created, and secondly, the trustworthiness of the creator must be assessed.

If the credibility or trustworthiness of the creator is to be used as a measure of the credibility of a particular fact or statement, then a strong link between the creator and the statement needs to be created, in order to verify the identity of the creator of a statement or fact. In terms of architecture, either a centralised or decentralised approach can be taken. A centralised system uses only one or a few trusted third parties to vouch for the trustworthiness of an author, while in a distributed system, trust is calculated by each agent when required. In centralised systems, this strong link is easily created by the trusted third party – the centralised system. In a decentralised system, such as the Semantic Web, a suitable strong link needs to be created in some other way. The use of cryptographic digital signature is a well tested method of ensuring this link in other domains, and may also be applicable with RDF. RDF allows a particular set of semantics to be represented in a number of ways using RDF/XML [18], and a canonical serialisation of RDF into RDF/XML is therefore desirable to make cryptographic digital signatures useful. This is discussed further in Chapter 7.

The second factor is that it is unlikely (and possibly unnecessary) that every agent (human or automated) is able to assess the trustworthiness of every other agent on the Semantic Web. A number of techniques for propagating or transferring trust exist, but their applicability to the Semantic Web has only been tested and examined to a limited extent.

1.2 Intended work

In this research project, the author aims to investigate a number of sub-problems which exist in these problem domains. The concepts of the Semantic Web and trust as it relates to computing are not familiar to many computer scientists, and so this work begins by introducing the basics in both of these areas. These introductory chapters will build up knowledge of the required tools and concepts to move towards a solution for solving the problem of trust on the Semantic Web. In order to do this, the author defines the following goals:

1. Investigate and understand the Semantic Web, and how it relates to other current trends on the Web.
2. Investigate and understand the notion of trust, how it applies to the Semantic Web, and how it may be propagated.
3. Investigate suitable technologies for establishing authorship, and how these relate to the Semantic Web, for the purpose of establishing trust relationships

These goals will be met through literature review, construction of systems, and experimentation with the tools currently available at the time the research was performed.

The literature of this area is dynamic, and easily available electronically. In order to stay on the cutting edge, the use of electronic resources is important. Further, many academic publications are also available in an electronic format, and so the author has endeavoured to provide URLs for as many resources (of all types) as possible.

1.3 Outline

This thesis begins by introducing the two core ideas of this research: The Semantic Web and trust. These introductions are intended for those who are familiar with the workings of the Web, but not familiar with the idea of the Semantic Web or RDF. These introductory chapters also review previous work related to the problems which the author aims to investigate in this work.

The task of understanding trust and its relation to the Semantic Web can be divided into a number of sub-tasks. Namely,

1. Understanding of the Semantic Web, and the technologies used to realise it

2. Understanding the notion of trust
3. Relating existing trust concepts and existing methods for assessing Web credibility to the Semantic Web

Each of these tasks were tackled by creating sub-projects which contributed to the author's understanding of Semantic Web technologies, trust issues and the technologies involved. The projects tackled were:

1. A case study showing manual means of checking a website's authorship and affiliations in order to establish its credibility for more technically inclined users will be presented in Chapter 4. This is important as a first step towards understanding automated processes which may be used to check website credibility;
2. The development of a social networking website using RDF technologies, intended to gather trust data and gain a better understanding of development using RDF as the basic data store for such a project, which will be presented in Chapter 5;
3. The implementation of a Semantic Web scutter and smusher to data gathered from the Semantic Web will be presented in Section 6.1;
4. The analysis of the application of XML Digital Signature to RDF/XML, as a means of proving authorship of data, including a discussion on the canonicalisation of RDF/XML, which will be presented in Chapter 7

The design, implementation, and findings from these systems and experiments are detailed in the respective chapters.

This thesis concludes with a summary of the work done and findings, and some possibilities for the extension of this work are suggested.

A glossary of terms is presented as Appendix A, and the remaining appendices are longer pieces of code referred to from within the body of the work.

Chapter 2

An Introduction to The Semantic Web

> ***Goose:*** *How about terrific, terrific, terrific?*
> ***Charlotte:*** *Cut that down to one terrific and it will do nicely. I think terrific might impress Zuckerman.*
> ***Wilbur:*** *But Charlotte, I'm not terrific.*
> ***Charlotte:*** *You're terrific as far as I am concerned.*
> ***Charlotte:*** *Does anybody know how to spell it?*
> ***Goose:*** *I think it's T double-E double-R double-R double-I double-F double-I double-C, C, C.*
> ***Charlotte:*** *What kind of an acrobat do you think I am? It would take me all night to write a word like that into my web.*
>
> - Charlotte's Web [155]

The Semantic Web is a vision of Tim Berners-Lee, the creator of the World Wide Web, which seeks to create a "web of knowledge" [27, 28] – a rich information space in which concepts can be better described in terms which machines can more easily understand. These rich descriptions can then allow computers to more precisely process information gathered from the Web, better serving the human users. This contrasts with the current state of the Web – an information space largely designed and used for human consumption. Currently, documents on the Web are marked up with languages such as HTML [144] and XHTML [137], which convey some information about the structure of the document, but are largely used as formatting languages, along with supporting technologies such as CSS [30]. CSS is a technology which allows better separation of content and formatting information in HTML and XML. These sorts of technologies allow

website designers to tailor the look and feel of websites, conveying information to humans which is not readily usable by machines for information processing beyond display. Support for CSS has recently lead to support for techniques described as "semantic markup" [121, 120, 168], an initial step towards the widespread understanding of the importance of semantically rich content being published on the Web.

The problem of creating such a web of knowledge can be tackled from two angles: either intelligent computer systems which can understand the human languages used in web pages can be created, or information can be placed on the Web in a format which expresses the information in a rich, machine understandable format. It can be argued that the first approach has been taken (and used very successfully) by systems such as Google [38], which analyses the structure of documents found on the Web, and the links between these documents, and infers information about the content, semantics and relevance of web pages based on this link structure [134]. However, the amount of computing power harnessed by Google [13] points to the high barrier to entry which exists in this approach. The second approach allows the design of a system which is better suited to solving the problem of creating machine understandable content on the Web, and it is with this in mind that the idea of the Semantic Web was conceived.

Tim Berners-Lee, the inventor of the World Wide Web, originally described a system with richer semantics than the Web which currently exists [27]. Discussed in Section 2.3, Berners-Lee described a system which in many ways more closely resembles the emerging Semantic Web than it does the Web of human readable, graphically rich pages which currently exist. Haustein has claimed in [87, 88] that publishing information on the Semantic Web is too difficult for most who wish to publish information. The difficulty inherent in publishing semantically rich information may be one of the reasons why the Web evolved as it did, rather than with the richer semantics originally envisioned by Berners-Lee. In many ways the Semantic Web is taking a step back from the current trends and technologies, and looking at what needs must still be met by the Web. The Semantic Web Activity Statement states that "the Web can reach its full potential only if it becomes a place where data can be shared and processed by automated tools as well as by people" [91]. The aim of the Semantic Web is to make this machine processable data a reality.

This chapter is broken into 7 sections, beginning with a motivation of the Semantic Web, then contrasting the ideas with the recently popular ideas around Web 2.0. Next, the Resource Description Framework (RDF) is introduced as the technology which realises the Semantic Web, followed by a discussion on vocabularies. The Simple Protocol and RDF Query Language (SPARQL) and Gleaning Resource Descriptions from Dialects of Languages (GRDDL) are then introduced as technologies for querying RDF data and extracting RDF from other languages,

respectively. The chapter concludes with a summary of the concepts presented throughout the chapter.

2.1 Motivating the Semantic Web

The Semantic Web is an idea which is easy to motivate, but because of its conceptual nature, is difficult to explain. Tim Berners-Lee, the creator of the World Wide Web and the Semantic Web tells of how getting the idea of the Semantic Web across is reminiscent of the early days of the Web [27]. Three benefits which can be gained by creating a more machine-understandable web of knowledge are briefly discussed below:

1. Re-use and aggregation
2. Specific query and pattern analysis
3. Inference

2.1.1 Re-use and aggregation

Many Web users are already familiar and comfortable with the idea of RSS[1] [138] as a technology for syndicating content. vCard [52] and iCal [53] have also become popular machine readable formats for sharing contact and calendar information respectively. Publicly accessible calendars can also provide re-usable (syndicated) content, which can be aggregated for personal or group use. This syndication allows providers, such as bloglines [163], technorati [152] and others to re-use information from an RSS feed on their web pages. It also allows them to aggregate different news feeds and web logs, more commonly known as blogs, creating new and interesting information from the aggregated information. Geoblog [116] is another interesting example of the use of aggregated content which displays recently updated blogs on a world map, flashing titles of new posts as they are made.

The idea that information is published in a semantically rich format has been key to the recent explosion of blogs and aggregated news available in the RSS formats, thereby demonstrating the power of providing information in semantically rich formats.

[1] A number of similar formats, all called RSS, distinguished from each other by version numbers (they are not strictly versions) exist, including RSS 0.9x, RSS 1.0 and RSS 2.0 [138]. Atom is another similar format which has gained widespread use [85].

An already widely used RDF vocabulary is Friend of a Friend, or FOAF (see Section 2.4.1), which allows users to specify personal details, including basic relationship information. Generally, users publish information only about themselves and their relationships in their "FOAF file". By aggregating information from around the Web, a search engine or browser could allow a user to see more information than was recorded within any one particular file.

Both re-use and aggregation are important in making central portals of information more useful, and by using semantically rich, standardised formats for data, they become more easily deployed.

2.1.2 Specific query and pattern analysis

Currently, search engines operate on a free text analysis of documents intended for human consumption. This introduces the problem of ambiguity. For example, the search term "Rhodes" may refer to Cecil John Rhodes, Rhodes College (in Memphis, Tennessee), the Rhodes Scholarship, the island in Greece, or, as the author might be looking for, Rhodes University in Grahamstown, South Africa. Figure 2.1 shows some of these, and other search results returned by Google. Also evident in Figure 2.1 is that the Google search engine attempts to provide a rudimentary specific search system by grouping results, and allowing the user to perform the search again with added search terms to restrict the results.

By offering a refinement based on the *type* of information the user is looking for, a search engine may be able to offer better search results. Alternatively, a refinement in the form of a filter may be offered, for example filtering for web pages which relate to Rhodes University in Grahamstown, rather than Rhodes College in Memphis.

For technically inclined users, a query language which allows such specific search of RDF data stores is already available in the form of SPARQL, which is described in Section 2.5.

By creating a more structured description of concepts and how they interrelate, the Semantic Web enables developers to analyse the structure of the relationships between concepts. Often in understanding how concepts interrelate, we understand more about the system as a whole than we could by simply looking at each concept in isolation. In a corporate environment this could lend itself to organisational structure analysis, and the day to day life of an information worker may be enriched by more powerful search tools which allow them to find closely related concepts. Whilst data-mining and text analysis techniques currently provide such results, the resulting descriptions are often "shallow" and need to be enhanced by humans [61]. In addition, few of them provide a way to share the information on the Web, and are limited to internal use

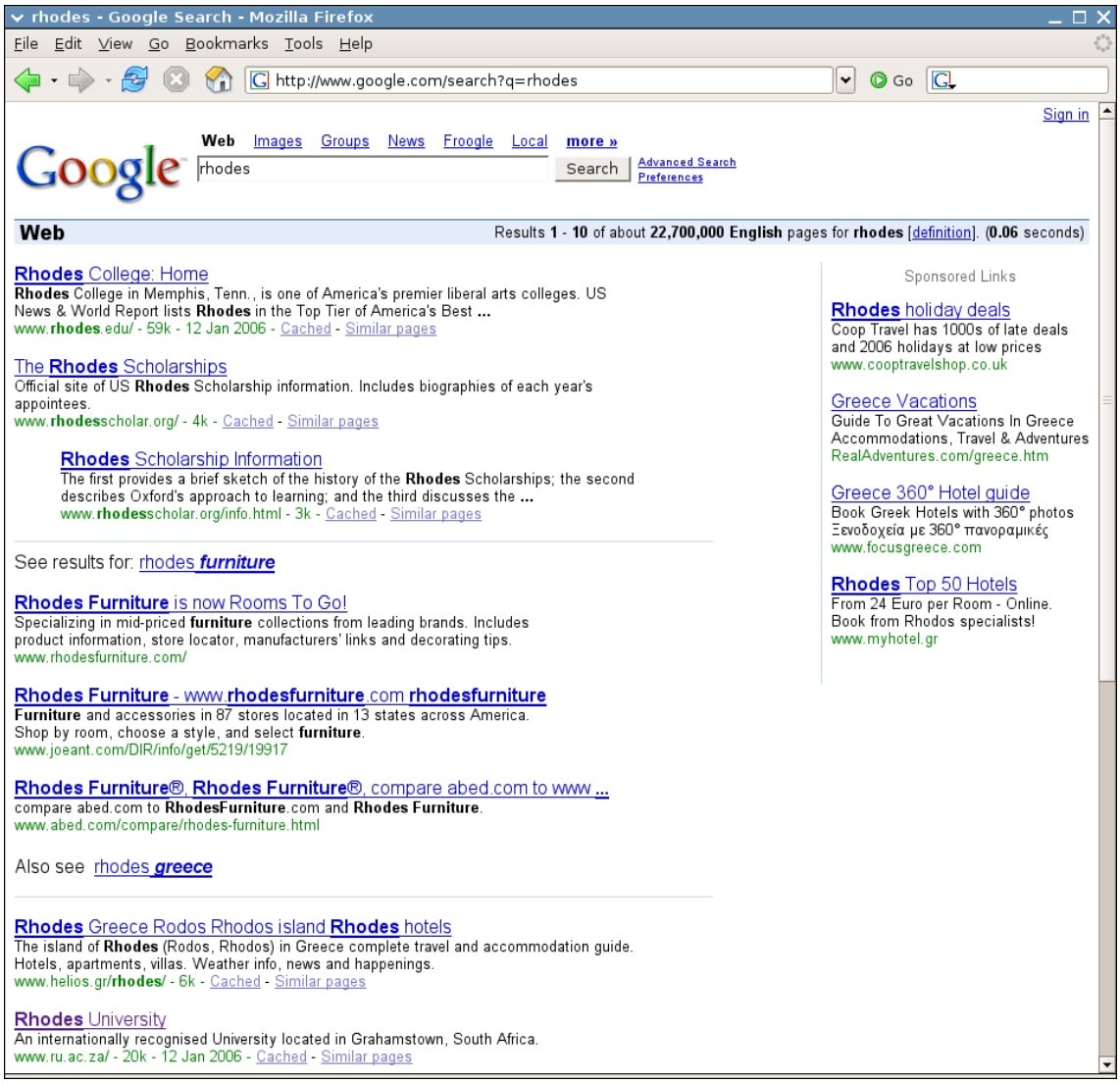

Figure 2.1: A search on Google for “Rhodes” returns a wide variety of results

within an organisation.

This sort of pattern analysis can also allow better search results, because a computer will be able to infer that two concepts, although labelled differently, and described by different resources (that is, located at different Uniform Resource Identifiers, or URIs) are the same thing because of the properties which they share. This is similar to the topic of inference, but differs in that human (or other outside) specification of the links between concepts is not necessary.

2.1.3 Inference

By using well written vocabularies (discussed in Section 2.4) to describe information and a language which supports inference to express these descriptions, it becomes possible for a computer to infer information which was not expressed directly by a human when the information was created.

Let us take as an example some person selling a sedan. The software (or website) which she is using may only present the user with options about the vehicle being sold, such as "sedan" or "coupe". When publishing the information on the Web, the information would link to a vocabulary which describes motor vehicles, and includes statements such as "all sedans are motor cars". A potential buyer, searching for all cars for sale, would be able to infer that the sedan for sale is also a motor car, and that item would therefore appear in the search results, even though the publishing website did not publish a statement saying that the item for sale was a motor car.

2.2 Web 2.0

"Web 2.0" is a term coined by Dale Dougherty of O'Reilly Media, with the original definition offered by way of examples, rather than as a concrete definition [157]. Some of the examples which have been used to help explain Web 2.0 include [132]:

- Services: "Web 1.0" DoubleClick contrasts with the Web 2.0 Google AdSense. DoubleClick required contracts with each new advertiser, whereas AdSense harnesses the "long tail" by providing advertising to advertisers of a diverse budget range.
- Information resources: Britannica Online contrasts the collaborative Wikipedia. Wikipedia promotes participation rather than centralised publishing by allowing anybody to contribute.

- Modus operandi: screen scraping is no longer necessary with pervasive web services and content provided in formats such as RSS, which promote syndication and re-use.
- Thought patterns: participation replaces publishing, as exemplified by Wikipedia.

A "meme map", developed at the O'Reilly Media conference FOO Camp [133], which illustrates the core concepts of the Web 2.0 idea is shown in Figure 2.2, and provides some examples of the services and ideas which embrace the ideals of Web 2.0. At the top of the meme map are those services which take advantage of Web 2.0 ideals, and at the bottom the ideas to which they subscribe. At the center of the meme map are the ideals core to Web 2.0.

Tim O'Reilly provides a number of insights into what Web 2.0 is really about in "What is Web 2.0" [132]. The most important of the insights provided – the long tail, the network effect, data being a key asset and software being delivered as a service – are discussed briefly below.

The long tail

O'Reilly suggests that businesses looking to fully exploit the Web 2.0 should be cognisant of the power of the "long tail". Companies in the "Web 1.0" list, such as DoubleClick [56], had a business model which focused on traditional contracts, signed with big advertisers and big content sites. The Web 2.0 model rather allows customers to use self-service, and has ensured that services such as Google's AdSense [81] are visible on a hundreds of thousands of websites, which, it is suggested, contain a large portion of the Web's content. This contrasts with the DoubleClick, which proudly claims "over 2000 successful installations". O'Reilly suggests that Web service companies "leverage customer-self service and algorithmic data management to reach out to the entire Web, to the edges and not just the center, to the long tail and not just the head." [132]

The long tail refers to those smaller volume customers which cannot be reached using traditional business models. Although each of these customers is small when standing alone, there is a large number of them and they can therefore provide a large volume of business. Companies such as Amazon.com [5], which have only an online presence, can provide a wider variety of products to customers, as stock display space is not limited by a physical store. Online services, such as Google's AdSense, can reach a large number of customers due to its self-service nature.

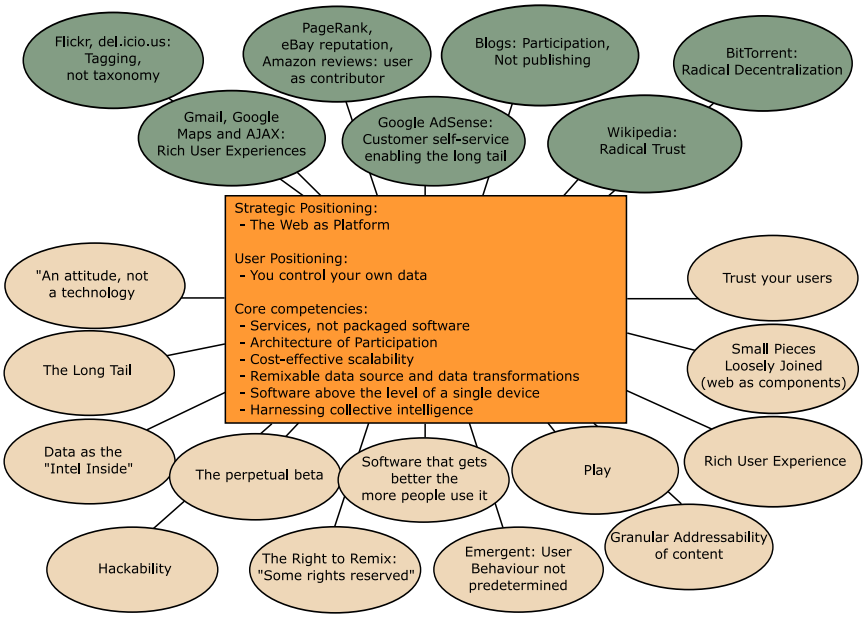

Figure 2.2: Web 2.0 “meme map” developed at FOO camp. This image is reproduced from [132]

The network effect

One of the earliest examples of the use of users' data to improve their experience is with Amazon.com [5]. Amazon.com has successfully used user contributed reviews and ratings to power a website which helps users to make informed purchasing decisions easily. Amazon also feeds information gathered about the user, such as her purchase and browsing history, back into the system, further strengthening and bringing personalisation into recommendations. As O'Reilly puts it, "network effects from user contributions are the key to market dominance in the Web 2.0 era." [132]

The network effect encompasses two further properties of the Web 2.0 idea, namely that sites which subscribe to the Web 2.0 ideal harness the wisdom of the crowds and that they promote participatory rather than centralised publishing. This has been seen with the popularity of blogging, a technology used both as a personal journal or diary and also as a source of news. The participatory, Web based nature of blogging means that there are no gate keepers to regulate the content available on the Web, and that readers can choose which content to consume out of a wide variety of available content. Further, collaborative content filtering allows people to filter out content which their peers rate poorly, and only view content which is highly rated amongst their peers. This so called "wisdom of the crowds" helps to separate good content from bad content, but also allows readers to only view that content which is suitable for them.

Data is a key asset

Data is a key asset of Web 2.0 sites. A powerful example discussed by O'Reilly is that of the recent emergence and popularity of online map services. Perhaps the most popular, and most Web 2.0 friendly, is Google Maps [83]. Other big players have entered the market too, with products like Yahoo! Maps [164] and MSN Maps and Directions [122]. Whilst some sites, such as MapQuest [115] have been in the market for a long time, Yahoo!, Google and MSN were able to competitively enter the market simply by licensing the data.

Similarly, Amazon licensed a database of book information from an ISBN registry provider, R. R. Bowker [132], which put them on a similar footing to competitors like Barnes and Noble [12]. However, Amazon has enriched that data, by providing useful information such as cover images and tables of contents. Additionally, by coupling the network effect with their enriched data source, Amazon has gained a distinct advantage over competitors such as Barnes and Noble [132].

O'Reilly puts forth that "data is the next Intel Inside", implying that the valuable data behind these sites is a key point where the components around that data such as the software, is either largely commoditised or freely available as open source software.

Software as a service

In the Web 2.0 ideal, software is not distributed as a product, but rather becomes a service that is delivered across the Web. This drastically changes the way the software is deployed and used, and also has implications on the important competencies required within companies.

One of the important new competencies required is operations. A company delivering a service over the Web has much higher systems infrastructure demands than a company which distributes its software through traditional channels. In addition to server infrastructure, special maintenance operations are important too. For example, search engines need to maintain crawlers, indexes and large numbers of servers, and blogging services must ensure that spam detection and removal tools are accurate and applicable to the latest threats.

Another phenomenon which the "software as a service" paradigm introduces is the "perpetual beta". When software is delivered as a service, it is possible for it to be continuously and transparently updated. As an example, it has been reported that Flickr update their application, and thus their product, as frequently as once every half an hour [47]. Again, an operations competency in providing rapid deployment is important in enabling this.

It has been seen that services which provide lightweight, accessible APIs that open their data to external use have enjoyed widespread use. Amazon.com report that 95% of their web services usage is through the lightweight REST API, rather than the heavy SOAP API [132]. Lightweight APIs lend themselves to easy use by casual programmers who wish to create a quick, fun hack, leading to interesting results, such as HousingMaps [143], which combines the mapping abilities of Google Maps, and the mass of information on housing for sale and rent available on craigslist [49].

O'Reilly says that "a key aspect of the Web 2.0 idea is the ability to extract and query information held across many different ad hoc, third-party apps, services, or repositories". Many sites which are cited as examples of those carrying the Web 2.0 flag have been used in these "remixed" or "mashup" services. These mashup services provide "innovation in assembly", gaining value from the data available from other sites.

2.2.1 Web 2.0 is not the Semantic Web

The Web 2.0 is an idea which promotes a way of thinking about Web technologies, and even the concept of the Web in general, and not a group of technologies or even an encapsulated, well defined concept. The Semantic Web, on the other hand, is a concept which describes a new way of sharing information – a web of facts and knowledge, which is machine processable to the extent that a computer may make inferences from the knowledge gathered from the Web. The Semantic Web is also realised by a specific set of technologies, including RDF [102, 89], RDF/XML [18] and OWL [16], which realise the goal of a web of knowledge, whereas any technology or service (new or existing) could be described as a part of Web 2.0 if it promoted or harnessed a manner of thinking consistent with the Web 2.0 ideal.

Like the Semantic Web, Web 2.0 can be seen as a logical evolution of the World Wide Web. Web 2.0 seems like a more natural evolution; a progression using some technologies which have been around for a long time. Proponents of Web 2.0 seek to exploit these technologies to their full potential by using them correctly and effectively, or in novel new ways. The Semantic Web, however, has its basis in an academic goal, and seems a more forced progression. Semantic Web technologies have been designed to meet a rather lofty goal, as opposed to the practicality of Web 2.0 technologies, which are adaptations of, or new use cases for older technologies. The Semantic Web could be described as an architected evolution of the Web, whereas the Web 2.0 is an organic evolution.

2.2.2 Web 2.0 in summary

Web 2.0 is a term which describes a current trend, rather than a specific technology. Whilst the term has no solid definition, it describes the zeitgeist in Web development and Web commerce. The definition is soft, and an understanding of the term can best be gained through understanding the examples which epitomise the Web 2.0 trend, and contrast against “Web 1.0” examples. In general, these examples highlight properties of the Web 2.0 idea. DoubleClick can be contrasted against Google Ads, where Google Ads harnesses the power of the long tail. Structured Content Management Systems can be compared to Wikis, and columns published by journalists compared to blogs and the blogging phenomenon, highlighting the collaborative nature of the Web 2.0 idea, and the power of popular vote rather than centralised media control. The examples and ideals are limitless, and Tim O’Reilly has attempted to summarise the essence of the idea into a compact definition [131]:

> Web 2.0 is the network as platform, spanning all connected devices; Web 2.0 applications are those that make the most of the intrinsic advantages of that platform: delivering software as a continually-updated service that gets better the more people use it, consuming and remixing data from multiple sources, including individual users, while providing their own data and services in a form that allows remixing by others, creating network effects through an ”architecture of participation,” and going beyond the page metaphor of Web 1.0 to deliver rich user experiences.

The Web 2.0 idea differs from the Semantic Web idea in that Web 2.0 promotes the lightweight, accessible exchange of data through any means, whereas the Semantic Web prefers a structured approach to data sharing, which will allow specific queries and reasoning across data of a known structure.

2.3 The Resource Description Framework

We have described the vision of the Semantic Web: a web of knowledge in which concepts can be interlinked and described in a rich, machine understandable format. The Resource Description Framework (RDF) provides the technology which realises this vision.

RDF is a framework designed for describing resources present on the World Wide Web [21, 27]. Although originally intended as a metadata format, RDF data can be thought of as data in its own right, rather than metadata for other data on the Web. For example, FOAF [37] is an RDF vocabulary which describes people and the links between them. It is more natural to think of FOAF as data rather than metadata.

RDF aims to be as lightweight as possible, and to impose as little on the underlying application as possible. Extensions to RDF, such as RDF Schema [36] and the Web Ontology Language (OWL) [16] provide semantics on top of the basic framework, which support advanced operations such as reasoning across data sets.

At the core of RDF is the notion of the Uniform Resource Identifier (URI) [26], which was one of the core enablers of the Web. In an abstract sense, a URI is a decentralised, human readable labelling scheme for resources which are a part of the Web.

In the Web as we currently know it these labels are used as handles for resources which can be retrieved over the Internet. This is the key difference between a URI, which is simply an abstract label for a resource, and a Uniform Resource Locator (URL), which identifies an address for the

resource. Historically, the difference was considered more important, but the terms have become interchangeable (even in technical documentation) as URLs have become by far more common than URIs which do not address a resource (which are known as Uniform Resource Names, or URNs). So, in a more concrete sense, a URI can be thought of as an address from which a document may be retrieved.

By "dereferencing" the URI, an agent is able to find out what protocol should be used to access the resource (HTTP [62] is commonly used on the Web), which computer system the resource exists on, and where on that system it resides. URIs are used in markup languages such as HTML to provide links between documents, which may be used by the user to navigate through the Web until the information she desires is found. There is, however, no technology which allows the description of these resources (the value of which is discussed in Section 2.1). RDF is designed to fill this gap.

The data model of RDF is very simple, and consists simply of a collection of triples. Each triple consists of three parts: the subject, predicate and object of the statement which the triple represents. For both humans and machines, these simple statements are easy to understand and work with. Resources on the Web can be described with simple properties, such as "creator", which can be given values, such as "Tim Berners-Lee". These properties are always resources (referred to by a URI), and the values may either be literal values, or another resource.

The subject and object of the triple can also be thought of as nodes in a graph, and the predicate as a labelled arc between them, as shown in Figure 2.3. Figure 2.3 also shows that the arcs in the graph are directed, as well as labelled. The arc is directed from the subject to the object of the statement, and the URI of the predicate is the label. Thus, RDF can be represented as a graph, as well as a set of triples.

The original proposal for the World Wide Web [21] proposes a system with labelled arcs, which connect objects which not only include documents, but also people, concepts and organisations, as shown in Figure 2.4. Unfortunately, the World Wide Web did not materialise in that form, with only a weak and largely unused type system now existing to describe a link (given by the relation, `rel`, attribute to the anchor, `a`, tag). However, the Semantic Web aims to fulfill that vision.

Figure 2.4 emphasises the graph based nature of the World Wide Web, but also demonstrates the graph based nature of the web of knowledge which the Semantic Web aims to become. Although it doesn't map strictly and purely to RDF, RDF goes a long way to achieving this with its graph model which includes directed, labelled arcs. An effort to describe this original proposal in RDF


http://www.w3.org/DesignIssues/Naming

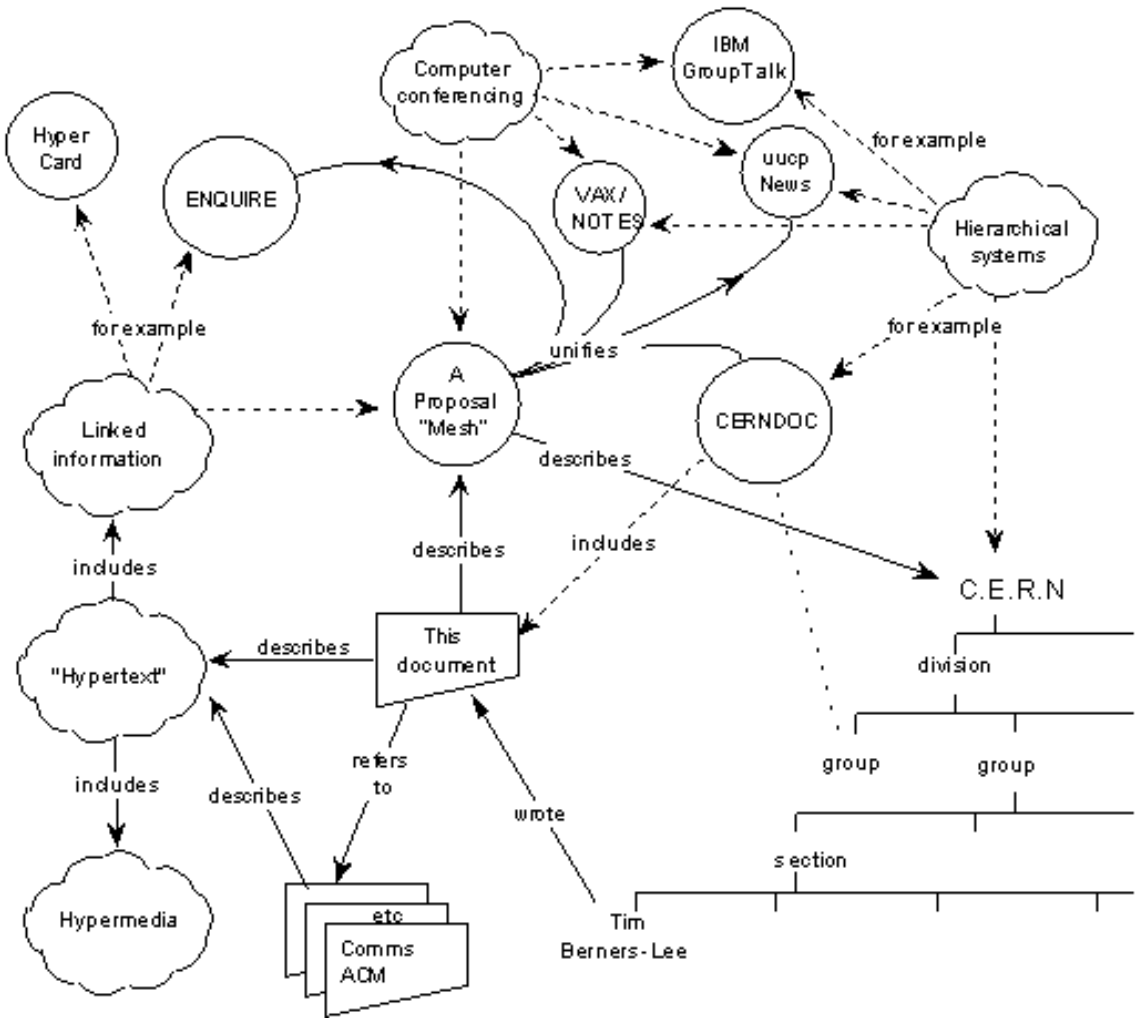

Figure 2.4: A figure from the original proposal for the Web by Tim Berners-Lee suggests a graph structure similar to that created with the Semantic Web [21, 34].

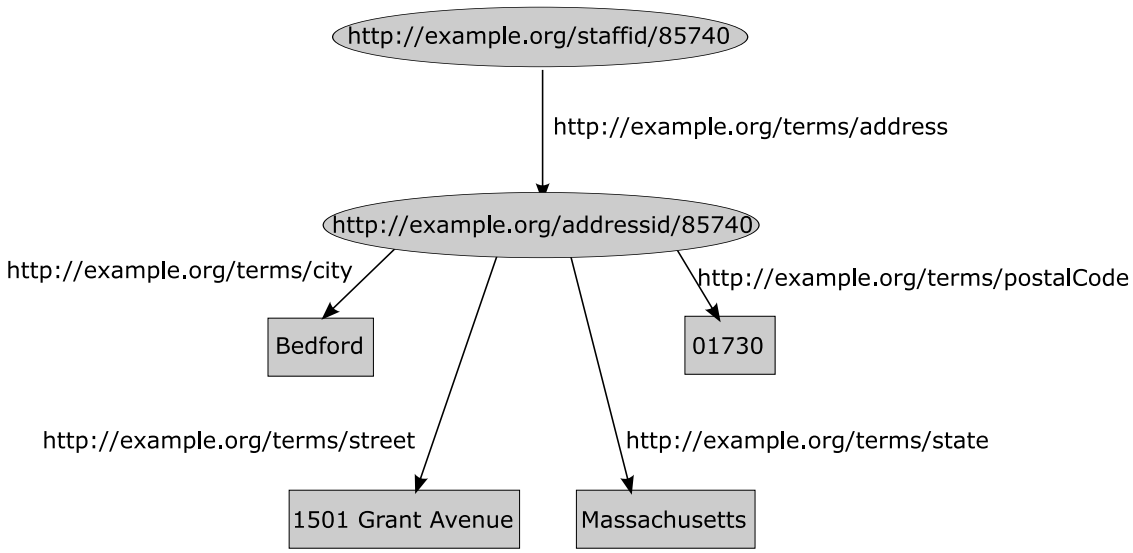

Figure 2.5: Using a resource as a value to introduce structure, reproduced from [114]

is described in [34], including a link to an annotated RDF document which achieves much of this goal.

In some more complex cases it is necessary to give a property a value which is better expressed as discrete, structured pieces of information, rather than as a single literal value. This may be done by making the value of the property another resource, which, in turn, can have properties and values. This is illustrated in Figure 2.5, where a mailing address is expressed as a structured set of properties, including the city, street, state and postal code.

Resources should have labels (URIs) which can be used to identify the data. For security and privacy reasons this is not always appropriate. For example, some people may be uneasy with the notion that they have a global identifier (in the form of a URI) associated with them. This can be remedied by the introduction of a special type of resource, called a blank node or anonymous resource which does not have an identifier, as shown in Figure 2.6. Because blank nodes do not have identifiers, they can not be referenced from outside the graph in which they exist. They can, however, be referenced from within the serialisation of triple set of the graph in which they exist using a local identifier.

Expanding on the author example above, it is possible to model reality more closely, by describing the the author of the document as a resource of type person, which in turn has properties, such as the person's name, title, and contact details. By using blank nodes, we are not forced to give

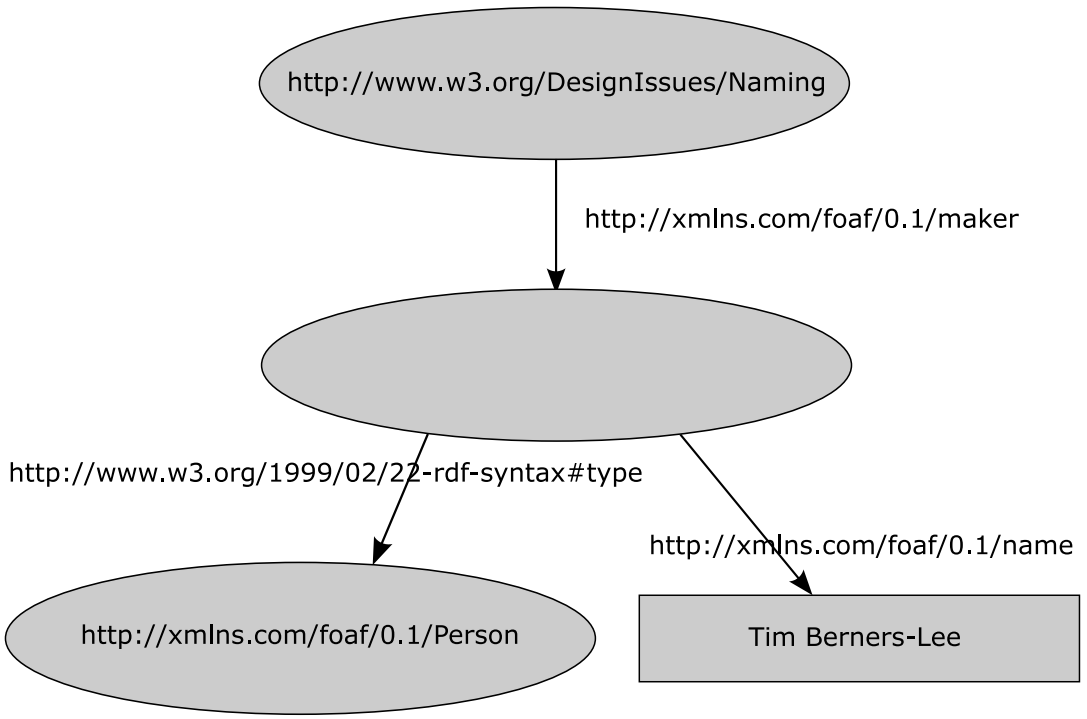

Figure 2.6: A blank node in an RDF graph does not have a URI

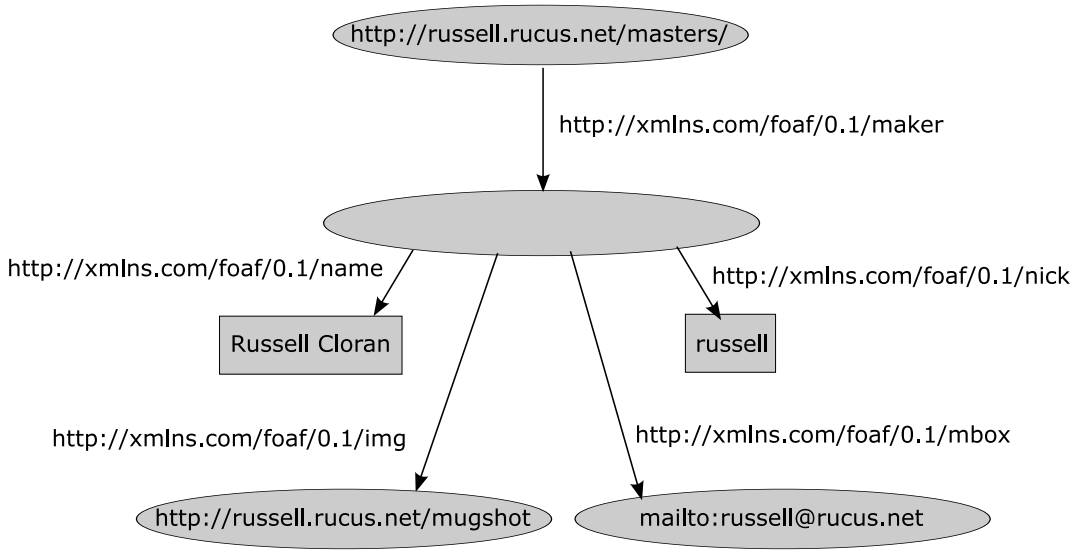

Figure 2.7: Using a blank node avoids having to assign a URI to a person

the person a universally identifiable label (URI). In Figure 2.7, we see a graph which describes the author of `http://russell.rucus.net/masters/` as a resource, with properties such as a name, nickname, email address and photograph.

The use of URIs to label resources is convenient, and familiar to many users of the Web. Writing long URIs each time a resource is referenced is, however, not always convenient. The concept of namespaces and namespace prefixes is thus borrowed from XML and used when writing RDF in various formats (RDF serialisation formats are described in Section 2.3.4 and Section 2.3.5).

2.3.1 Namespace prefixes

XML allows the notion of "namespaces", which may have an associated prefix or Qualified Name (QName) [32]. A namespace can simply be thought of as a URL fragment. Elements within that namespace have a URL which is constructed by concatenating the namespace URL fragment and the element name. Figure 2.8 shows an XML fragment which uses namespaces. The default namespace in this example is `http://xmlns.com/foaf/0.1/` (the FOAF namespace), the `rdf` and `rdfs` namespaces are defined as `http://www.w3.org/1999/02/22-rdf-syntax-ns#` and `http://www.w3.org/2000/01/rdf-schema#` respectively. These are the correct namespaces for RDF and RDF Schema (Section 2.3.2).

```
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns="http://xmlns.com/foaf/0.1/">
</rdf:RDF>
```

Figure 2.8: An XML fragment using namespaces

QName	URI
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs	http://www.w3.org/2000/01/rdf-schema#
foaf	http://xmlns.com/foaf/0.1/
dc	http://purl.org/dc/elements/1.1/
vcard	http://www.w3.org/2001/vcard-rdf/3.0#

Table 2.1: Commonly used QNames as used in this work

As described above, the URL for an element is constructed by concatenating the URL for the namespace with the element name. The URI for the XML element shown in Figure 2.8 would therefore be `http://www.w3.org/1999/02/22-rdf-syntax-ns#RDF`, constructed by concatening the URI for the `rdf` namespace with the element name, `RDF`.

Throughout this work a number of QNames are used, with the assumption that their meaning is known. These are listed in table 2.1. The QNames used in this work are also the widely used QNames for the respective URIs. It should be noted that it is not necessary to use these QNames for the URIs in any documents.

2.3.2 RDF Schema

RDF defines a framework for metadata statements, but does not define any terms or semantics which may be used to describe resources. In order to create a rich machine-understandable web of knowledge, it is necessary to be able to describe the terms which are used in making statements in a manner which richly describes their semantics in a machine understandable form.

RDF Schema provides a type system for objects, a system for describing properties, collections of resources (see Section 2.3.3), and a number of other terms allowing features such as annotation and data-typing on top of RDF [36].

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
<foaf:Person>                  <rdf:type>         <rdfs:Class>
<foaf:Person>                  <rdfs:subClassOf>  <foaf:Agent>
http://russell.rucus.net/me/#  <rdf:type>         <foaf:Person>
```

Figure 2.9: An example of using RDF Classes, using the N3 notation described in Section 2.3.4.

The RDF Schema definition itself, and documents which use RDF Schema to describe resources (see Section 2.4) are valid RDF, and may be processed by RDF tools. In order to take advantage of the meaning provided by RDF Schema, tools need to understand the meaning of the extra terms introduced by RDF Schema, which exist in both the `rdf` and `rdfs` namespaces.

One thing which may be noted (in previous and following discussion) is the convention of using an uppercase letter for the first letter of the local part of a classes URI, and a lower case letter for the first letter of properties and class instances. This is not a requirement, but is encouraged for uniformity.

Types and classes

In RDF Schema, the notion of a "Class" is similar to that in object oriented programming, and corresponds to the generic notion of a "type" or "category" of objects. A class is a resource (it is only useful if it has a URI), and has the type (`rdf:type`) `rdfs:Class`. The URI of a class can then be used as the type for *instances* of that class. Figure 2.9 illustrates this using the FOAF vocabulary, and includes some statements which exist in the FOAF vocabulary specification.

Figure 2.9 also includes an example of the use of sub-classing, a notion which also exists in object oriented programming, and is often called specialisation of objects. In this case, a `foaf:Person` is described as a subclass of a `foaf:Agent`, implying that all instances of `foaf:Person` are also instances of `foaf:Agent`.

Properties

RDF Properties are the terms which are used as the predicates of statements which help to describe resources. For example `foaf:birthday` is a property of the `foaf:Person` class.

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
<foaf:Agent>   <rdf:type>     <rdfs:Class>
<foaf:gender>  <rdf:type>     <rdf:Property>
<foaf:gender>  <rdfs:domain>  <foaf:Agent>
<foaf:gender>  <rdfs:range>   <rdfs:Literal>
```

Figure 2.10: An example of describing an RDF property

Properties are resources which are instances of the class `rdf:Property`. Like a class, a property is only useful if it has a URI, which is used as the predicate of statements.

The most important part in the specification of a property is the *domain* and *range* of the property. The context of these terms is the same as when describing mathematical functions, and they refer to the types of class to which the property applies, and what type of value the property may have, respectively. Put another way, the `rdfs:domain` of a property is the class (or set of classes) which may be the object of a statement where the property is the predicate, and the `rdfs:range` is the class (or set of classes) which may be the subject of such a statement. Figure 2.10 illustrates the description of the `foaf:gender` property. In the figure `foaf:gender` is described as a property which has a domain of `foaf:Agent`. In other words, any resource of type `foaf:Agent` may have a `foaf:gender` property. `foaf:gender` has a range of `rdfs:Literal`, which means that a `foaf:gender` property may only have literal content as its value.

One of the types of data structures provided by RDF, rather than RDF Schema is the description of various types of groups, provided by containers and collections.

2.3.3 RDF containers and collections

RDF containers allow the description of groups, and are specified in RDF Schema. Three types of containers exist: the `rdf:Bag`, `rdf:Seq` and `rdf:Alt`. A container is expressed by using a resource of type `rdf:Bag`, `rdf:Seq` or `rdf:Alt` as the object of a statement. This resource can be thought of as the collection, and each property of that resource is a member of the container. This is more clearly illustrated in Figure 2.11, which uses an rdf:Bag container to describe the members of an interest group.

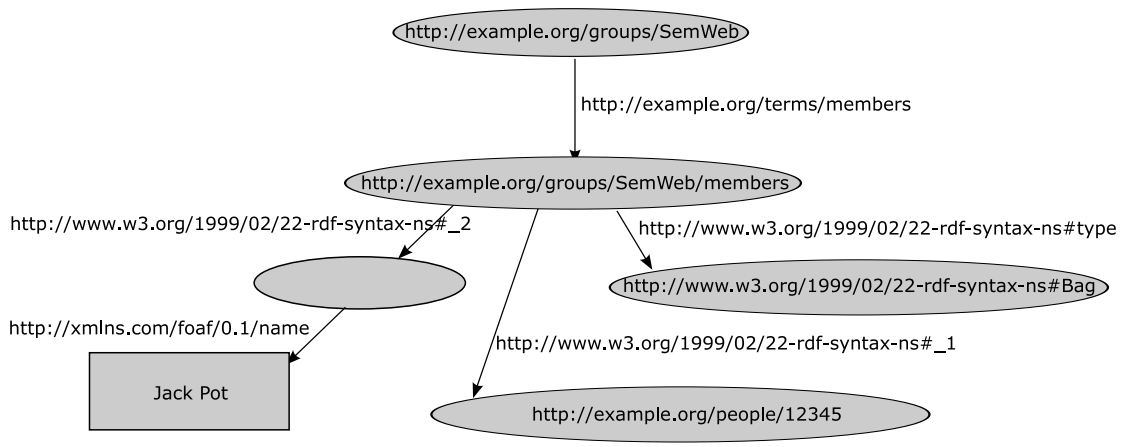

Figure 2.11: Using an `rdf:Bag` container to describe members of an interest group. The node with URI `http://example.org/groups/SemWeb/members` is the bag (has type `rdf:Bag`), with two members: a blank node and the node with URI `http://example.org/people/12345`

- A `Bag` describes an unordered group, for example the members of an interest group, as shown in Figure 2.11. This is similar to a mathematical set.
- A `Seq` describes a sequence, or ordered group, of resources or literals. A group of finishers of a sporting event may be maintained in an `rdf:Seq` container, helping to describe finishing position.
- An `Alt` describes a group for which each member is an alternative, for example media with a number of possible titles could use an `rdf:Alt`.

RDF collections are similar to containers, but allow the group to be closed. With an RDF container, only resources or literals which are contained within the group are expressed, but there is no way of expressing exclusion from the group.

2.3.4 Simple RDF Serialisations

RDF is only a data model, that is, a *conceptual* framework for the representation of knowledge using simple building blocks. In order to exchange this knowledge, some sort of encoding, known as a serialisation, is necessary. A few "simple" RDF serialisations exist. These serialisations

```
<http://www.w3.org/2001/08/rdf-test/> <http://purl.org/dc/elements/1.1/creator>   "Dave Beckett" .
<http://www.w3.org/2001/08/rdf-test/> <http://purl.org/dc/elements/1.1/creator>   "Jan Grant" .
<http://www.w3.org/2001/08/rdf-test/> <http://purl.org/dc/elements/1.1/publisher> _:a .
_:a                                   <http://purl.org/dc/elements/1.1/title>     "World Wide Web Consortium" .
_:a                                   <http://purl.org/dc/elements/1.1/source>    <http://www.w3.org/> .
```

Figure 2.12: An example of a simple N3 file [84]

are commonly used for expressing RDF graphs which will be viewed by humans in a textual manner, or written by humans. Because of this, they are often used with tools such as Closed World Machine (CWM) [23] for research purposes. As compared to RDF/XML (Section 2.3.5), these serialisations generally lack the benefits of using XML as a serialisation language. There are three such serialisations in common use, N-Triples, N3 [24] and Turtle [19]. All three are very similar, because N-Triples and Turtles are subsets of N3.

N3

N3, or Notation-3 is the oldest serialisation language, and aims to be "a compact and readable alternative to RDF's XML syntax" [24]. N3 can be used to express graphs which are not compatible with RDF, although it does allow the expression of all RDF graphs. N3 is serialised as UTF-8 [166], an encoding of the Unicode character set [153]. Unicode aims to encode characters used in all languages around the world, and so this provides a basis for N3 which supports full internationalisation. A simple N3 file is shown in Figure 2.12. This file illustrates the simple basis of N3. Each triple is thought of as a line, terminated by a period. URIs are enclosed in angle brackets, and string literals in double quote marks. The file also illustrates the use of blank nodes by using an underscore, and providing an identifier (used only within the document) for the blank node after a colon.

There are a few other main and commonly used features in N3, namely namespaces, predicate-object and object lists, and an alternative blank node syntax.

Namespaces The first is the use of a namespace, or prefix, in order to reduce the verbosity of an N3 document. RDF vocabularies are defined with a certain base URI, and each term is appended to that URI in order to obtain the full URI of the term [114]. Thus, Figure 2.12 could be rewritten to use the prefix of `dc` (for "Dublin Core" [29], the vocabulary in use, discussed

```
@prefix dc:  <http://purl.org/dc/elements/1.1/> .
<http://www.w3.org/2001/08/rdf-test/> dc:creator   "Dave Beckett" .
<http://www.w3.org/2001/08/rdf-test/> dc:creator   "Jan Grant" .
<http://www.w3.org/2001/08/rdf-test/> dc:publisher _:a .
_:a                                   dc:title     "World Wide Web Consortium" .
_:a                                   dc:source    <http://www.w3.org/> .
```

Figure 2.13: An N3 file using a namespace prefix to improve readability and decrease verbosity, derived from Figure 2.12

```
@prefix dc:  <http://purl.org/dc/elements/1.1/> .
<http://www.w3.org/2001/08/rdf-test/> dc:creator   "Dave Beckett", "Jan Grant" .
<http://www.w3.org/2001/08/rdf-test/> dc:publisher _:a .
_:a   dc:title "World Wide Web Consortium"  ;  dc:source <http://www.w3.org/> .
```

Figure 2.14: Using commas and semicolons to further reduce the verbosity of an N3 file, derived from Figure 2.13

further in Section 2.4.1) to mean `http://purl.org/dc/elements/1.1/` when writing the property (predicate) URIs, as shown in Figure 2.13.

Predicate-object and object lists The second important feature in N3 is the shorthand use of a comma and semicolon. The comma indicates that the term to follow is an object of the previous subject and predicate, and a semicolon indicates a new predicate and object for the previous subject. Figure 2.14 shows how the comma and semicolon can be used to improve the readability of an N3 file by reducing its verbosity. The comma and semicolon have been used separately in this example for illustrative purposes. Figure 2.15 shows how commas and semicolons may be used together to further simplify the N3 file.

```
@prefix dc:  <http://purl.org/dc/elements/1.1/> .
<http://www.w3.org/2001/08/rdf-test/>  dc:publisher _:a ;
                dc:creator   "Dave Beckett", "Jan Grant" .
_:a   dc:title "World Wide Web Consortium"  ;  dc:source <http://www.w3.org/> .
```

Figure 2.15: Using commas and semicolons in combination to reduce the verbosity of an N3 file even further

```
@prefix dc:  <http://purl.org/dc/elements/1.1/> .
<http://www.w3.org/2001/08/rdf-test/> dc:creator   "Dave Beckett", "Jan Grant" .
<http://www.w3.org/2001/08/rdf-test/> dc:publisher
     [ dc:title "World Wide Web Consortium"  ;  dc:source <http://www.w3.org/> ] .
```

Figure 2.16: Using square brackets to describe blank nodes in an N3 file. The line wrap is neither necessary, nor incorrect. N3 ignores white space.

Alternative blank node syntax The final important feature of N3 is the use of square brackets to describe anonymous resources, or blank nodes. Figure 2.16 shows how square brackets can be used to eliminate the need for document-level labels for anonymous resources, by serialising them in-line with the statement of which the blank node is an object. This allows anonymous resources to be truly anonymous, but it also means that the node cannot be used as the object of a statement elsewhere in the graph.

N-Triples

N-Triples is a very limited subset of N3 [84], primarily designed for consumption by simple scripts. N-Triples documents are encoded in US-ASCII, but can escape UTF characters, and internationalisation is therefore supported. N-Triples does not allow any of the features of N3 which improve readability, such as the features discussed above. Additionally, each triple must be contained on a line.

Turtle

Turtle is also a subset of N3 [19]. Turtle is based on N-Triples, but also takes the most useful features from N3, but restricts the syntax so that only valid RDF graphs may be expressed. Notably, the encoding of Turtle is in UTF-8, unlike the ASCII encoding used in N-Triples, and it adds all of the features of N3 which are discussed above, amongst some other features, as listed in [19]. It thus follows that all of the examples used above, from Figure 2.12 to Figure 2.16 are all valid examples of Turtle files.

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:dc="http://purl.org/dc/elements/1.1/">
    <rdf:Description rdf:about="http://www.w3.org/DesignIssues/Naming">
       <dc:creator>Tim Berners-Lee</dc:creator>
    </rdf:Description>
</rdf:RDF>
```

Figure 2.17: A basic RDF graph of one statement encoded in RDF/XML with emphasis to show the striped nature of the RDF/XML serialisation.

2.3.5 RDF/XML

The XML serialisation of RDF is the official RDF syntax, as specified in [18]. While the RDF Concepts and Abstract Syntax [102] and RDF Semantics [89] recommendations use N-Triples (see Section 2.3.4) in the documents, this usage is mainly for communicating the ideas to readers.

A simple way of looking at most RDF/XML is by considering it as a striped syntax [35]. As has been previously discussed, RDF is conceptually a graph consisting of nodes, linked by directed arcs. Each node in the RDF graph is described using an XML node, and each arc in the RDF graph is also described using an XML node. RDF graphs can be considered as a collection of paths which traverse the graph. A collection of traversals such as this is encoded into RDF/XML as a sequence of XML elements, with the first node as the parent element, each arc from that node as a direct child of it, and the nodes which the arcs point to as children of that element. Such alternation between nodes and arcs leads to the striped syntax, as shown in Figure 2.17. A common mistake amongst those unfamiliar with RDF/XML is to use multiple child nodes within a node which describes a predicate. This syntax is not allowed, and would also break the striped syntax.

XML elements in the RDF/XML encoding are translated to the URIs of the RDF model by using XML namespaces. The name of each node is concatenated with the URI of the namespace to obtain the full URI of the RDF node or arc being described. Figure 2.17 uses the RDF graph shown in Figure 2.3 as an example of how an RDF graph is encoded into RDF/XML.

Blank nodes are serialised in RDF/XML simply by not specifying a URI in the `rdf:about` attribute. If the object of a statement is a resource about which no further description is required, it is possible to use an `rdf:resource` attribute within the predicate's XML node, leaving out children nodes for that node. Both of these concepts are illustrated in Figure 2.18.

RDF/XML:

```
<?xml version="1.0"?>
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:foaf="http://xmlns.com/foaf/0.1/">
   <foaf:Person>
      <foaf:name>Russell Cloran</foaf:name>
      <foaf:knows>
         <foaf:Person rdf:about="http://example.com/people/54321" />
      </foaf:knows>
      <foaf:knows rdf:resource="http://example.com/people/12345" />
   </foaf:Person>
</rdf:RDF>
```

N3:

```
@prefix  rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix  foaf: <http://xmlns.com/foaf/0.1/> .
_:1   <rdf:type>      <foaf:Person>  .
_:1   <foaf:name>     "Russell Cloran"  .
_:1   <foaf:knows>    <http://example.com/people/54321>  .
<http://example.com/people/54321> <rdf:type> <foaf:Person> .
_:1   <foaf:knows>    <http://example.com/people/12345>  .
```

Figure 2.18: An example RDF/XML serialisation with equivalent N3 showing blank nodes, typed nodes and the `rdf:resource` attribute. Emphasis shows that the RDF/XML "striped syntax" is not always valid.

Resources in RDF have a type, specified by an `rdf:type` property of that resource. RDF/XML allows a shortcut to this, allowing XML elements which specify nodes in the RDF graph to be a qualified XML name which encodes the URI of the type of the resource, instead of using the `rdf:Description` name. This is illustrated in Figure 2.18. This feature of RDF/XML is often called "typed nodes".

Another feature of RDF/XML is that XML literals may be embedded inside an RDF/XML document. In the graph, this is encoded as a typed literal with the special built-in type of `rdf:XMLLiteral`. In RDF/XML, an XML literal value is expressed by specifying the attribute `rdf:parseType` in a predicate node, as shown in Figure 2.19.

RDF/XML may also be embedded into other XML, providing the DTD of the other XML document allows this. XHTML 1.0 is therefore an exception to this, as its DTD does not allow for other XML to be embedded within it. There are several approaches which can be taken to

```
<?xml version="1.0"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:foaf="http://xmlns.com/foaf/0.1/"
  xmlns:svg="http://www.w3.org/2000/svg">
<foaf:Person>
   <foaf:depiction rdf:parseType="Literal">
      <svg:svg width="4cm" height="8cm" version="1.1">
         ...
      </svg:svg>
   </foaf:depiction>
</foaf:Person>
</rdf:RDF>
```

Figure 2.19: Embedding XML fragments inside an RDF/XML document

work around this, as presented in [135]. The suggested approach for XHTML 1.0 (and HTML 4.01) is thus to link the RDF metadata using a `link` tag. Unlike XHTML 1.0, XHTML 1.1 [4] and XHTML 2.0 [11] break compatibility with HTML 4.01, and allow arbitrary XML to be embedded within the document. RDF/XML may therefore be embedded directly into these formats.

As XML is conceptually a tree based data structure, while RDF is graph based, almost all RDF graphs have a number of possible representations as trees, depending on the starting node, the depth of traversal and where cross references occur (if at all). The order in which elements are listed is also arbitrary when the tree is serialised, as RDF as a set of triples is not sensitive to order. These ambiguities present an interesting challenge for comparing serialised RDF, and this is dealt with in more depth in Section 7.2.4.

2.4 Ontologies and Vocabularies

Global standardisation on formats or data schemas can be difficult in many domains. Consider an electronic data interchange for invoices and other commercial data. Information which is important, or necessary, in Peru may not be relevant, or may even be difficult to obtain, in Pakistan. Peruvians should not impose their data requirements on Pakistanis doing business with other Pakistanis. Only when two people from the two countries transact should a common ground

need to be found. In this case an inference language should be able to describe the discrepancies between the two languages in which the Peruvians and Pakistanis describe their transactions. An inference process would then allow the two parties to transact seamlessly, ideally providing enough common information for the the transaction to take place. This is because the inference language focuses on the semantics of the two languages in use, rather than the syntax or terms used in the two languages.

In the human case, the two parties use two different natural languages to write the invoices in, but the general form of the invoice will be similar to other invoices which the other party has seen, and it will therefore be understandable. Information such as local tax numbers or physical addresses may be irrelevant and therefore ignored by the receiving party.

In the machine case, the common ground is found by descriptions written using an inference language, which is able to describe the similar or identical terms used by the machine vocabulary used by the two parties. This inference process can help the receiving party to recognise the invoice, and effectively translate the components necessary to effect payment.

Thus, two local "dialects" of the languages of invoices are created, one for each country, and a translation between the two is performed by what is often called ontological glue. This ontological glue can be as simple as a set of statements stating which terms in the divergent ontologies have an identical meaning, thus allowing translation between the terms. Through the use of ontological glue, the convenience of local dialects and the power of global understanding can be brought together.

2.4.1 Examples

Here some examples of ontologies which are currently in use on the Semantic Web are presented. These ontologies have been chosen because each of them describes people in some way. Each has a different data model, and thus the semantics, as described by the vocabulary, are subtly different. The manner in which people and names are described in these vocabularies is discussed, with a focus on the key differences.

DC (Dublin Core)

Dublin Core provides a vocabulary for describing metadata about Web resources. Dublin Core does not prescribe a particular framework, such as RDF, and encodings are available for HTML (and XHTML) [140], XML [141] and RDF [123, 103].

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .
<http://russell.rucus.net/masters/> dc:creator "Russell Cloran"
```

Figure 2.20: An example of simple Dublin Core in RDF (N3)

```
@prefix dcterms: <http://purl.org/dc/terms/>
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
<http://russell.rucus.net/masters/>   dc:creator   _:a
_:a   a        rdf:Bag
_:a   rdf:_1   "Russell Cloran"
_:a   rdf:_2   "Fictional Person"
```

Figure 2.21: Qualified Dublin Core in RDF (N3)

Dublin Core maps well to RDF. A Dublin Core Element is a property of a resource [54], similar to the RDF notion of resources having properties. Two RDF encodings for Dublin Core exist – a simple encoding which includes an XML DTD, and is usable by plain XML parsers, and a qualified encoding, which provides a richer way of describing metadata, including the ordering of elements, and the specification of alternates.

The simple encoding is most interesting here, as it differs from other models, in that people are described only in terms of their names, as shown in Figure 2.20. The `creator` property of a resource has a string literal as a value, used for the name of the creator.

The qualified Dublin Core RDF encoding uses a model more similar to the two vocabularies discussed next, rather than the simple Dublin Core encoding. A blank node is used as the object of the statement with the `dc:creator` predicate, and an `rdf:value` property of that node is used to specify the literal value of the author's name. This can also be used with the RDF collections, such as a `Bag`, `Alt` or `Seq`, which allows listing multiple authors using the semantics associated with that sort of collection (unordered, alternates or ordered). Figure 2.21 demonstrates this usage.

vCard

The vCard standard [52] specifies a profile, or set of properties, used to describe information about people, most notably contact information, and is commonly used with the file format de-

scribed in [94] for the exchange of contact information. An RDF/XML representation of vCard objects has been formulated in [95], with a model which closely follows the original vCard model.

The RDF/XML formulation of vCard is similar to the qualified Dublin Core RDF encoding, in that people are described as resources which have properties such as a name (`vcard:NAME`), telephone number (`vcard:TEL`) or birthday (`vcard:BDAY`).

The RDF/XML formulation of vCard provides only limited RDF Schema information at the namespace URI, which does not describe what type of resources may be given the properties (that is, the valid domains of the properties) defined in the specification.

FOAF (Friend of a Friend)

The Friend of a Friend (FOAF) project [124] has created an RDF vocabulary [37] which has as its main aim the creation of social networks by allowing people to create a machine readable (RDF) homepage which describes the links between people, as well as allowing some detail about the person to be entered. FOAF is an interesting vocabulary because it has gained popular usage, and become the basis of other vocabularies which allow more precise description of people and the relationships between them.

Although FOAF is constantly evolving, the basic model is stable. The data model used in FOAF is very similar to the model used in the RDF/XML formulation of vCard, although a more complete RDF Schema document is provided. People are considered to be resources, and have a type of `foaf:Person`. Like any other resource, resources of this type can have properties, such as name (`foaf:name`), gender (`foaf:gender`) or home page (`foaf:homepage`). A complete set of terms is available at the FOAF namespace URI[2].

2.5 Querying the Semantic Web: SPARQL

Querying RDF data presents an interesting challenge. The language preceding Simple Protocol and RDF Query Language (SPARQL) [142, 20, 44], the RDF Data Query Language (RDQL) [148], was a member submission to the W3C, and was never standardised. RDQL was, nonetheless, widely used and implemented, for example in the Rasqal RDF Query Library[3], which is a

[2] `http://xmlns.com/foaf/0.1/`
[3] `http://librdf.org/rasqal/`

```
@prefix dc:   <http://purl.org/dc/elements/1.1/> .
@prefix dc10: <http://purl.org/dc/elements/1.0/> .
@prefix :     <http://example.org/book/> .
@prefix ns:   <http://example.org/ns#> .
:book1  dc:title    "SPARQL Tutorial" .
:book1  ns:price    42 .
:book2  dc:title    "The Semantic Web" .
:book2  ns:price    23 .
:book3  dc:title    "RDF Primer"
:book4  dc10:title  "The Semantic Web Lifts Off"
```

Figure 2.22: Data set for SPARQL query examples (N3 format). Adapted from [142].

part of the Redland RDF toolkit [17]. SPARQL is currently a working draft within the W3C, on track to be published as a W3C recommendation.

SPARQL consists of three parts, the query language [142], an XML format for returning query results [20], and a data access protocol [44] which specifies protocols for remotely querying RDF databases.

In the following discussion on SPARQL the data set on which the queries operate is described in Figure 2.22.

2.5.1 Basic Syntax

As the name suggests, SPARQL is similar to SQL [50], a design trait it shares with RDQL, and evidenced by the SELECT and WHERE keywords, as well as the general query structure. The primary advantage of this is that it is easy to learn for those already familiar with SQL.

The other important part of SPARQL is the triple pattern. A triple pattern is simply an RDF statement (triple) which can include wildcards instead of complete information. Triple patterns may be chained together, creating a complex graph structure which may be matched against. SPARQL is defined to use Internationalized Resource Identifier (IRI), a complement to URIs, which uses the Universal Character Set (Unicode) [57].

Figure 2.23 shows a simple SPARQL query, which extracts the Dublin Core property “title” from the book with IRI `http://example.org/book/book1`. The result of the query is returned in the variable “title”. The variable is signified by prefixing a question mark to its name in the

Query:

```
SELECT ?title
WHERE {
  <http://example.org/book/book1>
    <http://purl.org/dc/elements/1.1/title>
    ?title .
}
```

Result:

title
"SPARQL Tutorial"
"The Semantic Web"
"RDF Primer"

Figure 2.23: A simple SPARQL Query [142]

triple pattern – the triple pattern matches any triple which has the same subject and predicate as the pattern, but any object.

Variables

As shown in Figure 2.23, variables are used in place of unknowns. When the result is returned, it can be conceptualised as a table of results, with one variable per column. In Figure 2.23 the variable is `title`, and is written as `?title`. Variables can be prefixed with either `?` or `$` in order to indicate that the term is a variable. Triple patterns may use the predicate-object list (semicolon) or the object-list (comma) forms as they exist in N3, discussed in Section 2.3.4.

Blank nodes

Blank nodes used within the query may have labels which only have scope within the query. The label used has no relation to labels used within the data being queried.

Syntactic Sugar

Two other syntactic conveniences exist. Firstly, parentheses may be used to indicate RDF collections (described in Section 2.3.3), as illustrated in Figure 2.24. Secondly, the keywords "`a`", "="

```
(1 ?x 3) :p "w" .
```

expands to

```
_:b0   :p          "w" .
_:b0   rdf:first   1 .
_:b0   rdf:rest    _:b1 .
_:b1   rdf:first   ?x .
_:b1   rdf:rest    _:b2 .
_:b2   rdf:first   3 .
_:b2   rdf:rest    rdf:nil .
```

Figure 2.24: Using parentheses in a SPARQL query to indicate RDF collections [142]

```
?x a <http://purl.org/dc/dcmitype/PhysicalObject>
```

expands to

```
?x rdf:type <http://purl.org/dc/dcmitype/PhysicalObject>
```

Figure 2.25: Using the keyword "a" in place of the `rdf:type` URI [142]

and "=>" can be used as the predicate of a statement in place of the `rdf:type` , `owl:sameAs` and `http://www.w3.org/2000/10/swap/log#implies` IRIs, respectively, shown in Figure 2.25.

2.5.2 Advanced features

The features above allow the most simple and perhaps common SPARQL queries, along with some features which make the task of creating queries a little easier. In querying data on the Semantic Web, complex queries will occur, and cases of publishers using different vocabularies to describe the same type of information will also occur. The following are some of the features which SPARQL includes to allow these queries.

Query:

```
PREFIX  dc:  <http://purl.org/dc/elements/1.1/>
PREFIX  ns:  <http://example.org/ns#>
SELECT  ?title ?price
WHERE   {
    ?x ns:price ?price .
    FILTER (?price < 30) .
    ?x dc:title ?title .
}
```

Result:

title	price
"The Semantic Web"	23

Figure 2.26: Using the `filter` keyword to only include some results [142]

Filters and constraints

Data may be filtered out using a constraint. Constraints are expressions which have a boolean value, and many will be familiar to those who have used SQL or programming languages before. Figure 2.26 demonstrates the use of a filter to exclude results for a book search where the price is greater than or equal to 30 (or non-existent).

Optional Pattern Matching

Normally, when specifying a pattern to be matched, all components of that pattern must exist for it to match. The `optional` keyword may be applied to a pattern so that if data exists it is matched (and any variables within that pattern are returned), but it does not limit the result if the data does not exist. The use of the `optional` keyword to include results which do not include some of the data set is shown in Figure 2.27.

Unions

The `union` keyword is used to specify pattern alternation, in much the same way as the "|" operator in standard regular expressions [72] operates. This is useful when querying, for example, data which uses two different vocabularies, as shown in Figure 2.28.

Query:

```
PREFIX  dc:  <http://purl.org/dc/elements/1.1/>
PREFIX  ns:  <http://example.org/ns#>
SELECT  ?title ?price
WHERE   {
   ?x dc:title ?title .
   OPTIONAL { ?x ns:price ?price } .
}
```

Result:

title	price
"SPARQL Tutorial"	23
"The Semantic Web"	42
"RDF Primer"	

Figure 2.27: Using the SPARQL `optional` keyword to include results which are missing some data [142]

Query:

```
PREFIX  dc:    <http://purl.org/dc/elements/1.1/>
PREFIX  dc10:  <http://purl.org/dc/elements/1.1/>
SELECT  ?title
WHERE   {
   { ?x dc:title ?title }
   UNION
   { ?x dc10:title ?title }
}
```

Result:

title
"SPARQL Tutorial"
"The Semantic Web"
"RDF Primer"
"The Semantic Web Lifts Off"

Figure 2.28: Querying data from different versions of a vocabulary using the `union` keyword. Adapted from [142].

```
<html
  xmlns="http://www.w3.org/1999/xhtml"
  xmlns:data-view="http://www.w3.org/2003/g/data-view#"
  data-view:transformation="
http://www.w3.org/2003/12/rdf-in-xhtml-xslts/grokFOAF.xsl
http://www.w3.org/2003/12/rdf-in-xhtml-xslts/grokCC.xsl">
...
```

Figure 2.29: Using the `transformation` attribute as intended for general XML documents. Adapted from [90].

2.6 Gleaning Resource Descriptions

Not all information is published in RDF, nor is it desirable to require all authors to publish data as RDF. Instead, it may be possible for other document formats to be transformed into RDF through some sort of translation mechanism. More specifically, a restricted subset (or dialect) of the original language can be used to create semantics which can be easily translated to RDF. The mechanism for Gleaning Resource Descriptions from Dialects of Languages is known as GRDDL[90].

GRDDL currently has two profiles for transformation to RDF, one for XHTML and the other for general XML documents.

The XHTML profile takes advantage of the metadata profiles of HTML, as defined in HTML 4.01 [144]. When the document has a profile as specified by GRDDL, a link of relation transformation is taken to be a relation to a document which transforms the original document whilst preserving its meaning. Multiple transformations may be specified by specifying a number of links to different transformation documents.

In general within an XML document, a `transformation` attribute within the `data-view`[4] namespace may be used to reference a transformation. A number of transformations may be specified as a space separated list. Figure 2.29 shows a fragment an XHTML document which has been treated as a general XML document. By including the `data-view` namespace, and then using the `transformation` attribute from that namespace, three transformations for this XML document have been specified.

[4] `http://www.w3.org/2003/g/data-view#`

A GRDDL transformation can be specified either in the document in question, as shown above, or within the document referenced at the XHTML profile or XML Namespace URI. If the document referenced (the *namespace document*) includes an RDF statement which identifies a `profileTransformation`[5] property of that namespace document, then the value of that property can be used to transform the original document. For example, if `http://example.com/index.html` is an XHTML document, and references the profile `http://example.com/rdf`, which contains the following RDF statement (N3 format):

```
<http://example.com/rdf>
   <http://www.w3.org/2003/g/data-view#profileTransformation>
   <http://example.com/transform/rdf> .
```

Then `http://example.com/index.html` can be transformed with `http://example.com/transform/rdf`. Notably, the RDF statement does not have to exist in the RDF/XML form - GRDDL or a similar technology could be used to extract this statement from the namespace document.

2.7 Summary

The Semantic Web is a vision to create an information space with rich semantics. The aim is to create a common language which can be used to create a distributed database, which can be used by machines in advanced ways, for example supporting reasoning across data gathered from the Semantic Web. The Resource Description Framework (RDF) is the technology which is designed to realise the goals of the Semantic Web by allowing the creation of such a distributed database.

Creating a "web of knowledge" has a number of advantages, some of which have been demonstrated already by simple formats, like RSS, which have stronger semantics than largely presentation oriented languages, like HTML. Some of these advantages include:

- Re-use and aggregation
- Specific query
- Pattern analysis

[5]The full URI of the GRDDL `profileTransformation` property is `http://www.w3.org/2003/g/data-view#profileTransformation`

- Logic applied across data

The Semantic Web and its associated technologies have been slowly evolving for a number of years. "Web 2.0" is a relatively new term coined by Tim O'Reilly to describe a new way of thinking about the Web. Core to the ideas of Web 2.0 are user involvement harnessing network effects, realising the power of data, and providing software as a service rather than a product. Many technologies and ideas which are bundled together with the Web 2.0 idea help to improve the understanding of the Semantic Web, but the concepts are different. Web 2.0 describes a way of understanding and leveraging current Web technologies, whereas the Semantic Web proposes a new set of technologies to create a Web of data, most useful for machine processing.

The RDF data model is simple, consisting of a set of "triples". These triples can be understood as simple statements with a subject, a predicate and an object. URIs are used to reference resources, and any concept can be referenced by a URI, including, for example, the predicates of a statement. The set of triples can also be interpreted to be a graph, with the subject and object of each statement forming the nodes of the graph, and the predicates forming directed, labelled edges between the nodes.

RDF itself contains a minimal set of semantics, and allows other languages define this. Two such languages are RDFS (RDF Schema) and OWL (Web Ontology Language). Through RDFS and OWL concepts such as sets, functional relationships, and equality may be expressed.

RDFS and OWL provide the building blocks for other vocabularies (or ontologies) to be defined. Vocabularies allow the description of a particular type of concept, for example, people and related contact information. Whilst it may seem that allowing different vocabularies to describe the same information will lead to incompatible data, this is not necessarily the case. In fact, this is one of the strengths of the Semantic Web. By allowing information to be described using a locally relevant vocabulary, and then applying ontological glue, global understanding of the information can be achieved.

As a data model, RDF is simply a concept. In order to exchange information, some sort of serialisation is required. Simple serialisations, such as N3, are useful for describing RDF to humans, or for humans to input RDF. The powerful, and recommended, means of serialising an RDF graph is with RDF/XML, an XML based serialisation of RDF. By using XML as a basis for the serialisation of RDF, RDF is able to leverage the features of XML, such as internationalisation, and the powerful tools which have already been created for XML.

A number of RDF query languages exist, and one of the newest and most popular is SPARQL.

SPARQL has a SQL-like syntax, which makes it easy to learn for those who are already familiar with SQL, and a triple pattern notation which makes it simple to match arbitrary triples.

RDF data is intended for machine consumption, but for some information publishers it is not feasible to publish both human readable and machine understandable information. GRDDL (Gleaning Resource Descriptions from Dialects of Languages) allows information to be published using one language, but in a specific "dialect" of that language which allows it to be transformed into RDF. GRDDL, thus, can be seen as a means to bridge the current Web with a web of knowledge – the Semantic Web.

Using information in a completely automated process, as can be done with RDF data, brings with it the problem of determining which information to trust, as introduced in Chapter 1. The notion of trust is discussed in the following chapter.

Chapter 3

An Introduction to Trust

Artoo Deetoo, you know better than to trust a strange computer!

- C-3PO, The Empire Strikes Back [113]

The notion of trust is one that is difficult to pinpoint, and it is thus a difficult concept to define. When discussing such issues, it is important that some common definition is agreed upon to avoid confusion. In this chapter a number of different approaches to defining trust are explored. The author then attempts to find a definition of ”trust” which is suitable for use in this work. This investigation begins by exploring why it is important to have some notion of ”trust” on the Semantic Web. As the discussion develops it is discovered that it is important to be able to transfer trust between agents on the Semantic Web, and towards the end of this chapter some of the methods which are currently used for transferring trust amongst agents on the Semantic Web are discussed.

3.1 Why do we need “Trust”?

The Semantic Web (as discussed in Chapter 2), whilst difficult for most people to conceive, is simply an extension of the Web as many people currently know it. As a distributed document retrieval system, the Web allows any party to publish information, and to make this information available to any other party (or restrict access as they see fit). The Semantic Web uses this system of protocols as its core means of communication, but places a web of linked, machine understandable data on top of the document retrieval abilities of the Web. One of the visions

of the Semantic Web is that automated ”agents” will be able to search through the distributed databases available, and be able to process this data in intelligent ways [28], using more advanced techniques than those currently used in data processing (for example, machine-driven reasoning). The Semantic Web is a vision which seeks to enable a generation of computers and applications which work better for their users, by providing the information that those computers need in a rich format.

On the Web, it is up to the user (who reads web pages directly) to determine whether or not the information published by a web page is either credible or trustworthy (the distinction is further discussed in Section 3.2.3). This decision is often made with relative ease by skilled and experienced users, and previous studies which examine how users determine the credibility of a web page which they are reading are discussed in Section 3.4. For an automated agent, blindly trusting information which is obtained from the Web may lead to inaccurate or incorrect conclusions. The user of the agent may not question the results that the agent presents (perhaps due to a blind trust in technology [66]), or it may be difficult to diagnose which data or step of reasoning caused the agent to present incorrect results to the user. It is therefore important that an automated agent is able to assess the credibility of each component of data it has obtained from the Web in order to determine whether it should use the information for further processing or not. The decision of whether or not a piece of data is credible is a difficult decision for an automated agent to make, as it is unable to properly take into account either the context of the information, or some of the heuristics which a human user of the Web may employ to examine the source and quality of the information presented to them.

Automated agents such as these will be prevalent on the Semantic Web, and without a means of automatically assessing the credibility of information which they have gathered, the utility of such automated agents may be drastically reduced.

3.2 What is Trust?

While some work has been done on the topic of trust in information systems, the author believes that his initial thoughts about the topic are worth examining, as they indicate the essence of the work which he is trying to do, without clouding the issue with formal definitions, or definitions which are appropriate for other works. The discussion will return to other, more fully investigated definitions later in order to arrive at a final conclusion.

As mentioned above, an agent on the Semantic Web needs to be able to decide whether data it

obtains from the Web is fit for further processing, where its ”fitness” is not yet a clearly defined concept. Importantly, this fitness is the fitness of the data which is obtained, rather than trust of the person or system which generated the data. However, an assessment of every piece of data is unrealistic, and assessing an author (human or machine) is a practical means of classifying the information which that author has published.

Briefly (and with the understanding that this definition is a preconception), the author's notion is that *trust is the belief that data is accurate, or fulfills criteria which the consumer believes it should*.

3.2.1 Human trust

Gambetta, in “Can We Trust Trust” [73], presents a summary and conclusion for a volume of collected works which discuss why people trust each other and how that trust is built up. Most interestingly and concisely, a potential definition of trust is presented [73]:

> Trust (or, symmetrically, distrust) is a particular level of the subjective probability with which an agent assesses that another agent or group of agents will perform a particular action, both before he can monitor such action (or independently of his capacity ever to be able to monitor it) and in a context in which it affects his own action.

As discussed by Abdul-Rahman and Hailes in [1, 2] and Gambetta this definition has important implications:

1. Trust is subjective: Abdul-Rahman and Hailes conclude that because trust is subjective, trust systems based around a “trusted authority” are not valid. This could be extended by saying that a *group* or *global* rating of trustworthiness of an agent is not valid, because a trust rating is subjective for each agent. This has important implications for some of the trust metrics discussed in Section 3.7, which depend on the group metric.

2. Trust is important in actions which we cannot monitor.

3. “The level of trust depends on how our own actions are in turn affected by the agent's actions” [2]: Because trust is relevant only in a context in which it affects the agent which holds the trust, the level (or severity, or cost) to which a break in the trust affects the agent is important. This is discussed further later in this section.

4. Trust is relevant in a situation where ignorance or uncertainty is a factor: If the outcome of an action was known before the action, trust would be absolute and probably implicit. The uncertainty of the outcome is an important factor in the requirement for trust. While some level of trust may be gauged (as a probability, according to the definition) by various (subjective) criteria, the uncertainty still remains.

Fogg and Tseng present a different view on trust in [66], where it is asserted that "trust indicates a positive belief about the perceived reliability of, dependability of, and confidence in a person, object or process" [66].

Why do people trust?

Gambetta presents a number of arguments on whether cooperation between people (especially members of a community) is desired or required, with the conclusion that there are situations in which it is required, and situations in which it is or is potentially detrimental. Competition amongst people can have the effect of improving performance of an individual or a group, and can act as an important personal motivator. If put to the extreme, however, a complete lack of cooperation can have the effect of being detrimental to individuals and the group. Consider, for example, a community of "robbers and murderers". From the perspective of members of that community, some cooperation is important in order to assure self-preservation. Without such cooperation the community would simply not exist. From this it seems that some (unspecified) combination of competition and cooperation leads to the optimal situation, both for the individual and the group.

The prisoners dilemma, a classic problem in game theory, is stated anecdotally as the choice of a prisoner to either defect from or cooperate with a prisoner who was captured simultaneously. If one prisoner defects and the other cooperates, the defector receives the largest possible reward (for cooperating with the captors), whilst the prisoner attempting to cooperate with the other prisoner receives the smallest possible reward, or largest punishment (for being perceived as the only crooked prisoner by the captors). If the prisoners cooperate with each other, they receive a reduced penalty (increased reward), but if both defect against each other the penalty is almost as large (the reward as small) as if they had been defected against.

The iterated prisoner's dilemma, a variation on the standard prisoners dilemma where the game is played repeatedly, was introduced by Axelrod in "The Evolution of Cooperation" [10]. Cooperation in repeated instances of the game would lead to a large long term reward. This reward

would not be as large as if the other prisoner were to receive the smallest possible reward every time, however. The prisoners must trust that the other prisoner will act in the interests of both prisoners, i.e. they should always cooperate. This trust is not only one way, because it is not enough for Prisoner A to trust Prisoner B to act in Prisoner A's interests; Prisoner B must also trust Prisoner A to act in Prisoner B's interests. If Prisoner A *fears* that Prisoner B will not trust him, the best strategy is to defect. Thus it can be seen that reciprocal or mutual trust is important in building cooperation.

Importantly in real-world interactions, it is notable that not all possible actions of the other party are predictable, and therefore not all possible actions can be protected against. Because not all possible actions can be protected against, interactions with others require some level of trust. Real-world interactions often offer pre-commitments, with the possibility of recourse (or imposing a *cost* of some form) for defection. This is manifested with the existence of contracts which offer recourse.

What criteria are used to gauge trust?

Trust could be gauged differently by different people, because it is a subjective measure. Gambetta observes that our knowledge of human psychology is limited, and thus we cannot begin to understand all of the factors which may influence a person's beliefs. Trust is one such belief which may be held by a person, and so the factor's influencing one person's trust in another cannot be fully understood [73].

The nature of trust, can, however be understood in general terms. Gambetta observes that trust (or distrust) is a result of cooperation, but is also a possible precondition of cooperation. Trust, therefore, can be said to exist somewhere between faith and confidence. Faith, as a belief held possibly without any prior experience of cooperation, contrasts with confidence, an informed opinion of someone's ability to perform, possibly based on prior experience of performance. If initial cooperation is tentative, based purely on faith, it may become more confident as the belief that the other party is trustworthy grows through experience of that party's actions.

On the other hand, people are also prone to "blind trust", or trusting (or distrusting) a person to an unreasonably high degree, despite evidence which might suggest that the contrary is the reasonable choice.

An assessment of the cost involved for the trusted party of defecting can provide some clue as to the probability of whether or not the trusting party will request the action to be performed. If the

trusted party artificially imposed a cost on the trusting party (especially an unrealistically high cost), this could be described as coercion. In the face of coercion, trust is no longer valid, as the trusting party is not acting with completely free will.

Whilst these criteria for trust are noted, the degree to which they can be properly analysed is limited by the scope of this work. By noting them, the author aims to gain a better understanding of what trust is, and how it is used in a human context in order to build better models in a programmatic context.

3.2.2 Modelling trust

A level of trust can be specified as a probability (p), where the probability can be described as the belief that an agent will successfully and correctly carry out a specific action. However, at what point can a decision be made as to whether the subject is trustworthy enough to perform an action? This depends on the cost, to us, of the subject of our trust defecting. If the cost is low, we may choose to act even in the face of distrust. Conversely, if the cost is high, we may require an extraordinarily large amount of trust to be present before trusting the subject.

Further, there may be a cost to not acting because of a lack of trust. Gambetta [73] presents the example of a choice of restaurants at which to eat. Restaurants should be chosen for their ability to produce good quality food which is nourishing and will not lead to illness. Given a selection of restaurants, all of which are distrusted (perhaps because of the appearance, but as discussed earlier, the value of trust is subjective), the cost of not selecting any restaurant (starvation) may be higher than the perceived risk of choosing the restaurant perceived most trustworthy. Here, the cost of inaction outweighs the perceived risk of action.

If the subject is new, and there are no criteria available on which to judge them, a value for which to assign p is hard to arrive at. Should it be 0.5, indicating that the subject has a 50% chance of performing as expected, or should it be 0 or close to 0, indicating that the subject has a small chance of performing as expected? Similarly, a malicious agent should be distrusted, in other words the value of p for that agent should be 0 or close to 0. However, their action is unpredictable, and therefore (without further evidence), should be close to 0.5. Even by observing past behaviour of an agent known to be malicious, a high value for trust would be incorrect, as the agent may behave correctly 80% of the time, but act maliciously only when the stakes are high. The assignment of a numerical value for trust is thus difficult, leading to complications in modelling trust.

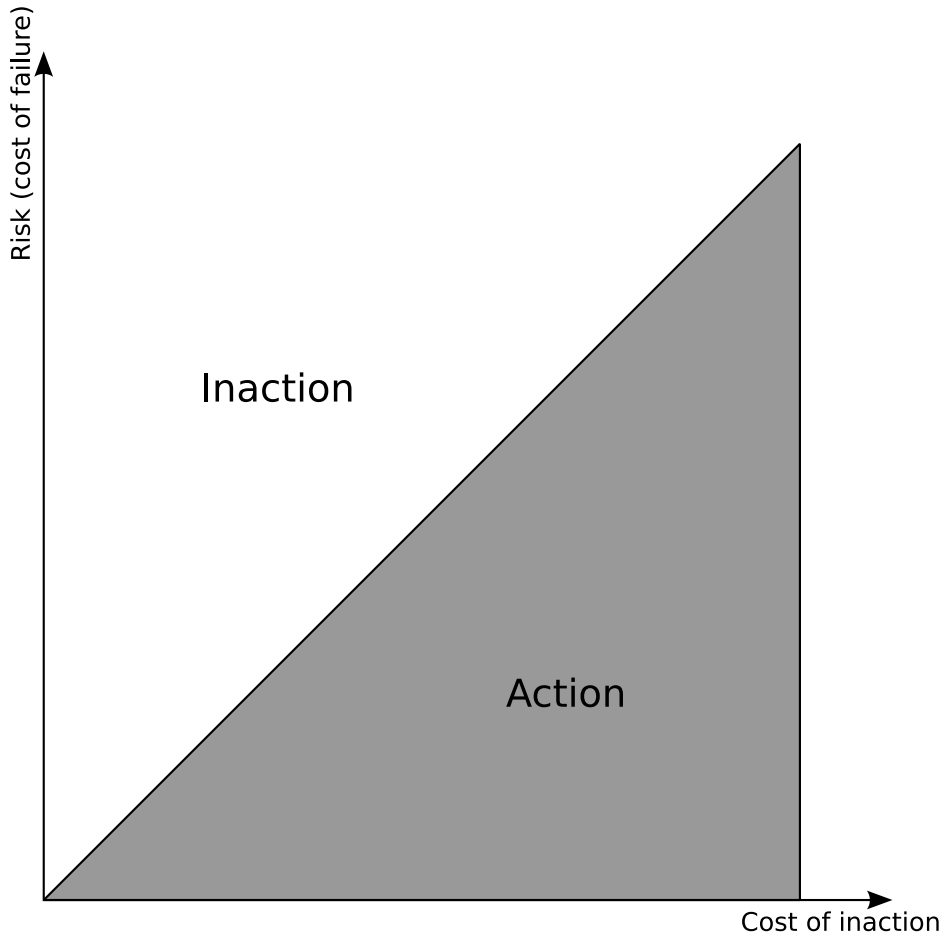

Figure 3.1: Taking action (trusting) as a function of risk and the cost of inaction

3.2.3 Trust, credibility, honesty, reliability

As discussed above, people will trust a person and therefore the information which they provide based on perceived trustworthiness. This perception may be based on many factors, such as perceived credibility, honesty and reliability. Many of these are human qualities, described by [101] as qualities of a ”passionate entity”. However, qualities such as reliability may also be attributed to data or to a computer system. In Section 3.2.1 the use of a probability as a measure of trust (called p) was discussed. In this section, the author will discuss a potential expansion to this idea, using familiar and intuitive concepts, and an intuitive understanding of them to increase the model from a single value to a more meaningful metric.

Credibility is defined as ”reputation impacting one’s ability to be believed” [161] and thus is a quality which can be assigned either to an agent or to data. Pages with poor presentation are often discredited by many people who surf the Web, as the presentation gives the user the feeling that the information may be of a poor quality too [65]. These factors influence the credibility of the web page (see Section 3.4). Credibility is a factor which contributes to the value of p, as perceived by the trusting party. Trust has been discussed not as a property of the target, but rather as a belief held by the trusting party. Credibility, on the other hand, can be seen in many ways as the mirror of trust – it is the set of properties which something has which leads to it being trusted. In other words, credibility is similar to believability [66].

“Perceived honesty” is considered as the belief that an agent is not malicious. Perceived honesty is the belief that the agent will act in the manner intended, and not wilfully *deviate* from this at indeterminable times. This deviation could lead to a decrease in the value of p, or, more accurately, could be expressed as an as yet undetermined deviation (for example, standard deviation). Let this deviation be called d.

Reliability is similar to honesty, but is not a result of malice. A lack of reliability is rather an inability to accurately fulfill expectations. Again, reliability contributes to p, but also contributes to the deviation from the norm, d. An agent which is honest and credible will receive a high value for p and a low value for d, but if the agent is unreliable, p will decrease for each failure, and the value of d will increase. [66] says that trust is a combination of perceived reliability, dependability and confidence.

Whilst the values p and d are intuitively neat, the author suspects that the use of a deviation value in a real metric may be less useful than the discussion suggests. Consider a trust value which is obtained from observation, where p is 0.5. Observations of actions can either lead to the contribution of a correct score (1) or an incorrect score (0) to the trust rating. Because all scores

are uniformly far from the mean, using a deviation measure would be meaningless. Standard deviation, in this case, would always be 0.5.

3.3 Properties of trust

When modelling trust relationships, it is useful to be aware of the properties which trust has. Trust relationships are said to have some degree of transitivity, should be composable, are personal and asymmetric.

3.3.1 Transitivity

The concept of transitivity of trust seems natural. If A trusts B, A should trust B to make recommendations to A about others. Golbeck argues [75] that a person should hold two beliefs about another person: trust in the person, and trust in their ability to make recommendations. According to other work, which suggests that trust differs depending on the knowledge domain, this is a false dichotomy, for trust may be held in a number of different domains. If everything else is to be grouped into the general trust group, what property of making a recommendation makes it deserving of its own special category? It seems as though the applicability to this particular problem, and a resolution between the simplicity of only one (or a few) ratings, and the accuracy gained from providing a large number of ratings prompted this separation. Golbeck goes on to argue that, in fact, the trust rating may be represented as one rating only, in order to simplify user interaction. Golbeck also claims that trust does not display "perfect transitivity", but rather the amount of trust propagated along a chain should gradually decrease.

3.3.2 Composability

Golbeck claims that when a number of recommendations about a person's trustworthiness are received, the trust values in those recommendations should be able to be composed into a single belief on the person's trustworthiness.

3.3.3 Personalisation

Another important property of trust is that it is a personal opinion or view, and because of this ratings on a trust network will seldom be consistently held.

3.3.4 Asymmetry

Related to the above is the idea of asymmetry. It is not necessary for B to trust A highly if A trusts B highly. B's trust in A is independent (depending on B's motivation for trusting in A – see Section 3.2.1) of A's trust in B.

3.4 How people assess the credibility of a website

Perceived credibility of information or the author of that information is potentially an important factor in assessing the trustworthiness of the information (or the author in general). The literature suggests that while people may say one thing about their assessment of the credibility of websites, the actions taken are different. Surprisingly [65, 64] shows that the most important factor used in the assessment of the credibility of a website is the perceived quality or professionalism of the layout and design. It is hypothesised that users judge the effort put into the content of a website by the effort put into other areas, of which design is the most prominent.

[67] makes a number of suggestions for improving the credibility of a website, which offer insight into the results of the study on how people assess a website's credibility. These suggestions are [67]:

1. *Make it easy to verify the accuracy of the information on your site.*
2. *Show that there's a real organization behind your site.*
3. *Highlight the expertise in your organization and in the content and services you provide.*
4. *Show that honest and trustworthy people stand behind your site.*
5. *Make it easy to contact you.*
6. *Design your site so it looks professional (or is appropriate for your purpose).*
7. *Make your site easy to use – and useful.*
8. *Update your site's content often (at least show it's been reviewed recently).*
9. *Use restraint with any promotional content (e.g., ads, offers).*
10. *Avoid errors of all types, no matter how small they seem.*

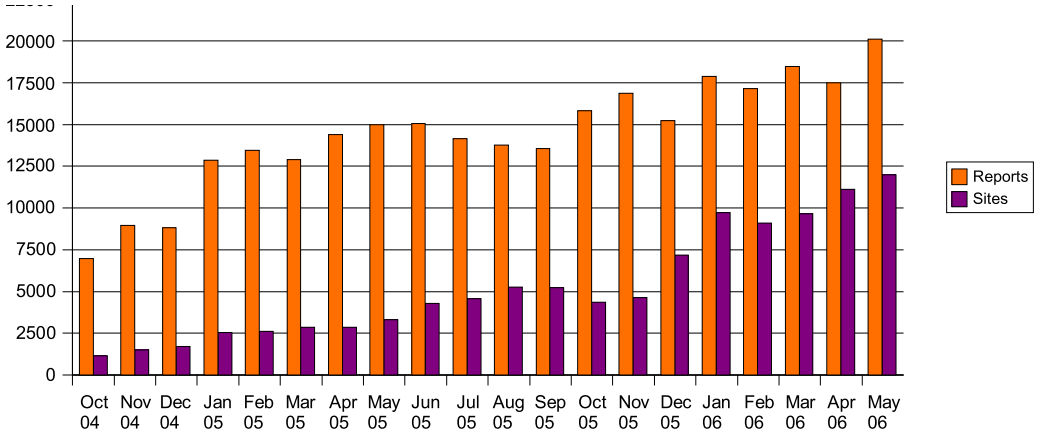

Figure 3.2: Incidences of phishing reported to the Anti-Phishing Working Group [6, 7]

These points do not mention factors which users use to decide whether a website they are using is the website they expect to be using. With a growing number of ”phishing”[1] attacks occurring through email and the Web [6, 7], education is leading users to become more aware of inconsistencies in web pages, such as irregular domain names or mismatching HTTPS certificates. Other factors which may catch a user's attention are uncommon practises, or unusual elements on a web page which is otherwise familiar. Each of these factors diminishes the credibility of a particular web page or website. These factors affect the user's trust in the authenticity or accuracy of the web page.

Chapter 4 discusses technically difficult ways in which users may assess the credibility of a website, and present a case study of a site which harmed its credibility by not following the guidelines above, as well as making other errors, further discussed in that section.

3.5 Trust in distributed systems

Abdul-Rahman and Hailes argue in [1] that trust is a subjective measure, and therefore must be distributed, in some sense. No central Trusted Authority can provide a perspective simultaneously congruent with the perspective of all users, and therefore trust has to be distributed. This

[1] phishing attacks attempt to lure web users into providing sensitive information (such as banking details) to unauthorised third parties, by posing as authorised parties.

does not necessarily imply that it should be a part of a distributed system, just that trust metrics should be calculated for each unique perspective on the system, rather than overall.

Josang discusses in [101] the meaning of trust with applicability to distributed systems. The focus of the work is on security, although the ideas are applicable in the Semantic Web area. One of the important insights presented is the separation of "passionate" and "rational" entities, and the work describes how each should have its trustworthiness evaluated differently. Josang describes trust as a positive concept, a belief that an entity is not malicious. This statement also illustrates two of his other arguments. Firstly, trust is not a property of an entity, but rather is some belief held by another entity about that entity. Secondly, the fact that trust is a belief means that only a passionate entity can evaluate another entity's trustworthiness.

Passionate entities are described as any entity which is human, or involves human interactions. Passionate entities are distinct from rational entities in that their behaviour is influenced by a large number of factors, and is thus not predictable. Josang argues that even the entity itself cannot reliably predict its own behaviour. This *free will* is what separates these passionate entities from rational entities. Trust in a passionate entity can thus be defined *[101]*:

> Trust in a passionate entity is the belief that it will behave without malicious intent.

Rational entities are simply described as entities which are not passionate – entities which do not have free will. The entity can therefore not make a choice as to its behaviour, as the behaviour is predetermined under any known circumstance. The belief in the accuracy of a system, therefore, is the belief that it is able to resist manipulation which will alter that behaviour. Trust in a rational entity can thus be defined [101]:

> Trust in a rational entity is the belief that it will resist malicious manipulation by a passionate entity.

Josang further claims that only passionate entities are able to assess trust [101]. This is based on the claim that trust is evaluated from a large number of sources, many of which a machine may not have or have access to. For example, trust may be based on experience, knowledge of the trusted entity's nature, recommendations, or faith. Put simply, "passionate behaviour can only be appraised by other passionate entities" [101].

Abdul-Rahman and Hailes discuss transitivity in trust networks in [2]. [101] implies that fully automated trust propagation is not possible because of the claim that only passionate entities may assess trustworthiness. Abdul-Rahman and Hailes allow conditional transitivity, if the following conditions are met [2]:

> a) Bob explicitly communicates his trust in Cathy to Alice, as a "recommendation"
>
> b) Alice trusts Bob as a recommender, i.e. recommender trust exists in the system
>
> c) Alice is allowed to make judgements about the "quality" of Bob's recommendation (based on Alice's policies).
>
> d) Trust is not absolute, i.e. Alice may trust Cathy less than Bob does, based on Bob's recommendation.

The trust model, based on the above, includes suggestions for

1. Decentralisation: Discussed above, decentralisation gives each agent the ability to exercise its right to ultimately manage its own trust
2. Generalising trust: To form a consistent model, the idea of trust is generalised and reduced to a trust category and a trust value
3. Explicit trust statements: This reduces ambiguity, and is analogous to recommendations having a separate category
4. A recommendation protocol: A specific protocol for exchanging trust information facilitates the exchange of trust related information

The necessity for a recommendation protocol is reduced in a system like the Semantic Web, where recommendations can be made through published information. The actual trust metric from [2] is shown in Section 3.7.3.

3.6 Trust in recommender systems

Recommender Systems [147] are popular, especially with online retail sites, for making suggestions to users about which items might be useful or appropriate for them. Many people will be

familiar with the recommendations which *Amazon.com* provide, based on criteria such as browsing and purchase histories. Two main types of recommender systems are prevalent: content based and collaborative filtering systems.

In a content based system, items (and their associated metadata) are compared to the user's profile to find items which strongly match what the user has specified as their area of interest or items which the user has previously purchased. The biggest problem suffered by such systems is that accurately tagging a large number of items is a time-consuming and necessarily centralised task. The task is necessarily centralised because there is no simple means of resolving discrepancies, such as which of two users' descriptions of some piece of content should be accepted.

Collaborative filtering [80] uses suggestions made by comparing a user's ratings of a number of items with other users' ratings of the same items, thus helping to find users who are similar in some sense. The system is then able to recommend items which "similar" users have rated highly to users who have not rated those items.

These techniques are not limited to online retail; another increasingly common application is news and blog services which offer filtering based on what other users are reading and have rated highly. Bloglines [163] and last.fm [108] are both examples of this. Bloglines provides recommendations based on people with similar RSS feed subscription sets, whilst last.fm provides recommendations based on what other people with similar tastes in music are listening to. Applications such as these are aimed at helping to overcome the so-called information overload.

Collaborative filtering is subject to three weaknesses, identified in [118]:

Data sparseness Because millions of items potentially exist in a system which uses a recommender system, and users only review a small number of items, the likelihood that two users have overlapping reviews is low. A value for data sparseness can be thought of as the number of cells that are empty when a grid of all users versus all items reviewed by the users is created. The data which was processed in [118, 117] consisted of 49290 users, who rated 139738 items with a total number of 664824 reviews. The sparseness of this data is therefore 99.99%. This leads to computational problems, as correlation between all users will be calculated, even if there is no overlap.

Cold start Similar to the data sparseness problem, new users who have a low number of ratings will obtain poor recommendations from a recommender system based on collaborative filtering, because matches will be very poorly generated.

Easily manipulated Collaborative filtering systems are open to manipulation by malicious users, for example a "copy-profile" attack, as discussed in [117]. An attacker need only duplicate a user's rating profile, and then add the item or items which they wish to see recommended. Such an attack is only feasible if ratings profiles are publicly available, and if the attacker is able to easily and inexpensively create new accounts.

All of these weaknesses may be lessened through the addition of a trust filtering mechanism. Instead of a new user having to review many products to overcome the cold start problem, only a few trusted friends with similar tastes could be specified, and their profiles used as the basis for recommendations. In this case, privacy issues should be closely considered, as a user will be able to know what items were highly rated by friends, which may have been private information. Using an attack-resistant trust metric would solve the manipulation problem.

O'Donovan and Smyth build on the work of [118, 117] in [128]. Producers (those giving recommendations) are given a profile-level trust rating, which rates how many times a given recommendation was correct (within a certain error threshold) for all consumers (those receiving recommendations) and all items, and item-level trust ratings, which rate how often a producer was able to give the correct recommendation for a particular item (to all consumers). For $CorrectSet(p)$ as the set of all correct recommendations made by a producer p, and $RecSet(p)$ as the set of all recommendations made by a producer p, profile level trust, $Trust^P$, is given by:

$$Trust^P(p) = \frac{|CorrectSet(p)|}{|RecSet(p)|}$$

Item level trust rating $Trust^I$ for producer p and item i, and for all consumer, item tuples (c_k, i_k) in the sets of correct recommendations and all recommendations is given by:

$$Trust^I(p, i) = \frac{|\{(c_k, i_k) \in CorrectSet(p) \ : \ i_k = i\}|}{|\{(c_k, i_k) \in RecSet(p) \ : \ i_k = i\}|}$$

O'Donovan and Smyth use the trust value both as a weighting to alter a standard recommendation value, and as a filter to disallow profiles which are not highly trusted from contributing to the recommendation value. Strategies which combined these two approaches were also used. They measured results against the standard Resnick recommendation formula [147] to measure the effectiveness of using trust to improve recommendations.

By filtering out profiles which had poor trust ratings, the experiments showed the most number of

improvements over the standard Resnick recommendation formula, with 69.8% of recommendations being better than those using the standard Resnick formula. Unfortunately, however these recommendations were not significantly better, showing only a 3% improvement in accuracy, with a mean absolute error of 0.84. By using the combined strategy of filtering and weighting on item-level trust, the number of recommendations that were better than the Resnick recommendation formula was almost as high (66.8%), and the recommendations were significantly better recommendations (22% improvement in accuracy, with a mean absolute error of 0.68).

Applicability to Semantic Web

The trust metrics above require a rating of similar users in order to be effective. Users with similar sets of ratings are used to obtain ratings about other products, with filtering or weighting based on their past accuracy. Canny suggests that such homophilous diffusion is not the only desirable means of propagating recommendations [39]. Instead, users often wish to turn to experts in an area to obtain ratings for items in that area. They therefore turn to dissimilar users, and a heterophilous distribution of knowledge occurs.

On the Semantic Web, trust is not only used for recommendations, but also general information consumption. Ratings based on accuracy might not always be possible, and therefore the trust computation might not be possible. Users may also wish to override or pre-seed trust values based on real-world interactions and preconceptions. Some sort of trust propagation, where experts' opinions can become valuable to dissimilar users or users without any experience of interaction with the expert, seems desirable.

3.7 Propagating trust: Trust models and metrics

It is not imaginable that every agent in a large social network can evaluate and make assertions about all other agents in the network. If we assume that evaluation of other agents is an expensive process, in some terms (time, bandwidth, computational power, human time, etc), then (at the very least) it becomes an expensive process in those terms. Section 3.2.1 describes trust as a subjective measure, falling between faith and confidence. Initial interactions are tentative, and based on a level of trust which is gained from some pre-existing knowledge. Trust metrics seek to emulate the ability to seek this initial level of trust; a trust metric should accurately describe an agent's trustworthiness for a particular action where no previous interactions exist for an existing trust level to be decided upon.

Trust metrics, therefore, seek to accurately predict the trust level which an agent should have for another agent. The entire trust system (the metric, plus protocols) should also be resistant to manipulation of these trust levels by malicious parties.

3.7.1 PKI/X.509

X.509 was designed as a means for providing authentication and access control for directory entries by binding public keys to directory names [165]. It was originally published in 1988 [42], and underwent three revisions [97, 98, 99]. The IETF produced an X.509 profile designed for operation on the Internet, known as PKIX [93], in order to reduce the generality provided by the ITU-T standards, and tailor X.509 for use on the Internet.

The X.509 Public Key Infrastructure (PKI) is based around the idea of a trusted Certification Authority (CA), which issues digitally signed (by the CA) certificates which embed the subject's name and public key, amongst other information. Certificates may also be chained, at the CA's option, and thus a hierarchy (tree structure) of certificates can be constructed, with the CA thought of as a "trust anchor". In order to check the validity of a certificate, a user need only check the issuer of the certificate, recursively if necessary, until an issuer which is a trusted CA is found.

X.509 certificates are familiar to users because they form the basis of HTTPS [146] – the HTTP protocol [62] over TLS (Transport Layer Security). CA certificates are commonly distributed with browsers (such as Internet Explorer or Mozilla Firefox), and thus the user's trust in a site ultimately passes through the browser vendor (and/or operating system), whom they trust to bundle only trustworthy CA certificates. Users may also add CA certificates which they trust to the bundle of their own accord. Browser vendors trust the CAs whose certificates they include in the bundle only to include details which they have verified to be correct in certificates signed by them. An untrustworthy CA may issue certificates which certify an incorrect binding between a name (domain name) and a public key, allowing false authentication to occur, and therefore presenting no warnings to the user.

Two approaches to ensuring trustworthy CAs are recommended by [93]. Firstly, CAs may create self-signed certificates: certificates which have the same issuer as they have subject. Secondly, CAs may cross-sign certificates, thus becoming trust anchors for each other.

In X.509, the only validation provided is the link between the name and the public key, and thus the "trust" is limited. If a CA is mandated with the task of only signing certificates of "trustwor-

thy" institutions, the task may become overly onerous, and will probably lead to certificates only signed for parties which have a point of view which concurs with the CA. For this reason, global trust metrics such as this are not desirable.

3.7.2 PGP Web of Trust

Pretty Good Privacy (PGP) is software designed by Zimmerman [167] to allow the secure exchange of files and messages, and is widely accepted as a means of providing confidentiality, integrity, authentication and non-repudiation (the goals of digital signature) for a low cost, compared to a PKI.

Similar to X.509, the validity of a name/key binding is certified by the digital signature of the user's certificate, where the name usually includes (or consists only of) the user's email address. However, the major contrast with X.509 is that there are no central certifying authorities, and it relies on previously known "introducers" to certify the key of a previously unknown user.

PGP also extends the ideas in X.509 by allowing users to specify four distinct levels of trust when signing a key, and thus a user's software can calculate a "score" for a public key which it has not previously encountered.

3.7.3 A distributed trust model

In [1], Abdul-Rahman and Hailes discuss the meaning of trust, and its definition. This contribution seems to be valuable. The other contribution of the work is a trust metric, which separates recommenders' trust values and general trust values. That is, they recognise that trust exists in different domains, but only separate recommendations as a domain separate from "general trust". The trust metric uses a scale of -1 to 4, as shown in Table 3.1. When judging a recommender's trustworthiness, the same scale is used, with values 1 to 4 indicating the recommender's closeness to the agent's own judgement about a third party's trustworthiness.

The trust value for a target T through path p is given by:

$$tv_p(T) = \frac{tv(R1)}{4} \times \frac{tv(R2)}{4} \times \ldots \times \frac{tv(Rn)}{4} \times rtv(T)$$

Where

$tv(Ri)$ Recommender trust value of Ri

Value	Meaning	Description
-1	Distrust	Completely untrustworthy
0	Ignorance	Cannot make trust-related judgement
1	Minimal	Lowest possible trust
2	Average	Mean trustworthiness. Most entities have this trust level
3	Good	More trustworthy than most entities
4	Complete	Completely trust this entity

Table 3.1: Semantics of trust values in [1]

$rtv(T)$ Recommended trust value of target T (by last recommender in the chain)

$tv_p(T)$ Trust value of target T through path p

By dividing the trust value of each recommender in the chain by 4, trust values are mapped to a $[0..1]$ range, if distrusted entities are never allowed to be in the chain. This would make sense, as a chain with two distrusted recommenders could, with this formula, create a positive trust value. The authors of the paper acknowledge that the formula is arbitrary (based on the author's intuition), but it demonstrates an important concept which seems to be prevalent in trust metrics: the longer the chain of trust, the lower the trust value will be.

The paper also discusses a protocol for the propagation of trust, including a subscription and notification system for revoking and refreshing recommendations. By adapting the protocol to only use requests, it could be made more appropriate for the Web, and thus be useful on the Semantic Web. The protocol should also include some form of access control. As discussed in Section 3.5, access control is necessary to prevent gaming of the system.

3.7.4 Advogato

The Advogato trust metric has gained much attention as an attack-resistant trust metric [109, 110, 111]. The metric has much in common with the PageRank algorithm [134], as a capacity constrained flow network. The idea of the metric is that good nodes will only rarely certify bad nodes, and although bad nodes might certify each other, trust will not flow from the bad section of the network to the good section. The Advogato metric is often seen as a global trust metric, with the implication being that it does not recognise trust as a subjective measure. The metric could, however, be implemented by each client, and thus be used as a local trust metric.

Levien shows in [109] that a capacity constrained flow network is best resistant to large-scale attack, and is on a par with the theoretical best performance of a trust metric.

3.8 Trust on the Semantic Web

With a clear understanding of what the Semantic Web is, and the technologies on which it is based, and now a clear understanding of what trust is, and ways in which it may be modelled, a discussion on trust and the Semantic Web together is possible.

The Web is a distributed information space, where anybody may publish any information they desire. The Semantic Web does not change the underlying protocols (like HTTP [62]) or the very basis of the Web, the URI [26], and is thus still an information space (tied together with URIs) where anybody may publish information (retrieved by HTTP). As discussed in Section 3.4, people assess the credibility of information retrieved from the Web in a number of ways, including visual clues such as the design of the site. On the Semantic Web, information is data with a simple form, and is designed to be processed by computers, which means that many of the means used to assess credibility on the Web are not applicable on the Semantic Web. On the Semantic Web, data should be assessed for trustworthiness before it is processed, so that results presented to the user (based, perhaps on many data sources) is accurate. Without a reasonable belief that a system will present accurate results, uptake of Semantic Web based technologies is not likely to be high.

Golbeck *et al* have produced the most trust research relating to the Semantic Web, published in [75, 76, 77, 78, 79]. Golbeck's greatest contributions in the area are two trust metrics which have been tested and measured [76], and an RDF vocabulary for describing trust relationships on the Semantic Web [74].

As discussed in Section 3.5, trust must be distributed. That is, each agent in the system has a unique perspective on the trustworthiness of other agents. Golbeck's trust metric is distributed in this sense, and in the sense that publication of trust values can be distributed and published like any other document published on the Web.

3.8.1 The trust ontology

The trust ontology is designed to extend the FOAF ontology, allowing users to describe how much they trust other people, in general or in a particular domain.

The ontology has changed compared to the description given in [79] and the description (and URI) given in [77]. The ontology as it is currently published[2] and described online [74] is

[2] `http://trust.mindswap.org/ont/trust.owl`

```
@prefix trust: <http://trust.mindswap.org/ont/trust.owl
_:a   rdf:type              foaf:Person .
_:a   trust:trustsRegarding _:b .
_:b   trust:trustSubject    <http://trust.mindswap.org/> .
_:b   trust:trustedPerson   _:c .
_:c   rdf:type              foaf:Person .
_:c   foaf:name             "Jen Golbeck" .
_:b   trust:trustValue      10 .
_:a   trust:trust10         _:d .
_:d   rdf:type              foaf:Person .
_:d   foaf:name             "Barry Irwin"
```

Figure 3.3: An RDF snippet in N3 showing example usage of the trust ontology

discussed in this section.

The ontology has a trust rating scale of 1 to 10, with no notion of explicit distrust, and allows the optional description of trust in specific knowledge domains. Figure 3.3 provides an example of both uses.

Figure 3.3 contains two trust statements. The first uses the `trustsRegarding` form to state that a `foaf:Person` with the name "Jen Golbeck" is trusted regarding the `trustSubject` which has a URI of `http://trust.mindswap.org/`, with a `trustValue` of 10. The second, simpler, form of a trust statement simply states that the first person has a `trust10` property, with a value of a `foaf:Person` with the name of "Barry Irwin". This statement means that the first person trusts the second with a value of 10 in all knowledge domains.

By allowing the explicit statement of trust using Semantic Web technologies, the trust ontology goes some way towards allowing the description of trust on the Semantic Web. With an openly usable description of trust values available, it is conceivable that the data could be aggregated and reused by a service which would infer trust ratings for as yet unknown users.

3.9 A trusted Semantic Web

Credibility of Web sites has become a concern amongst those who believe that the Web is a powerful tool, and has even lead to the formation of organisations such as Consumer Reports WebWatch [48], which "seeks to improve the credibility of content on the World Wide Web".

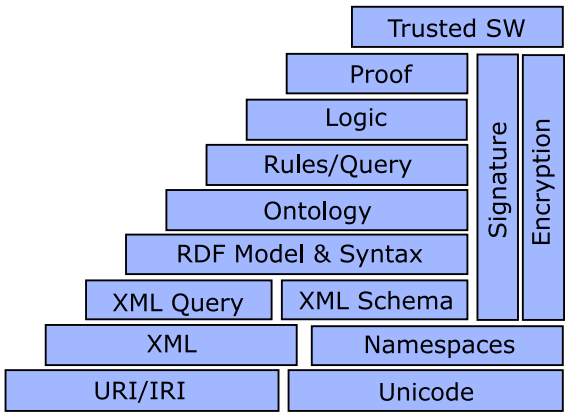

Figure 3.4: Trusted Semantic Web stack [25]

A part of the problem is that users may find they are unable to trust, in general, content found online [66, 64].

If the Semantic Web is to overcome this problem, the system, as a whole, must be seen as trustworthy, and must be largely resistant to common current attacks and problems, such as phishing and spam. The Semantic Web stack, shown in Figure 3.4, illustrates the technologies required to build a trusted Semantic Web. Key to this diagram are the placement of signature and encryption, which lie alongside the serialisation, logical model, ontology, query, logic and proof layers. Only together with the ability to prove where results originated, with reporting on the logic used to arrive at a result, will a truly trusted Semantic Web exist. The implication of this is that digital signature and encryption will be required to work with other technologies in order to support a Semantic Web in which users will be able to place their trust.

3.9.1 Social networks to propagate trust

Expressing trust in an author is a social expression, and falls well within the bounds of social networks and social networking software as it has recently become popular. In the Semantic Web context, FOAF [37] has become a common and popular RDF vocabulary for describing social network-like data. The use of FOAF in this research will be further discussed in Section 6.1.

Using social networks, it seems, is a natural method for the propagation of trust, and these

expressions can be created with FOAF and other vocabularies, such as Golbeck's Trust Ontology [74].

Privacy/social issues

On the Web, however, information is often published in a publicly accessible area, meaning that anybody could access the information. Two broad concerns about this are raised:

- If information is freely available, does this put the trust network at risk of manipulation (a security concern)?
- If information is freely available, how will this affect the social interactions of the parties involved?

The first concern can be addressed through research about the security of trust metrics, for example the work done by Levien in [109]. If necessary, access control (as provided by HTTP, for example) could be used to allow only trusted parties to access information – if it is not important that other parties are able to access it for their own trust determinations.

The second concern is more subtle. The author does not claim to understand all of the issues involved in it, as it involves complex social problems which computer scientists are not accustomed to tackling. However, consider a situation where a friend publishes poor quality information on the Web, for any of a number of reasons. Users who know this person, and the quality of the content published by her, should publish trust statements which effectively warn other users about the poor quality of her content. However, if trust is seen as one of the foundations of friendship (and the author suspects that it is) and the friend is able to see the low trust ratings, this could be detrimental to the friendship.

There are possible solutions for this problem, although it is beyond the scope of this work, and potential solutions are therefore not properly investigated. An authentication mechanism which only allows friends to get information about other friends (and not themselves) may be possible. This, however, would limit the depth of a distributed social network to two links away from the source, a major problem. As above, it would also disallow people with no authorisation from accessing the information, otherwise friends could simply access the information without login credentials to reveal the "secret" information, potentially leading to a breakdown in trust networks.

3.10 Summary

This chapter introduced the concept of trust, as something which sits between faith and confidence. Trust is a necessary precondition for cooperation between two agents, but trust also flows from successful cooperation. It is important where there is no full knowledge, or uncertainty of any kind may exist.

Trust is gauged differently by different people, and so it is a subjective measure. It is a belief held by an agent about another, and is not a property of a person. From this, we can also deduce that trust is asymmetrical – A does not have to trust in B simply because B trusts in A. Trust also occurs in different areas, or knowledge domains, categories, or subjects. Trusting somebody as a motor mechanic does not necessarily imply trusting that person in, for example, computer science. Trust is transitive in a limited sense. Trust communicated from one party to another, about a third party, is called a recommendation. Many trust models view these recommendations as a separate knowledge domain.

Trust metrics, which attempt to model how humans trust in the real world, have been used in different types of distributed systems, and recommender systems to help improve the accuracy of the systems. Attack resistant trust metrics are also available, and are resistant to manipulation my malicious parties.

Many systems on the Internet currently have some model of trust. X.509 (PKI and/or CA) and PGP are examples of those currently in use. A decentralised, peer-to-peer model of trust was thought to be the most effective for the Semantic Web, where social networks could be used to propagate trust.

The following chapter looks at how people can check the credibility of a website, based on its author, using a number of advanced techniques, and is based on a case study.

Chapter 4

Advanced checks for website credibility

Research, discussed in Section 3.4, has shown that humans assess the credibility of websites based largely on superficial factors, such as the appearance of the website. Further methods by which the validity of a website may be checked are shown in [46]. These methods are largely intended for technically inclined users, although there is no reason why a good user interface could be designed to simplify the process. In the same research, a case study of a website which seemed suspiciously malicious, but which was in fact benevolent and trustworthy, was presented.

In this section, the original reason for suspicion is described, and the additional checks used to come to the (incorrect) conclusion that the website was of a suspicious nature are presented.

4.1 Background

On 15 July 2004 a South African newspaper, DIE BURGER, published a story titled "Internet-sindikaat kry ook SA kredietinligting" (Internet syndicate obtains SA credit information). An English version of the story appeared on `news24.com` under the title of "419 scammers hack eBay" on 26 July 2004. The article claimed that advanced fee fraudsters had hacked into the eBay database, and obtained a list of credit card numbers. The article provided a link to `419legal.org`, and a link to a page which allowed the user to check whether their credit card number was on the list of numbers which were illegally obtained. The `419legal.org` website claimed to be affiliated to the South African Police Services (SAPS), and contained SAPS logos in the site banner.

By 28 July the issue had been resolved, and the police had made a statement saying that Rian

Visser was a police officer of the SAPS, and that he did run the `419legal.org` website, as the website claimed.

Reason for suspicion

The check on the credit card numbers was performed by allowing the user to enter their full credit card number into a web form. When the author initially visited the site, this form submission was across a plain text Hypertext Transfer Protocol (HTTP) connection, but the next day the form was submitted across a Secure HTTP (HTTPS) secured link. This raised the author's suspicion, as well as the suspicion of a number of the author's peers, as it is poor practise to provide credit card details to anybody who is not fully trusted. Investigation of details surrounding the website turned up many loose ends, and not much consistency in the information available. It was suspected that the website may have been the front for a "phishing" operation, a type of attack in which a user is lured into providing details to a third party. Ironically, this was one of the types of operations which the website advertised that it was fighting.

4.1.1 Special note

The author would like to point out that at the conclusion of investigations by both himself and the journalists involved, we were happy to find that `419legal.org` was a legitimate website run by a police employee, Rian Visser. This work is not intended to discredit Mr Visser or the organisations in which he is involved: the author is satisfied that the website is credible, and indeed run by Mr Visser, himself a credible person involved in the investigation of advance fee fraud.

This work only uses the `419legal.org` site as a case study due to the interest it created amongst the South African Internet community which the authors involve themselves in. Despite a number of loose ends and suspicious information, 419legal.org was shown to be credible, which is part of the interest in the site. A number of means for improving the credibility of the `419legal.org` (or any other) website through a few simple technical changes are suggested in the remainder of this chapter.

4.2 Web search

Search engines are tools which index the Web, and provide free text search and search with simple operators to users. On the whole, these tools aim to be impartial, and provide a useful means of quickly checking for information related to a particular site. Some techniques for using search engines, and the shortfalls involved in these techniques are presented in this section.

4.2.1 Names involved

A simple first step in finding out about a website may be to find the names of the people involved on the website which is being checked, and then searching for these names on the Web. Google [38] is currently one of the most popular Web searching tools in the world, if not the most popular. It is well recognised as being an impartial search engine which turns up accurate results. Google can thus be used to search for names, given that the names of the people who created or administer the website being checked are known.

This method is similar to the method described by Reagle [145], who suggests that "cryptographic signatures themselves might not be necessary to make a reasonable trust evaluation about a statement that has had time to grow into the tangled root structure of the Web." Whilst Reagle is talking about Semantic Web technologies, his suggestion may be taken further to simple facts which may be written about on the Web, such as names.

At first glance this method seems to be a good first step, but without suitable cryptography is is conceivable that an attacker may set up a website which takes advantage of an already well established name on the Web. In this point, it is necessary to diverge slightly from Reagle's hypothesis about Semantic Web technologies – the statement which can be verified through preponderance could be taken to be "a person with this name is involved in the subject with this website deals with". In most cases this statement will be implicit. Reagle's suggestion may not be entirely appropriate here, because there would not ordinarily be a large amount of information on the Web saying that the person who created, or was mentioned in, other websites of the same topic is the same person involved in this website, the only link would be the weak link of the same name.

Case study The author questioned the claims of the owners of the website on its own forums[1], as well as on a page on one of the author's blogs. We received the response that we should simply

[1] `http://419legal.org/index.php`(Unfortunately we do not think that this URL will be long lasting, it has already changed once since we have known about it)

"Google for [the name involved]" which would easily show that this person was a credible and well known fighter against advanced fee fraud.

Searching for "Rian Visser" revealed a number of web pages by and about Rian Visser. It appeared that there were two people who featured highly: A Dutch author of childrens books, and a South African who was involved in fraud (especially advanced fee fraud) investigation. This second profile fitted the person who claimed to be the author of `419legal.org` perfectly.

Whilst a large network of web pages and sites such as those found on Google would have been difficult to fake, we found it plausible that a fraudster could take advantage of the wealth of information already on the Web, and create a website which claimed to be owned by this person.

Solution A solution to this problem is to take advantage of some sort of cryptography. The author of the web page need not have been certified by people who could have reached through a trust network (such as a PGP web of trust), or even by anyone else at all. The simple fact that he would've been able to sign the new document with the same key as the large body of information which was also found by and about them on the Web would have been enough to convince anyone investigating an unknown site's credibility that an attack such as that described above was not being carried out. Other mechanisms, such as Thawte's free Web of Trust product[2] could have provided an even better assurance of the identity of the website administrators.

4.2.2 Web linking

The design of the Web is such that a link may be created from any page to any other page, but this link is a one way link only. This is part of the power of the Web; anybody can link to anybody else, but nobody can force anybody to link back to them. Some minimal semantic value is therefore encoded in a link, and this can be thought of as a vote for the page which the link is directed at. This hidden semantic value was recognised and harnessed by the creators of Google, now a popular and prominent Web search engine. Although these limited semantics are not explicit, and therefore not of great value, there is value for a user in being able to find other pages which link to a page. This gives some idea of related pages which the author of the page in question may not have linked to, and an idea of the popularity or perhaps even credibility of the page.

[2] `http://www.thawte.com/wot/`

Google provides a number of advanced queries, one of which allows a user to search for web pages which link to a particular page, and another which allows a search within a certain sub-domain. By carefully examining which pages link to a page which is being investigated – for example, the textual context in which the link exists – we can get an idea of the credibility of a certain page. The absence of links can, too, be taken as a possible indication of a lack of affiliation.

Case study Searching for "link:419legal.org" on Google at the time of the investigation offered a large number of results for websites about advance fee fraud which linked to 419legal, however a search for "site:gov.za 419legal" provided only one result, which was not in the South African Police Services (SAPS) domain. It was expected that a website which was affiliated to the SAPS would have had a link from the SAPS website, however no such link was found.

The large number of links from other websites dealing with advance fee fraud provided evidence which strengthened the belief that 419legal.org was a credible website.

Solution If `419legal.org` was linked to from a web page which should have been undeniably trusted, such as the SAPS web page on advance fee fraud, doubt as to the authenticity of `419legal.org` would have been greatly diminished.

4.3 Domain registration

The Domain Name System (DNS) [126, 127] is an important part of Internet infrastructure, which has as its main function the translation of easy to remember names into IP addresses used for Internet communications. Domain names must be registered, and this registration provides an avenue for checking details which are provided.

4.3.1 DNS

The Domain Name System is a hierarchical system [125, 139] which allows administrators of a domain to delegate responsibility for a sub-domain to another authority. An interesting implication of this is that there is a path of delegation which could be used to create a trusted path, similar to chained certificates in a PKI. Whilst is is not necessarily true that the path can be trusted for

commercial domains such as `.com` or `.co.za`, where very little fact checking occurs there is a stronger trust implication with government domains such as `.gov.za`. This means that it may be reasonable for a user to assume that a website within a government domain is accurate, because the administrators of that website are trusted by the administrators of the domain.

Case study In the case of 419legal, it was claimed that the website was run by a detective in the South African Police Service. It may have been reasonable to assume that if this website was a legitimate part of the SAPS they would have either obtained space on the SAPS website, or a sub-domain under the `.gov.za` domain or one of its sub-domains. Even a domain registered within the `.za` top level domain may have seemed more legitimate than the `.org` sub-domain which was registered. Because the domain was registered within the `.org` top level domain it was impossible to verify the identity of the registrant through DNS or whois, which will be discussed in Section 4.3.2.

Solution Creating your website in the correct domain can mitigate any confusion surrounding your website. If a user-friendly domain name is required, then techniques such as HTTP redirects may be employed to allow users to use the name which is easy to remember, but actually host the site on the domain, which will become the canonical name for the site.

If this is not desirable, or not possible, even a link from a page on a website in a valid domain might be sufficient. This is discussed above in Section 4.2.2.

4.3.2 whois

The whois protocol [86] was originally designed to run off one central server. As the size of the Internet became such that this was not possible, other protocols were developed, such as rwhois [162]. The nature of the service has remained the same, it is a means of querying a server for information about a particular object, be it a person or an Internet host name.

whois servers, in general, use a non-standard format to record information (there is no widely published standard format); the information is intended to be human readable, and not necessarily machine processable. For example Uniform, the authority for the .co.za domain, accept an email form as a part of the registration, and simply use the relevant parts of this form as the information provided by the whois server. As mentioned in Section 4.3.1, the DNS can be thought of as a

path of trust, and since domain authorities often run whois servers for the domains which they control, whois provides the means to check information of domain registrants.

Domain authorities for large domains, such as `.org` or `.co.za`, often do not check information of domain registrants. Registrars, which provide information to the domain authority in some domains, may also not be trusted to check information thoroughly. The utility of the whois servers provided by domain authorities is thus limited because the trust chain discussed in Section 4.3.1 does not always properly hold up.

Case study The `419legal.org` site was registered in the `.org` domain, a domain which uses registrars to do registrations. The information in the registration is not included in full, but in summary, the registrant was "D. Squire", the organisation which the domain was registered for was "E-Payments" and the physical address was recorded as an address in Hillary, Durban. The email address provided was in the `e-payments.co.za` domain, and the telephone number provided was a number which appeared to be a telephone number for Kloof, an area of Durban which is not near Hillary.

This information did not match any of the other information available, and visiting the website `http://www.e-payments.co.za/` did not provide any more useful information. The author believed at the time that "D. Squire" had registered the domain on behalf of Rian Visser, or `419legal.org`, an assumption which he still believes to be true. It would have been possible, however, for "D. Squire" to act as an agent for `419legal.org`, and register the domain in their name, or in the personal name of one of the people highly involved in setting it up. The fact that "D. Squire" had registered the domain in his name, in fact, provided a possibility for contacting "D. Squire" by telephone to verify the information about `419legal.org`. This path, however, is still open to attack.

Solution Information published by whois servers should not be trusted as authoritative or correct. In verifying the credibility of a website, checking whois information could confirm information already held, but it should not be regarded as holding much weight.

Stricter checking by domain authorities or registrars could be implemented to improve this situation, but this may prove prohibitively expensive.

Registration in the name of the person holding the domain would go one step towards improving the apparent transparency of the information provided, but may not add any real value to

the credibility of the website. As discussed above, registration in an agent's name may in fact improve the situation by allowing checking through a different avenue.

4.4 Professionalism

Professionalism is something which is hard to pinpoint. It is an overall impression created by the website in question; the manner in which things are done. A study on Web credibility found that a large percentage of users look at the overall design of a website as an indication of its credibility [65]. Website design is not the only point of professionalism, others include consistency over pages on the website and quality of information provided. The Stanford Guidelines for Web Credibility suggests 10 ways to improve the credibility of a website [67], many of which discuss the professionalism of the site.

Case study Although this is subjective point, the `419legal.org` did not leave an impression of professionalism in its design. Images were poorly scaled, colours were poorly selected and the layout was not optimal. The information provided on the site was also in English, presumably to improve international communication, but a lot of it was clearly written by a person who did not speak English as their first language.

The secure page for checking credit card details was hosted on a different site, as discussed in Section 4.4.1. This is also a matter of professionalism. Although a certificate just for 419legal.org may have been prohibitively expensive, an operation which wished to give a professional image would have found a way to cover the costs.

Since the incident the web page has changed drastically, and URLs for pages have also changed. This is a point of professionalism, discussed by the creator of the Web, Tim Berners-Lee in a short article called "Cool URIs Don't Change" [22]. This includes sections which are important to the website, such as its user forums, as noted in Section 4.2.1.

Solution Although costly, investing in a well designed site can improve its credibility. This also applies to copy which will appear on the site. By improving design, transparency, and quality of information, `419legal.org` would have shown that they were serious about their website, and thus improved the credibility of it and of themselves. Making use of an HTTPS certificate, as discussed below, would also improve the professionalism of the site.

4.4.1 HTTPS and PKI

HTTP over TLS (Transport Layer Security) [146], often referred to as "HTTPS" for the URI protocol identifier it uses, provides a means for transporting HTTP [62] traffic over TLS [55]. HTTP over TLS requires that a client match a certificate to the host name being queried, unless there is a good reason that that can not or should not be done, for example, the server is on a dynamic IP address and it does not have DNS which can be frequently updated. Certificate Authorities issue certificates for a domain, and check the details of the person or entity requesting the certificate. This means that there is now a verified link between the domain name and the person or company's information provided by the certificate authority, a trusted third party. HTTP over TLS also provides end to end encryption, which means that information can not be read by a third party while it is in transit.

Case study 419legal.org provided a facility to check whether a credit card number existed in the database of stolen credit card numbers that they held. The web page providing this facility[3] was hosted on `www2.securesiteserver.co.uk`, and the certificate was issued to a company called Donhost Ltd. There was no way to check this further relationship on the Internet.

Solution TLS certificates should be issued to the organisation which is going to be the end user. This allows a user to check that the website does belong to the organisation which claims to own the website. Using HTTP over TLS as it was used on `419legal.org` added confusion, however at least end to end encryption is provided.

4.5 Summary

The case study presented in this chapter shows how a web page which is poorly set up can do damage to its credibility despite the fact that it is valid. The use of search engines, the structure of the domain name system and domain registration information were discussed as ways of checking the validity of a website.

These advanced checks were done by a user with a better understanding of the technologies behind the Web than most users, as a means to gain a better understanding of how trust in a website may be checked. Many of the technologies behind this particular website were poorly

[3] https://www2.securesiteserver.co.uk/cbranch/card1.asp

configured, and it was shown that even technologies designed to increase a user's ability to place trust in a website, such as HTTP over TLS (HTTPS), can be poorly used, thus diminishing their utility in verifying the authenticity of a website.

This case study leads the author to believe that the use of technologies not explicitly designed to prove trustworthiness may not be useful in determining the trustworthiness of data gathered from a website.

Chapter 5

An RDF based website

The development of a social networking website, using Semantic Web technologies, was intended to fulfill the following goals:

- Serve as an investigation into programming using RDF based APIs
- Provide data which would form the basis of the measurement of trust metrics

Clique is a social networking website, written in PHP using the Redland library [17], and the PHP bindings provided. It aims to be a building block for socially based applications, such as sharing photographs, or making introductions for professional purposes. The implemented functionality is similar to the basic functionality of websites like Orkut[1] and Friendster[2], and allows the user to maintain a list of friends, and view their contact details.

In this section the development process is discussed, and some of the design decisions rationalised. Some of the happy coincidences that occurred as a result of the design decisions are also recorded, as well as some experiences with the periphery software used and tested.

5.1 Motivation

The motivation behind the development of Clique was that data was needed for the analysis of trust networks, amongst the social networks that would be built in Clique. This data may have

[1] `http://www.orkut.com/`
[2] `http://www.friendster.com/`

been obtained elsewhere, but there were two motivating factors behind the development of a new social network. Firstly, it was thought that the fact that most of the users would be students at Rhodes University would provide an extra avenue for the author to check data for validity, via student databases or similar channels. Secondly, the development of functional software using RDF based libraries would allow experimentation with this framework, giving a deeper understanding of the concepts, problems and advantages it presents.

5.2 Responsibilities of RDF tool authors

Users use tools to create content which will be published. It thus stands to reason that the quality of output of these tools will determine the general quality of the content, or at least the quality of the document structure, will be determined by the tools being used. Although a tool for publishing RDF should allow the user freedom to express the useful relationships between objects and ideas in her world, it should also prevent the user from creating expressions which are meaningless, or otherwise useless. Most importantly, it should prevent the construction of documents which are *syntactically* invalid (and therefore completely useless, as they cannot be parsed). It may also be a matter of discussion whether tools should allow users to publish information which is incorrect, or information which that user should not be publishing; should a tool prevent me from publishing all of the private information about another person?

A user's perception of the Semantic Web could rest on the tools which tout to be Semantic Web based. Without good tools, the potential of the Semantic Web will not be realised, not only because the tools are poor, but also because users may not participate in something which they initially perceive to be poor. It could be argued that users trust of the Semantic Web as a whole rested on their trust in the tools.

A thorough discussion on the role that authors of tools play, and the responsibilities which they therefore have, is beyond the scope of this work. It is, however, worth noting that the quality of tools available will have some sort of impact on the quality of data published, if only at the syntax level. Authors of tools have some responsibility to allow users to fully express whatever they wish, without allowing them to publish syntactically invalid documents, and this was one of the design goals with Clique. Unfortunately, this goal was not fully achieved, as shown in Figure 5.7, and discussed in Section 5.6.3.

5.3 RDF toolkits

Creating a website using the concepts in RDF is a large task in itself, and so using existing ready-to-use toolkits which abstracted many of the RDF ideas into usable APIs was a necessity. PHP was chosen as the language of implementation, because of the author's familiarity and comfort with the language. The task of finding an RDF toolkit was thus narrowed to finding an RDF toolkit which could be used from PHP. The choice came down to two toolkits, the RDF API for PHP (RAP) and Redland. RAP was used for an earlier project (which became the basis of some parts of Clique), but the development of Clique itself was done using Redland. In this section, the two toolkits are briefly discussed, and the choice of Redland for the main development is explained.

5.3.1 RDF API for PHP

The RDF API for PHP (RAP) [130] is written in PHP, and can parse and serialise RDF/XML, maintain and manipulate RDF models in memory, and store these models in a relational database. Since the initial investigation and usage of RAP, it has grown to include various means of accessing the data (statement-, resource- and ontology-centric access methods) as well as more advanced query interfaces, for example the support of SPARQL (see Section 2.5).

RAP was used as the server-side component of FOAFbuilder[3], a Web-based utility which allowed a user to build a FOAF file without understanding of the complexities of RDF. FOAFbuilder formed the basis of the personal profile editor found in Clique, and is discussed further in Section 5.6.3.

5.3.2 Redland

Redland is a popular RDF framework written largely by David Beckett in the C programming language [17]. It is robust and fast, and has been ported to a number of platforms. It is written using object oriented principles, and bindings in a number of different languages are available which take advantage of the object oriented design.

One of the main advantages of Redland over RAP is speed. Because Redland is written in C, and PHP bindings make it available from PHP, it is much faster than RAP, which is written entirely in PHP.

[3] `http://russell.rucus.net/2004/foafbuilder/`

Redland is also open source software, and although not an advantage over any other RDF library considered (they were all open source), this was an important criteria. Open source software is available with full source code, and a license which allows redistribution, such as the GNU GPL [151], ensures that the code will always be openly available. This is important if bugfixes or customisations are required.

In line with these decisions, PHP was chosen mainly for the author's familiarity with it, but also because it is a widely used, well supported programming language. Retrospectively, the low quality of the PHP bindings for Redland, slowed development somewhat, and the choice of another language, such as Python, may have saved some time in development, but because the issues were quickly addressed by the Redland developers, this was not an overwhelming problem.

The Apache Web server was chosen for its speed, security, and excellent support for PHP, and no problems in this choice were encountered.

The large community around Redland also promised a better chance of the software being improved in the future and expanded. Because a number of bindings for different programming languages existed for Redland it was possible to easily write tools which interfaced with the Clique database, without being stuck developing in PHP.

Models, streams and storage

Models in Redland are represented as an object, containing a set of statements. Simple operations, such as adding and searching for statements can be performed.

Many Redland search operations return a stream object, which is also a set of statements, but which must be iterated through. Stream objects may be added to the model (or another model), serialised or otherwise manipulated programmatically.

Searches can be performed either using a full query language, such as RDQL or SPARQL, or by searching for statements which match a certain criteria. The criteria is specified by creating a partial statement, with an empty (`NULL`) node for any of subject, predicate and object. Statements which match the statement criteria which are provided are returned. Additionally, these searches may specify one context from the model which should be searched on.

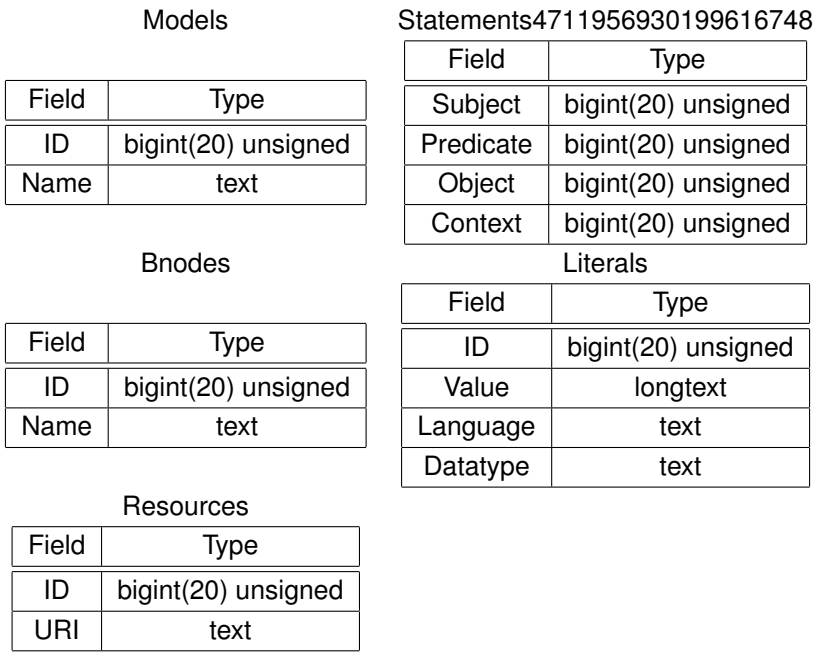

Figure 5.1: Database schema for a database created by the Redland MySQL storage module

The MySQL storage module for Redland

Redland provides a number of storage modules, including memory-only implementations, and persistent implementations with back-ends such as Berkeley DB (Sleepycat) and MySQL. The MySQL storage module was written by Morten Frederiksen [17].

One of the advantages of the MySQL storage modules over the others is that the MySQL database is easy to manipulate without the overhead of interacting through Redland. This manipulation should be carefully handled, so as not to produce invalid RDF data. The most useful aspect of the direct access is in reading information by using carefully designed or optimised queries, rather than the queries generated by Redland itself. The database schema is illustrated, by way of example, in Figure 5.1.

The `Models` table contains a list of the RDF models contained in the database. A `Statements` table exists for each separate RDF model, with the ID of the model appended to the word "Statements" to form the table name. The `Bnodes`, `Literals` and `Resources` table describe blank nodes, literals and resources, respectively. IDs used throughout are numerical representations of MD5 hashes.

Morten Frederiksen has written a useful utility for streamlining interaction with the MySQL database, the RDF Query Rewriter [70], which rewrites an RDQL query into a SQL query compatible with the database schema used by the Redland MySQL storage module.

A few Redland/MySQL utilities have also been made available by Morten Frederiksen, allowing direct manipulation of the MySQL database from the command line. The utilities are

redland-mysql-optimize This script ensures that the indices on the tables are up to date.

redland-mysql-clean This script is perhaps the most important for ensuring a well maintained database. It ensures that there are no duplicate statements in the Statements table, and that there are no extraneous records in the Literals, Bnodes or Resources tables.

redland-mysql-drop-model As the name suggests, this is a simple script which allows the deletion of a single model from the database.

librdfutil [69] (also by Morten Frederiksen) is a library written in PHP to enhance the utility of the PHP bindings. It also contains a number of useful functions for dealing with the MySQL module, and is discussed further in the following section.

The PHP language bindings for Redland

The PHP bindings for Redland have proven to be the most problematic area during the development of Clique. As mentioned already, Redland is written in C, using object oriented principles. Whilst C does not directly support object oriented design, a number of methods for achieving a similar effect have emerged. The PHP bindings are generated by the Simplified Wrapper and Interface Generator (SWIG) [15]. A number of patches to the default SWIG output are required in order to make the return of `NULL` pointers or empty strings feasible. A number of places in which these were not completely done were discovered, leading to functions which returned incorrect values, although they were of the expected return type.

The advantages of open source software were evident when bugs in Redland were encountered. From well constructed problem reports, the Redland author was fast to respond, and quickly (within hours) implemented the fixes and e-mailed updated source code. The bindings generated by SWIG do not take into account the object orientation designed into the Redland C library, and provided by PHP, and so a set of plain wrapper functions for the C API are provided. This causes code which is not as elegant as it could be. Another problem with this approach is that objects

are not automatically deleted from memory, and a call to Redland has to be made to destroy and delete objects from memory. Because, in PHP, memory is managed for the programmer, this proved to be a steep learning curve.

After getting to grips with the API, and the need to do memory management, development was fairly fast.

5.4 Design

The design of Clique slowly evolved, although a few key core principles were held throughout. The main principle was that Clique should be created in simple, modular fashion. A central library was written to interface with Redland, much of which was borrowed from librdfutil later in the development process.

The site was broken into functional components, the following being the most important to core functionality:

- Index and home page
- Edit personal detail
- Contact manipulation (send invitations, add friend)
- Invitation acceptance and signup

Some other components within Clique were the documentation (privacy policy and help documentation), the authentication and authorisation framework, and a rough implementation of a photo gallery.

Each of these components formed a single PHP file, with functionality shared amongst the different variations of that component. Also important in the design was the use of Apache's path info [8], a feature which allows URIs to be constructed with part of the file path portion of the URI occurring after the filename. For example, if a file `eg.php` existed in the document root of the server at `example.com`, a URL of the form `http://example.com/eg.php/test/pathinfo` would be valid, serving the content (or executing the PHP, in the case of PHP) of `eg.php`. In order to remove the `.php` from the path names, content negotiation [62], using the Multiviews feature of the Apache Web server [9] was employed. This design allowed the

creation of a URL space which would not change even if the implementation was changed in the future, allowing permanent links, and therefore user convenience, and other possible benefits (such as search engine visibility).

5.5 Implementation

Extracting information from remote sources

A utility to extract a user's buddy information from ICQ [96] was created, in the hopes of making an easier user experience. The utility was written in Perl, because of the already available `Net::OSCAR` CPAN module, which implements the OSCAR protocol. OSCAR is the protocol used by both AIM and ICQ, although some specific features are required for ICQ support. The ICQ support in `Net::OSCAR` was not complete, although the required functionality was relatively simple to implement, and was contributed back to the author of the `Net::OSCAR` module.

The process of implementing the services was not straightforward. Some of the difficulties involved included:

- Only 50 buddies' information could be retrieved in a 30 minute time period
- Unreliable networking and slow protocol often lead to timeouts
- The user was signed off any other signed-on sessions because ICQ did not allow multiple logins

The difficulties in implementing a robust feature to gather ICQ data dissuaded the author from attempting to interface with any other proprietary, or difficult to access, services.

Performance

The central library of Clique caused some early performance concerns. Libraries in PHP are simply included into the current file using the `include()` or `require()` constructs, and bringing in a large library for each and every request would be a costly operation, as the PHP in the library is parsed every time a page is loaded. PHP caching technologies such as turck-mmcache [154] or eaccelerator [59, 156], which allow PHP to cache the bytecode generated after compiling the script, proved useful in reducing the effect of these problems.

Queries

At the time of writing Clique, SPARQL had not yet been formulated, and was thus not available in Redland. RDQL, a predecessor to SPARQL in many ways, was used as the query language for queries which could not be solved by simple queries using partial statements. Some operations, however, were not possible, or were overly complex when using RDQL, and so a direct SQL connection to the underlying database was used. This has proved particularly useful when generating the Friends of Friends list, and when rewriting sections of the database (this is done when a user accepts an invitation).

RDF as a programming tool

Programming using RDF constructs natively proved to be useful in cases where properties which could have multiple instances per object were used, but was tedious for simple queries, such as person's names (to be used as labels). It was also useful from an extensibility perspective, as the addition of a new property did not entail editing a database schema. Creating an object oriented mapping for interesting properties may have reduced this programming complexity, however it would remove some generality from the implementation.

librdfutil, mentioned above, was obtained late into the development process, but it provided a number of ideas and functions which were useful in the completion, and speeding up, of the development process, and the refinement of existing code. librdfutil is written in PHP, and provides a thin, but very patchy layer over Redland. Functionality provided included:

- Dealing with streams as PHP arrays
- Direct interface with MySQL database
- Simplified creation of Redland objects (eg: nodes, statements)
- Simplified serialisation

From experience with other rapid development frameworks such as Ruby on Rails and Django based on relational databases, the author feels that although using RDF as a programming tool is acceptable, the frameworks based on relational databases are more mature, and provide a better programming environment.

Creating the social network

In order to form a closed network, in which all users were linked in some way, it was decided to make Clique an invite-only site. The biggest problem this presented was seeding the database. The author created a list of friends, with email addresses, and invited those users. 61 invites were initially sent out, with the author finally sending 105 invitations. 64 (61%) of these invitations were accepted. In total 218 invitations were sent out, with 119 (55%) accepted. The user other than the author who sent the most invitations sent 25, of which only 10 were accepted.

Figure 5.2 shows the number of invitations, the number of accepted invitations (ie, members) and the number of `foaf:knows` relationships represented on the system, over the first 10 weeks of its existence. The sharp increases in the number of invitations were the batches of invitations sent out by the author, and it can be seen that the take-up was more gradual than the invitations, which is to be expected since invitations aren't accepted instantly. Other invitations can be seen after the first batch of invitations, leading to the smooth curve, but otherwise not many invitations were made. The number of `foaf:knows` relationships can be seen to increase mainly in accordance with the number of accepted invitations, indicating that it seems users used the system when first invited, but were not active on it after being invited.

Although it seemed that the author's friends were enthusiastic about supporting the project insofar as they joined the site, take up beyond that point was disappointing. Lurking bugs and the lack of a "killer feature" are suspected to be reasons for a lack of support.

5.6 User interfaces to RDF data

There are a few examples of rich and novel interfaces to RDF data have been developed by Semantic Web researchers, allowing the user to interact with the set of data. One excellent example of these interfaces is foafnaut [112], an SVG based FOAF browser by Jim Ley which focuses on allowing the user to browse a social network of people. Foafnaut's biggest limitation is that it only presents a small set of personal information about each person, because it stresses instead the display of relationships between people.

Considering FOAF data, a common display requirement is the profile of one particular person, with the resulting display being similar to a personal homepage. A user would therefore find it interesting to know as much detail as possible about that person, and a minimal amount of detail about the people that they know. Properties of people that are known by the person who is the

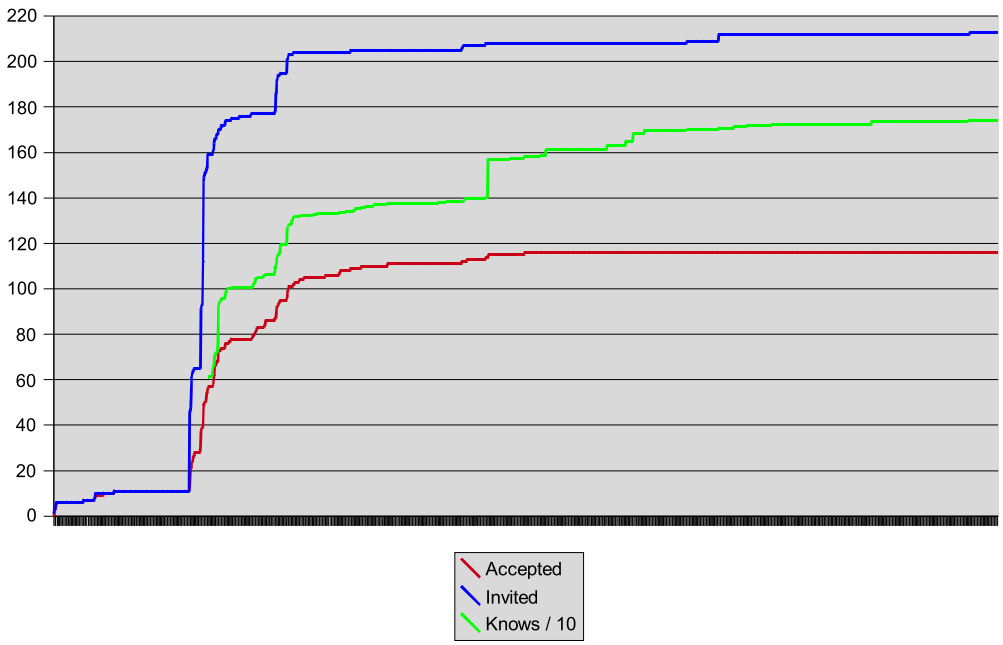

Figure 5.2: Take up of Clique over the first 10 weeks

primary topic of the data being presented may also need to be presented differently to the same property about the person who is the primary topic. For example, the name of the primary person in the document will appear as a heading, while the names of people that they know should be shown as links to their document.

Another important aspect is that labels which are useful to machines or developers may not be useful or important to users. Consider the Person class in FOAF, which has the label "Person": a useful label for a machine or developer sorting through large amounts of data, but probably useless to a user browsing a website containing FOAF data. The user will probably be able to make the assumption that the entity about which they are reading is a person from other data on the page, and the labelling is thus implicit. To people, a much more useful label for an instance of a class of type "Person" may be the person's first or full names, nickname or email address, in a rough descending order of utility. It is therefore not always possible to rely on the information encoded in the vocabulary in which the data is expressed to provide information required for displaying data to the user, but must rather create special display methods for each type.

The Extensible Stylesheet Language (XSL) [3] is well suited to simple transformations for displaying RDF/XML documents to the user. It is designed as a language for transforming XML, but has limitations in that it does not have any means of analysing the semantics of the input document, as it operates on a purely syntactic level.

An alternative (and probably more common) way to presenting RDF data is by using a fully featured programming language and an RDF toolkit[4]. Using a fully featured query toolkit makes taking advantage of the semantic richness of the RDF data possible in a manner which is impossible with XSL.

A number of possibilities and problems with using XSL to display RDF to the user will be discussed, as well as a pure programmatic means which takes advantage of the tree based nature of a large class of RDF models, as presented in [45].

5.6.1 Presenting RDF using XSL

XML has a tree structure, and the XML serialisation of RDF, RDF/XML, provides a tree based serialisation for RDF. This tree maps better to many of the more common display idioms than a graph, as a graph-like display method may require complex graphical interfaces (like foafnaut,

[4]For example: Redland [17], Jena [119, 92], 4suite [68], RDFLib [106], or RAP [130]

```
<foaf:Person rdf:nodeID="bNode1">
   <rdfs:seeAlso rdf:resource="http://russell.rucus.net/foaf"/>
   <foaf:mbox_sha1sum>6549e8d26d4669a08d0e97cc05f3b78de9b3deb4</foaf:mbox_sha1sum>
   <foaf:name>Russell Cloran</foaf:name>
</foaf:Person>
<rdf:Description rdf:nodeID="bNode2">
   <rdf:type rdf:resource="http://xmlns.com/foaf/0.1/Person"/>
   <rdfs:seeAlso rdf:resource="http://lair.moria.org/foaf.rdf"/>
   <foaf:mbox_sha1sum>37ecf3fb9ff5d19a227e097ab206adf1b2457d57</foaf:mbox_sha1sum>
   <foaf:name>Barry Irwin</foaf:name>
</rdf:Description>
<foaf:Image rdf:about="http://russell.rucus.net/blogdata/phoblog/MoreGrad2k4.jpg">
   <foaf:depicts rdf:nodeID="bNode1"/>
   <foaf:depicts rdf:nodeID="bNode2"/>
</foaf:Image>
<foaf:Image rdf:about="http://russell.rucus.net/blogdata/phoblog/BeerTower.jpg">
   <foaf:depicts>
     <foaf:Person>
       <foaf:name>John Oxley</foaf:name>
     </foaf:Person>
   </foaf:depicts>
</foaf:Image>
```

Figure 5.3: An RDF/XML fragment from a photo gallery, showing a number of different serialisations of similar data.

described above) which are not suited towards displaying large amounts of textual information. XSL provides a simple, custom means of displaying this data to the user.

An advantage of writing XSL stylesheets for RDF/XML documents is that the data may be displayed directly in modern Web browsers, without any server side processing, thereby decreasing load on the server. Delivering RDF/XML to the user agent also allows the client to receive the full benefit of the semantics expressed in the RDF data, while enabling a simple presentation to the user. Yet another advantage of sending the data as RDF/XML to the user is that it is possible to minimise the amount of data transferred, if the stylesheet is cached on the client side. Data need only occur once in the document, and data which should be displayed multiple times for the user can be referenced from other parts of the document. Figure 5.3 provides an example of this, in that the details of a person who appears in more than one photograph in the document need only have their details listed once. The blank node `bNode1` is referenced from the first `foaf:Image` instance, and may also be referenced in the same way from other `foaf:Image` instances.

RDF/XML documents must be served by Web servers with the `application/rdf+xml` MIME type according to [18]. In the testing of the major browsers currently in use (Internet Explorer 6, Firefox 1.0 and Mozilla 1.8) it was found that all of the browsers tested reported that

they could not deal with that MIME type, and asked the user whether she would like to download the file. Documents were therefore incorrectly served as `text/xml`, a MIME type which all of the browsers treated as XML, which they transformed using the XSL stylesheet.

The other major XML style language, cascading stylesheets (CSS) [30], has been considered for presenting RDF/XML to the user. Considering the RDF/XML fragment shown in Figure 5.3, we see that a number of different `foaf:Image` nodes refer to the same `foaf:Person` node. CSS (the current version is 2.1, and that is the version tested with) does not allow content from other portions of the document to be inserted into arbitrary places, and therefore the name of a person defined at an arbitrary place in the document cannot be used, for example, as the caption for more than one image.

The most common type of transformation of RDF using XSL is likely to be a transformation from RDF/XML into HTML or XHTML (referred to in combination as (X)HTML). (X)HTML is a well defined and widely used format for displaying documents on the Web, and it will therefore be easy for a Web developer to write an XSL stylesheet which transforms RDF/XML to (X)HTML. CSS also has a number of special cases for (X)HTML, which makes it easier to style (X)HTML than other XML languages using CSS.

As is often the case when dealing with RDF/XML, issues arise because of the flexibility of the serialisation. RDF/XML allows most statements to be represented in a number of different ways, and this flexibility can lead to difficulties when writing XML Path Language (XPath) [43] expressions for RDF/XML documents. As XPath expressions are not dependent on the order of sub-nodes, ordering of descendant elements is not a problem. XPath uses the expanded namespace value rather than its QName [32] to match nodes in the document. Differences in namespace declarations are also therefore not a problem.

RDF/XML serialisations allow a document to reference statements that occur in other parts of the document, or to nest tags to an arbitrary depth. This flexibility presents a problem when writing XSL transforms for XML/RDF, as the complexity of XPath expressions increases for each additional type of nesting or reference to another part of the document that code is written for.

One of the other problems that arise is that of resource typing. Resources can be given a type using the `rdf:type` property. These types are analogous to class types in object oriented programming, and can have descendant types which inherit all the properties of their parent. In RDF/XML these can be serialised in two ways, either like any other property as a sub-element of the resource description or as a typed node. Typed nodes are nodes which do not use a generic

RDF/XML tag, but use XML namespaces to construct a tag which refers to the resource itself [18]. The differences between the two are demonstrated in Figure 5.3. The node with `nodeID bNode1` is a typed node of type `foaf:Person`, and the node with `nodeID bNode2` is not a typed node, but is also of type `foaf:Person` (the first child node encodes this).

This problem also highlights another problem with using XSL to transform RDF/XML – there are a lot of semantic data encoded in the document (and in the vocabularies used by the document) that cannot be taken advantage of by processors acting directly on the RDF/XML. Take for example a vocabulary written to extend FOAF which includes a type ComputerScientist as a subclass of FOAF's Person type. A simple XSL stylesheet expecting a foaf:Person node will not be able to handle the ComputerScientist node. Semantically rich RDF processors and query languages will be able to read the new vocabulary and find out that a ComputerScientist is a Person too, thus any queries for Person types will return all ComputerScientist nodes.

Many of these problems can be fixed by partially pre-processing the RDF document using a fully featured RDF library, and serialising the document in a known subset of RDF/XML. Pre-processing would perform operations such as outputting RDF/XML of a known and consistent subset of RDF/XML, as well as the insertion of statements of semantic value that are otherwise implied by the source document. If necessary, portions of the referenced vocabularies may also be included. Pre-processing is especially applicable when serving RDF/XML documents to users on the Web, as the pre-processing need only occur once on the server side, the document specifies the stylesheet being used, and it does not require any extra work from the user agent. It can simplify the style sheet required, easing development and maintenance and also reducing the size the stylesheet (which may be important if users are likely to have a low bandwidth connection to the Internet).

Using an XSL stylesheet which considered many of the possible ways of serialising RDF/XML, the RDF/XML shown in Figure 5.3 was successfully transformed into into XHTML, displayed using Firefox 1.0 in Figure 5.4.

5.6.2 Presenting RDF programatically

A programmatic process can be easier to construct and maintain than an XSL transform of an RDF/XML document, because it is easier take into account the semantics of the RDF by using an RDF library or toolkit. XSL does not offer access to rich features of high level programming languages such as Python or C#, and is therefore less flexible than approaches to displaying RDF which use languages like these.

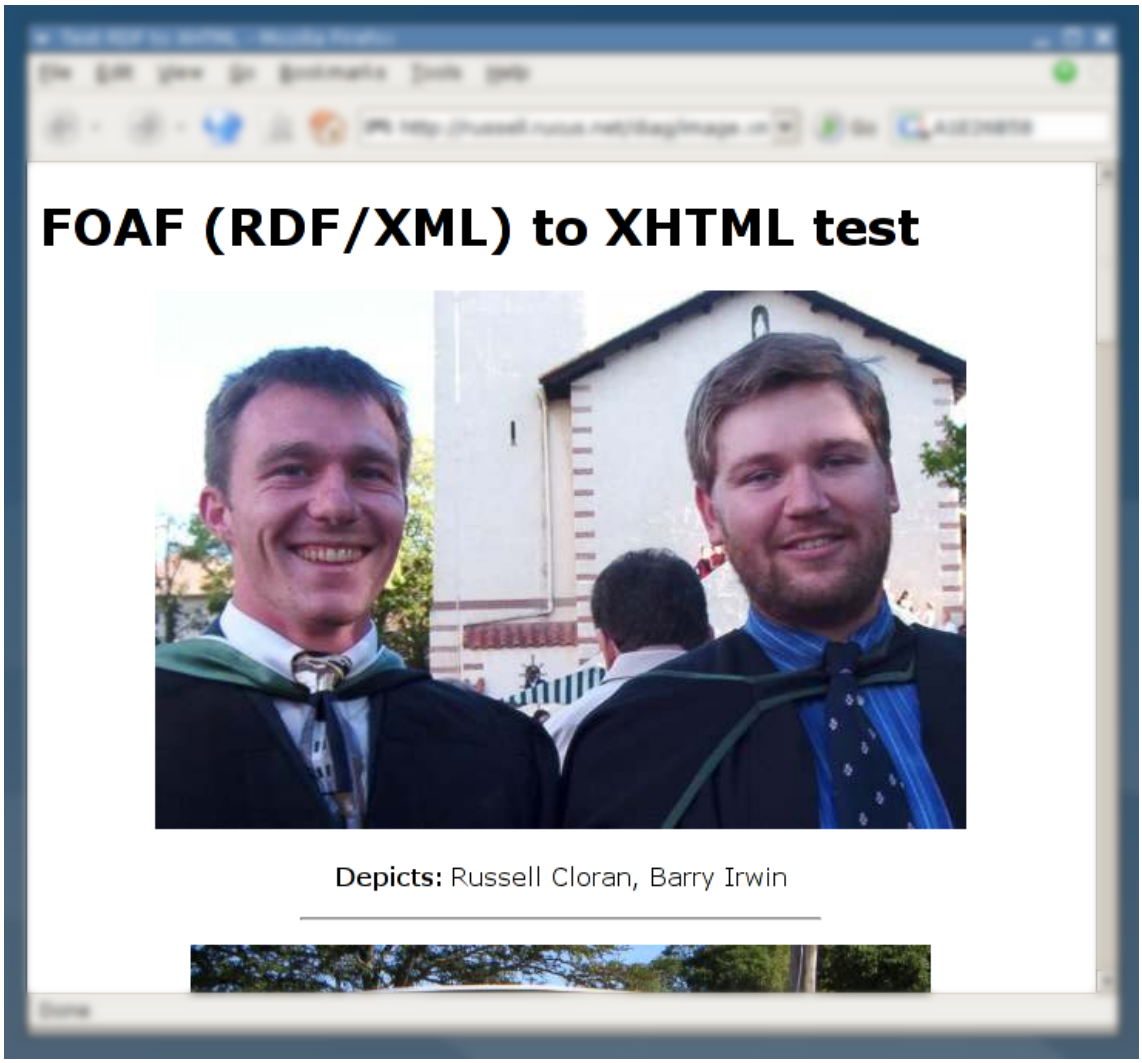

Figure 5.4: RDF/XML (from Figure 5.3) transformed into XHTML and displayed

A further advantage of using a standard programming language to display RDF is that it is possible to render the RDF using graphical toolkits and APIs such as GTK, Qt or Windows Forms, often essential for creating a rich user interface on the desktop. Interfaces such as these can also often be much richer than a Web interface, as interactivity can be increased, and richer interface elements can be used.

A number of RDF frameworks or libraries exist which provide features such as different query mechanisms, storage frameworks and even automatic statement generation based on semantics included in the vocabulary. When developing a rich Semantic Web application, the use of such semantic tools may be important, especially with a large and complicated data set. Frameworks such as these will also allow the selection of a subset of the data to be sent to the user, which may be advantageous if only a small portion of a large data set is needed.

A tree based method for rendering RDF

In the experiments, this method of rendering RDF to HTML was used to form the core of a user interface on two websites. The technique was implemented twice in PHP, once using RAP and the other using the Redland. The method is simple, and requires the invocation of a method specific to each property which the resource being rendered has. Each of these property methods then renders any necessary labels and the value of the property. The value of a property may be another resource, which is usually rendered by calling that resource's rendering function.

It was found that it was important to be able to know the context (position in the tree) of the node or property currently being rendered, as it provided some clue as to how it should be rendered. In the FOAF example, people who are not primary topics of the document should not be described in great detail (therefore not all properties about them need be presented. To this end passing context to functions in two ways was considered. The first and most basic would pass the context as a string, which could be matched with regular expressions in order to decide which version of the property or class rendering function to use. The second possible method would pass a tree structure, and use syntax similar to XPath to query the tree. It may be possible to simply pass a complete DOM tree and use XPath against that, with the advantage that an existing XPath implementation could be used. Using this method it would also be able to show sibling context, which may be valuable in deciding how to render some user interfaces.

Many property rendering functions will be performing the same task – printing a label and then either printing their literal value directly, or calling a function to print the value (whether it is a literal or a resource). A model which allows general and specialised versions of the function

seems to be appropriate for this task. The effectiveness of this may be demonstrated by the simple example of the function which prints the values of each node (required to implement the basic set of functionality needed). The function which prints a value might have two main specialisations: a resource printer and a literal printer. The resource printer could be specialised into two types, one for full URI's and another for blank nodes. A full URI may be rendered as a link to the resource, where blank nodes may be rendered simply as a label denoting the type of the resource (if it is specified). Specialisations of either, for example for a foaf:Person class, may print no information at all, but simply rely on the property rendering functions to output information such as the name and person's image based on the context information that is passed to them. Another example of a specialisation of a resource might be the foaf:Image type, which would not output a link to the resource, but instead embed the image directly into the output. It should be borne in mind that fake classes which are not part of the original ontology could be created by interface designers in order to adapt the class hierarchy to the hierarchy required for display.

As a core concept of object oriented design, generalisation and specialisation features are provided by many object oriented languages, and it would be beneficial to take advantage of these features. The ontologies created using RDF class and property specialisations can often be simplified to a taxonomy, allowing an easy mapping to the class hierarchy of object oriented languages.

One of the advantages mentioned about serving RDF/XML to the user, and applying a stylesheet on the client side was that full RDF could be served to the user, and that they would gain the full value of the content, including all semantics. A programmatic transform to HTML would probably have to occur on the server side, and thus the value of the semantics encoded in the RDF would be lost. RDF could be embedded within the HTML, but this would increase the size of the generated file, perhaps prohibitively.

5.6.3 Web based user interfaces for editing RDF

As introduced in Section 5.2 tools for creating RDF should allow the user to fully express ideas which she wishes to, but should not allow her to create invalid data or documents. This presents a challenge for data models as rich as RDF, and was one of the goals of FOAFbuilder[5], a small site built using PHP and RAP, with extensive use of JavaScript.

One of the features of FOAFbuilder, shown in Figure 5.5, is the ability to import data from

[5] `http://russell.rucus.net/2004/foafbuilder/`

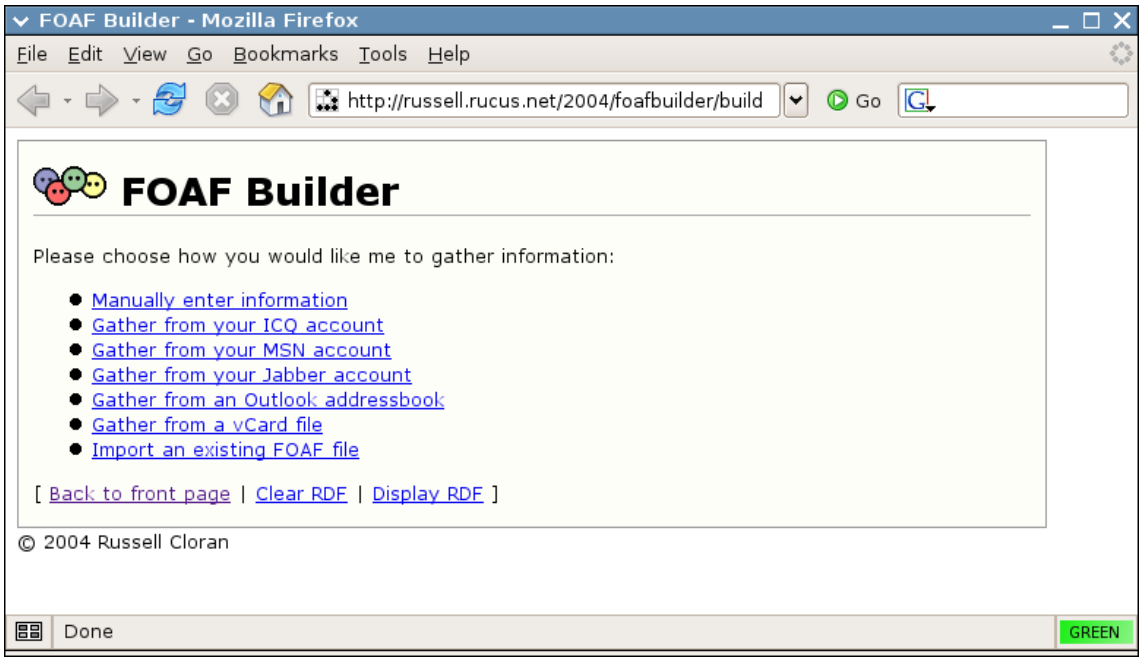

Figure 5.5: FOAFbuilder main screen

various data sources. Although not all interfaces were fully implemented, thorough testing was done using the ICQ import, and the issues surrounding translating the semantics present in the ICQ data to RDF semantics were explored. In general, conversion between ICQ properties to RDF properties was found to be easy, and because buddies were each queried separately and the buddy which properties related to was implicit in the response, the mapping to the RDF model was simple.

Figure 5.6 shows another two important features of FOAFbuilder. Firstly, the tabbed interface, made possible with JavaScript. This page was designed using degradable HTML techniques. In other words, browsers which did not support JavaScript would, instead, be presented with a number of sections on a single page, rather than a tabbed interface. Secondly, the "Nickname" field demonstrates the implementation of form fields which allow multiple instances of a property to be defined for the person. Unfortunately, there was no way of implementing this functionality for users without JavaScript, so users using browsers without JavaScript would only be able to enter zero or one instances of properties of this nature.

Figure 5.7 shows the final component of FOAFbuilder, which allows users to edit buddy lists and buddy information. Again, this component would not have been possible without JavaScript, and

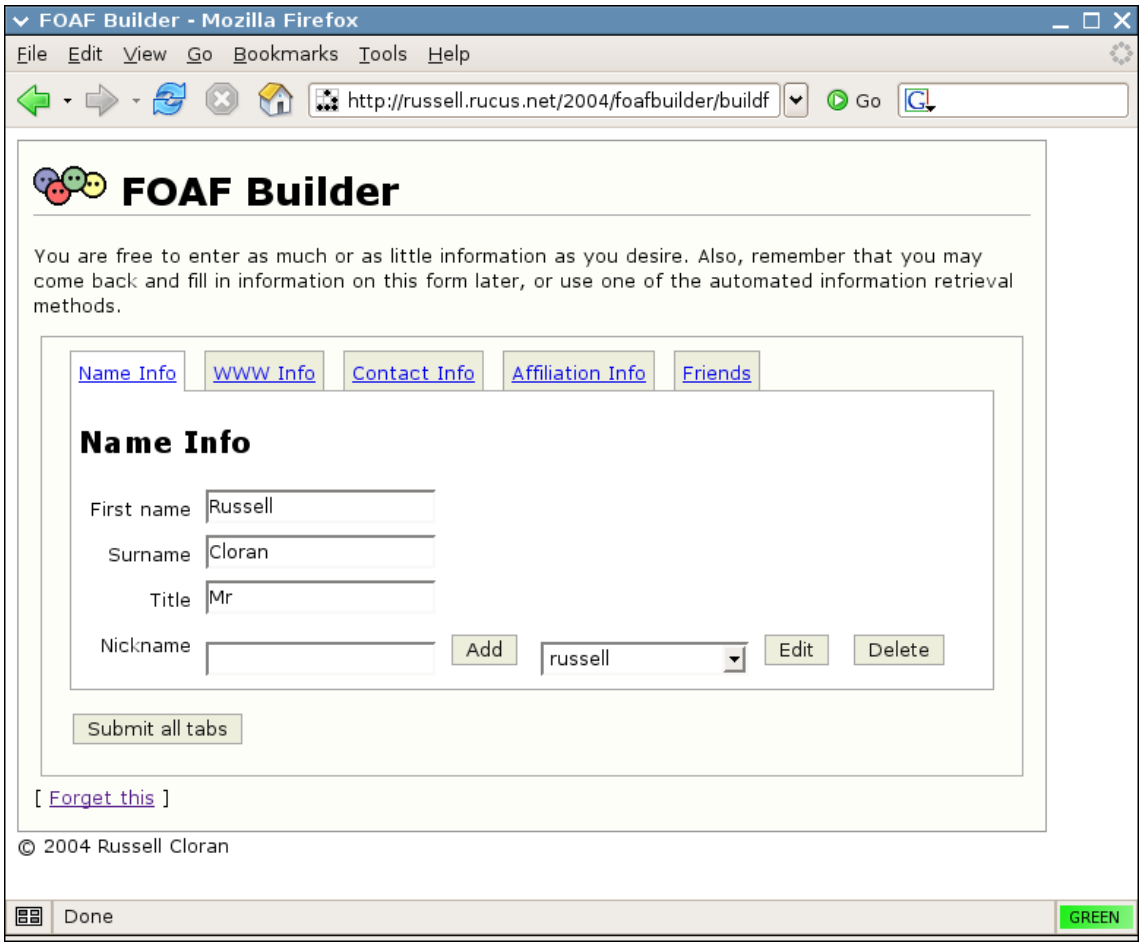

Figure 5.6: FOAFbuilder's tabbed manual editing interface

so is not available to users using a browser without JavaScript.

FOAFbuilder has two main components, the server side and the client side. As discussed, these were implemented in PHP and JavaScript, respectively. Upon completion of the form, a user may submit the entire form back to the server. Server side software then merges the information with information which already exists. This architecture was chosen to allow import of data which was not supported on the client side, but which would exist when the final RDF/XML was created. It did, however, cause a number of problems. The initial problem was that of deleting information from the server side which the user had deleted on the client side. Since a representation of RDF was transmitted, and not change information, obtaining this was a case of ensuring that for every property type submitted, the server side data was cleared, and then the data posted from the client re-added.

The second problem was that of speed. As RAP is implemented purely in PHP, large operations were considerably slower than hoped. Because a full smushing algorithm was run across the data set, speed was important. Refining the smushing algorithm reduced the run time to a reasonable time for the data sets tested.

Since our implementation, the term "AJAX"[6] [158] has been popularised to describe ways of updating a web page without reloading the entire page from the web server. These techniques use HTTP requests to the server which return XML, which is then manipulated in JavaScript. The use of such techniques may have been solutions or improvements to both of these problems.

5.7 Summary

Clique is a social networking website built using the Redland RDF application framework, and PHP. In the development of Clique, two RDF toolkits were used: RAP and Redland. Redland was found to be faster and more feature complete, but RAP easier to use in PHP. Redland had other advantages, in that the same data would be accessible from Redland bindings in other languages.

Clique did not see a large acceptance beyond the first tier of invitations, the author's direct friends. The data gathered was thus not useful in testing of trust modelling. The lessons learnt in its construction were, however, invaluable in furthering our understanding of RDF.

A number of other important lessons were learned. One of the biggest was the construction of user interfaces for presenting and editing RDF information. A programmatic means of presen-

[6] Asynchronous JavaScript And XML

Figure 5.7: FOAFbuilder's buddy editing interface

tation was used in Clique, and an XSLT means was also investigated. The programmatic means was found to be more flexible, but the XSLT valuable in that semantics were transferred to the user agent. In creating interfaces for editing RDF, we were careful to allow users the full expressions allowed by FOAF and RDF, without allowing them to create invalid data for publication on the Web.

The following chapter shows how data was finally gathered from the Semantic Web, and details the construction of a scutter and smusher.

Chapter 6

Propagating trust

For trust on the Semantic Web to be successfully used, it is necessary that there is some way of propagating this trust around a network of agents. It is not feasible for every user of the Web to evaluate the trust that she should place in each information source on the Web.

6.1 Gathering and merging data

In order to perform experiments about trust, some data is required. A popular application of RDF is Friend of a Friend (FOAF) [37], an RDF vocabulary which allows description of some personal information, as well as relationships between people. FOAF only provides a `knows` property, defined as [37]:

> A person known by this person (indicating some level of reciprocated interaction between the parties)

This simple definition can be supplemented by other vocabularies, such as the relationship vocabulary [51], which defines each of its properties to be sub-properties of the `foaf:knows` property. The popularity of FOAF means that there is a large amount of FOAF data available on the Semantic Web. Large sites, such as LiveJournal [63] provide user profile information as FOAF, meaning that it is not limited to technically minded people. Interesting aggregations have emerged, such as foafnaut [112] and LJNet [14], indicating a strong level of interest in FOAF. Other research on trust and the Semantic Web has also been conducted using FOAF as a basis

[74, 75, 76, 77, 78, 79]. For these reasons, FOAF seemed to be a sensible basis for research into trust.

Because FOAF data is distributed across the Web, a scutter is needed to download the information to a central server for it to become useful. Again, because of its distributed nature, the same person might be partially described in a number of different places. A smusher allows the information to be joined using uniquely identifying properties found in the data.

6.1.1 Scutter

A tool to gather data from the Semantic Web is generally known as a "scutter", and is analogous to the World Wide Web "spiders". The code for the scutter discussed in this section is available in C.

Whereas a spider for the Web would analyse HTML and search for anchor, image or object elements in order to find other documents on the Web, a scutter would analyse RDF (specifically RDF/XML) documents in order to find statements in the document which have the `rdfs:seeAlso` property as the predicate. RDFS [36] says that the object of these statements "may provide additional information about [the subject of the statement]". RDFS does not impose any restriction on the format of the data retrieved at the URI, however the data is usually another RDF document. Further sources to attempt to gather RDF data are:

- Namespace URIs: RDF namespace URIs often (but this is not mandatory) have an RDF vocabulary, expressed in RDF using RDFS (and perhaps OWL), as the representation of that URI when retrieved [114]. Although this differs from other XML, which may expect an XML Schema document at the namespace URI (although this is neither required, nor an explicit goal of XML Namespace) [32], it is convenient for an RDF scutter, as this is another potential source of RDF data. This potential source of RDF data extends to property and type URIs which are a part of the vocabulary, but might be at a different URI[1].

- Described resources: Although RDF is generally a metadata format, and is used to describe resources which are not RDF (instead, they are usually documents or images), some URIs which are described may reference an RDF resource. A scutter designed to obtain the

[1] FOAF, for example, has a namespace URI of `http://xmlns.com/foaf/0.1/`, and so all class and property URIs are different resources, not document fragments of the same resource.

greatest amount of information, whilst incurring large cost, may include this as another source of URIs.

The collection of RDF documents can be retrieved by a breadth-first traversal algorithm. URIs which have been downloaded and parsed (or have failed to be parsed) are recorded as having been seen, and any links which they contain which have not previously been seen are added to the tail of the queue of documents to retrieve.

The Standard for Robot Exclusion [104], whilst not perfect [105], is a widely implemented standard which specifies resources on a website which can be indexed, or which may be off-limits. The standard is simple, and works by a crawler attempting to retrieve a document called `robots.txt` from the root directory of a web server before retrieving any other content from the web server. The file format is simple, and consists of a number of records. Records consist of a number of fields. Each field is represented by a line in a text file, with records separated by a blank line.

The first field of each record is a `User-Agent` field, and the remainder of which are `Disallow` fields. The user-agent field is designed to be a pattern which matches the User-Agent header sent by HTTP clients.

Specific user-agents may thereby be disallowed from accessing content which is unsuitable for that user-agent. All automated web crawlers should respect this standard, and thus the restrictions put in place for automated web crawlers by the webmaster. The standard is, however, optional, and there is therefore no way of enforcing compliance by web crawlers. The scutter implemented for this work did *not* implement the Standard for Robot Exclusion.

Well behaved robots on the web should transmit an accurate "`User-Agent`" HTTP header when retrieving web pages. This allows web servers to monitor visitors, and record statistics on visitors. The scutter implemented used the default User-Agent header for urllib, a Python component for downloading Web resources.

Web crawlers need a starting point, or a number of starting points. Having a number of starting points is useful in cases where there are sets of documents which are not interrelated. The scutter implemented by the author only allowed for one starting point, however, the starting point was an RDF/XML file which contained many `rdfs:seeAlso` references to a number of other files. The starting points used are listed in Appendix D, and were obtained largely in two ways. Firstly, a few known resources were used, such as FOAF files of friends, FOAF data exported by Clique (discussed in Chapter 5,

The scutter implemented by the author also ignored failed retrievals. Certain error conditions should, in a fully implemented scutter, cause the resource to be queued for a retried retrieval at a later time. Only after a number of failed attempts should the URI be discarded.

Documents retrieved by the scutter were stored in a simple file system hierarchy, with the host name at the top of the hierarchy. URIs which ended in a trailing slash were taken to imply directory indexes, and so a directory of that name was created, and the file stored as a file named `index.rdf` in that directory. The slash character was encoded into the filename by creating a sub-directory, which is the common interpretation of URLs in most cases. Non-alphanumeric characters were encoded on the file system as their hexadecimal value, prepended by an underscore character, to ease typing of filenames if it was required. This is not a perfect way to store the documents, with the biggest problem being the possible clash in names with the `index.rdf` document which was created. In the author's implementation, if a file already existed on the file system, it was simply overwritten. Other possibilities exist, such as storing all data in a database rather than on a standard file system, or using a database with a hash lookup to a filename on a standard file system. The simplest implementation was chosen for its obvious nature, and to simplify deployment of the scutter on a number of machines.

Ideally, documents should be stored with metadata received in the HTTP headers. This is especially important if the crawl is not a once off event, so that bandwidth can be saved by using the caching mechanisms provided by HTTP, such as those provided in the `E-Tag` and `Last-Modified` headers, which require the headers' values to be saved.

6.1.2 Smusher

RDF data gathered from a number sources can be merged using a tool known as a smusher. RDF data can be thought of as a set of statements, and so can be naively joined by creating a set union of those statements. OWL [16] introduces some semantics about data which influence the effectiveness of this naive operation. Of importance are the `owl:FunctionalProperty` and `owl:InverseFunctionalProperty` property types, and the `owl:SameAs` property.

An `owl:FunctionalProperty` is a property for which the value is a function of the subject. This implies that for any single subject, there can only be one value for that property. The `foaf:gender` property, which specifies a `foaf:Agent`'s gender, is an example of an `owl:FunctionalProperty` – an agent may only have one gender.

An `owl:InverseFunctionalProperty` is a property type for which the subject of the

statement is a function of object of the statement. The `foaf:mbox` property, which specifies a `foaf:Agent`'s mailbox URI (an agent's email address) is an example of a property which is an `owl:InverseFunctionalProperty` – for any `foaf:mbox` may only have one `foaf:Agent` associated with it (the assumption of the creators of FOAF being that mailboxes are not shared).

An RDF vocabulary or ontology specification is often provided in RDF/XML, and the RDF for that vocabulary will specify properties of property types, such as `owl:InverseFunctionalProperty` or `owl:FunctionalProperty`. A smusher applied to a data store can either have a hard coded set of properties which should be treated as functional or inverse-functional properties, or it may obtain this information from the data store which it is smushing. In the case of obtaining this information from the data store, the data store should have the RDF specifications of the vocabularies which are used to describe the data in the store loaded into the store itself.

A smusher can infer that two resources identifiers refer to the same resource if both have an `owl:InverseFunctionalProperty` with the same value. A resource with two `owl:FunctionalProperty` properties with different values also implies that the objects of the statement are equivalent resources. A smusher has two options when two resources are detected to be the same. It may either rewrite the URI of one of the resources to match the other, or it may mark the two as identical by using the `owl:SameAs` property.

The results of a smusher are limited by the quality of the data. Some missing information may occur for a number of reasons, including temporarily unavailable resources or resources which have been removed from the Web. More complex may be the issue of accurate data which may not be linked to other data, because no shared inverse functional properties or functional properties exist.

A heuristic approach to smushing may also be taken, especially if human confirmation of the results is possible, or if the data is of a known subset. For example, if it is known that a list of people are all friends of one person, then it can be more safely assumed that two resources with the same value for a "name" property are the same person, than if the smushing was occurring on a global scale. Heuristically, this may be more generally coded as two resources which share many properties, and are both values of the same property of a number of other resources. For example, if two resources have the same `foaf:name`, `foaf:gender`, and `foaf:birthdate` and a number of other resources have `foaf:knows` arcs to both, then it is likely that they are the same resource.

Occurrences	Type URI
486675	http://xmlns.com/foaf/0.1/Person
23243	http://xmlns.com/foaf/0.1/Image
18444	http://xmlns.com/foaf/0.1/PersonalProfileDocument
11376	http://www.w3.org/2003/01/geo/wgs84_pos#Point
8961	http://www.aktors.org/ontology/portal#Researcher-In-Academia
8628	http://xmlns.com/foaf/0.1/Document
5439	http://purl.org/rss/1.0/item
3439	http://www.w3.org/2000/10/swap/pim/exif#ColorPicture
2459	http://purl.org/dc/dcmitype/Image
2335	http://xmlns.com/wordnet/1.6/Tree
1734	http://xmlns.com/wordnet/1.6/Clouds
1611	http://web.resource.org/cc/Agent
1561	http://xmlns.com/wordnet/1.6/Lens-1
1471	http://www.daml.org/2001/10/html/airport-ont#Airport
1302	http://xmlns.com/wordnet/1.6/Sky
1121	http://xmlns.com/wordnet/1.6/Road
1077	http://xmlns.com/wordnet/1.6/Building
1044	http://purl.org/rss/1.0/channel

Table 6.1: Number of occurrences of various data types in scuttered data

6.1.3 Results

The experiment did not return as large a dataset as expected. The data set is, however, large enough to be interesting. 50692 unique documents were successfully retrieved, consisting of 4211759 statements. After smushing, 4194978 statements remained. 486675 nodes with the `rdf:type` of `foaf:Person` were identified, with another 79 nodes with the incorrect `rdf:type` of `foaf:person` (note the incorrect case of the initial "p") also identified.

The size of the dataset contrasts with the approximately 1.3 million profiles available on LiveJournal in 2004 [107] (all available as FOAF data), and the limited set of 21506 profiles which mentioned the top 500 interests on LiveJournal in 2004 [136]. Although some data was obtained from LiveJournal, it is clear that the network obtained was isolated.

The few incorrect statements in the gathered data indicate that writing RDF by hand can be error-prone, even amongst an early-adopter audience who, it would seem, will be more technically proficient than the majority of the population. This, again, brings into consideration our

statements in Chapter 5 that authors of tools which generate RDF have a responsibility to generate correct RDF.

The large number of `foaf:Person` nodes in relation to the number of documents is interesting (on average, 9.6 nodes per document). FOAF encourages users to publish information about themselves, and not other people. This seems to indicate one or a combination of the following possibilities:

- The smusher was not able to correctly identify nodes which were the same as another, due to incomplete or invalid data
- Many people publish information about friends who do not have FOAF data of their own
- Documents referenced with an `rdfs:seeAlso` link were not retrievable

The second option seems to be the most intuitively acceptable. Consider, for example, the document which was used as a scutterplan – the author's own FOAF file. The document contains 123 nodes with an rdf:type of `foaf:Person`, but only 9 `rdf:seeAlso` references.

The disparity between the number of `foaf:Person` nodes and the number of documents highlights the crux of this work: to what extent should a person (or system) be trusted when their statements concern another person? The data seems to indicate that it is considered acceptable (by the existing Semantic Web community) to publish some information on others.

6.2 Summary

The development of a Semantic Web scutter allowed us to obtain a large amount of data from the Semantic Web. We showed that the vast majority of the data was FOAF data, suggesting that FOAF documents are not widely linked with RDF data of other types. The application of a smusher to the data did not drastically reduce the data-set size, although no analysis on further smushing possibilities was done.

The following chapter discusses means of strongly linking a creator with content, and the problems that the relatively free format of RDF/XML creates with this.

Chapter 7

Asserting authorship

A social network which allows people to assert their belief of other's trustworthiness is of limited worth unless the authorship of a piece of information can be validated. Cryptographic systems such as PGP and XML Digital Signature allow the authors of documents to digitally sign the documents that they have created, providing checks for authenticity and integrity.

7.1 Digital Signature

Digital signature uses asymmetric cryptography to provide authentication, integrity and non-repudiation checks to arbitrary data [159].

Simply put, asymmetric cryptography is cryptography in which a different public and private key exist for each agent. Data encrypted with the public key may only be decrypted with the private key, and data encrypted with the private key may only be encrypted with the public key.

Digital signature uses this by allowing the author of content to encrypt the content (or a hash of that content) with her private key. Anybody with her public key may then decrypt that, and check it aginst the plain-text (unencrypted) content. If the decrypted content (or hash thereof) matches the plain-text content (or its hash), then we can be sure, to the degree offered by the cryptographic tools used, that some agent with access to the author's private key signed the content.

Generally no semantics are attached with a signature. In the context of electronic mail, it might be assumed that a signature assures authorship. In the context of digital key or certificate verification, it might be assumed that a signature implies an acknowledgement of the accuracy of the data contained in the certificate.

```
   <edd> foafbot, edd's name
   <foafbot> edd's name is ``Edd Dumbill'', according to
Dan Brickley,
Anon35, Niel Bornstein, Jo Walsh, Dave Beckett, Edd
Dumbill, Matt
Biddulph, Paul Ford
```

Figure 7.1: An example IRC session querying foafbot

Therefore, if and only if an author keeps their private key private, digital signature provides checks for authenticity and conversely non-repudiation of the signature of the data. In a context where authorship is assumed or explicitly stated through some outside mechanism, digital signature can thus be used to check authorship of content.

7.2 Existing systems

It is interesting to note that the use of cryptography together with RDF in the field has been limited to one application which the author has seen. The Web of Trust (WOT) vocabulary [33] is an RDF vocabulary which uses Pretty Good Privacy (PGP) [167] as a cryptographic basis. WOT allows PGP signing events to be described, including the association of a signature with a document, a person with a signing event, and a signing event with a signature. Example usage of the WOT vocabulary, with specific reference to FOAF is provided in [58].

The WOT vocabulary has a certain appeal to the "hacker" community, who are often early adopters of new technology. It is available now, and uses the well known and popular technology, PGP. However, only one use of the WOT vocabulary, Edd Dumbill's FOAFBot[1] has been seen by the author. The FOAFBot is an "IRC Community Support Agent", and is designed to provide answers to users of an Internet Relay Chat (IRC) [129] channel (chatroom). The FOAFBot also attributes its sources by name, if there is cryptographic proof that the source made the statement.

Figure 7.1 shows an example query session with foafbot. Notice "Anon35", a person who has made a statement that there is a person with the IRC nickname "edd" who has the full name "Edd Dumbill", but has not backed this up with a PGP signature. Other users have signed the document in which the statements supporting the fact occurs, and so are attributed by name.

[1] http://usefulinc.com/foaf/foafbot

The author speculates that the very low usage rate of cryptography together with RDF occurs for two main reasons. Firstly, the Semantic Web at this point in time is reminiscent of the early days of the Internet. Most of the users are researchers who are more interested in experimental results rather than securing their communications. Secondly, the marginal gain from providing signed RDF data is low, as not many agents can process either XML Digital Signature or statements as they occur in the WOT vocabulary. The author again speculate that a wider uptake of Semantic Web technologies will ensure that cryptography is used where required.

7.2.1 XML Digital Signature

XML Digital Signature [60] is a W3C recommendation, and Internet draft standard (RFC), which provides a way to sign arbitrary digital information, and represent this signature in XML. XML digital signature was designed to allow any data which can be referenced as a URL to be signed, and it thus allows the signing of XML sub-documents through document fragment identifiers.

The process of generating an XML signature follows the steps described in [60]:

1. Generate the reference
   (a) Apply the transforms, as determined by the application, to the data object
   (b) Calculate the digest value over the resulting data object
   (c) Create a `Reference` element, including the (optional) identification of the data object, any (optional) transform elements, the digest algorithm and the `DigestValue`.
2. Generate the signature
   (a) Create `SignedInfo` element with `SignatureMethod`, `CanonicalizationMethod` and `Reference`(s)
   (b) Canonicalize and then calculate the `SignatureValue` over `SignedInfo` based on algorithms specified in `SignedInfo`
   (c) Construct the `Signature` element that includes `SignedInfo`, `Object`(s) (if desired, encoding may be different than that used for signing), `KeyInfo` (if required), and `SignatureValue`

We see here that XML signature provides opportunity for the application (XML document containing RDF data in this case) to create a canonical version of its data prior to signature calculation.

A useful side effect of the fact that XML signature allows XML fragments to be signed, and the representation of signatures within an XML document, is that a signature and the signed content may be transmitted in the same XML document. This is especially useful with RDF/XML, as it is conceivable that an author wishes to sign multiple chunks of data, or even individual statements, separately, so that they can be separated, and later verified separately.

7.2.2 XML Canonicalisation

Something which is canonical has been "stated or used in the most basic and straightforwardly applicable manner" [160] or is "of admitted authority, excellence, or supremacy" or is "authoritative; orthodox, accepted; standard" [149]. XML Canonicalisation "refers to the process of applying the XML canonicalisation method to an XML document or document subset" [31].

XML Canonicalisation allows two XML documents to be compared for identity in a bit-wise manner, after they have both been transformed into the canonical form. This is the crucial first step in allowing any type of digital signatures (PGP is another example which is distinct from XML digital signatures) to be applied across XML documents, as a signing operation requires that the initial input and the input to be checked are identical.

The basic process of XML canonicalisation can be broken down into 4 steps, as described in [31]:

1. Normalize line feeds
2. Normalize attribute values
3. Replace CDATA sections with their character content
4. Resolve character and parsed entity references

Notably, the process of creating a canonical XML document does not explicitly mention the form of the document used as input. A particular XML language may have multiple ways of encoding a certain meaning, one of which should be chosen as the "canonical document form". It should be assumed that canonical XML should be generated from a canonical document form, if more than one exists.

7.2.3 Signed RDF

Canonical XML only addresses the semantics introduced by XML and not the application which is using it. RDF/XML introduces semantics which mean that a number of different XML documents can represent the same RDF model, due to the flexibility of the RDF/XML serialisation standard. Figure 7.2 shows an RDF graph and two possible (valid) RDF/XML serialisations for the graph. Some of the factors in RDF/XML that cause the possibility of different serialisations are:

- Typed nodes are optional (see Section 7.2.6)
- An RDF graph may be broken into a tree in a number of ways (see Section 2.3.5)
- Order is not always important (with some exceptions, such as the `Seq` property)

The consequence of this is that without some means to create a canonical RDF/XML document from any model, RDF documents cannot easily be compared for identity unless some means of a canonical serialisation is created.

7.2.4 Canonical RDF/XML

As shown above, canonicalisation of RDF/XML is not fully encompassed by XML canonicalisation. In order to create a canonical RDF/XML serialisation from an arbitrary data store, the RDF data should be interpreted as a graph. This graph needs to be mapped into an XML tree, and the tree, finally, serialised as XML. Choices in the conversion from a graph to a tree greatly affect the XML which may be produced. These various mappings can be thought of as distinctive levels.

Levels of semantics

In "Towards the Semantic Web" Broekstra, Kampman and van Hermelen [100] suggest that RDF may be queried at three levels: the syntax, structural and semantic levels. The author proposes that these basic distinctions may also be applied to the process of creating a canonical RDF document. These are discussed below in a top down fashion.

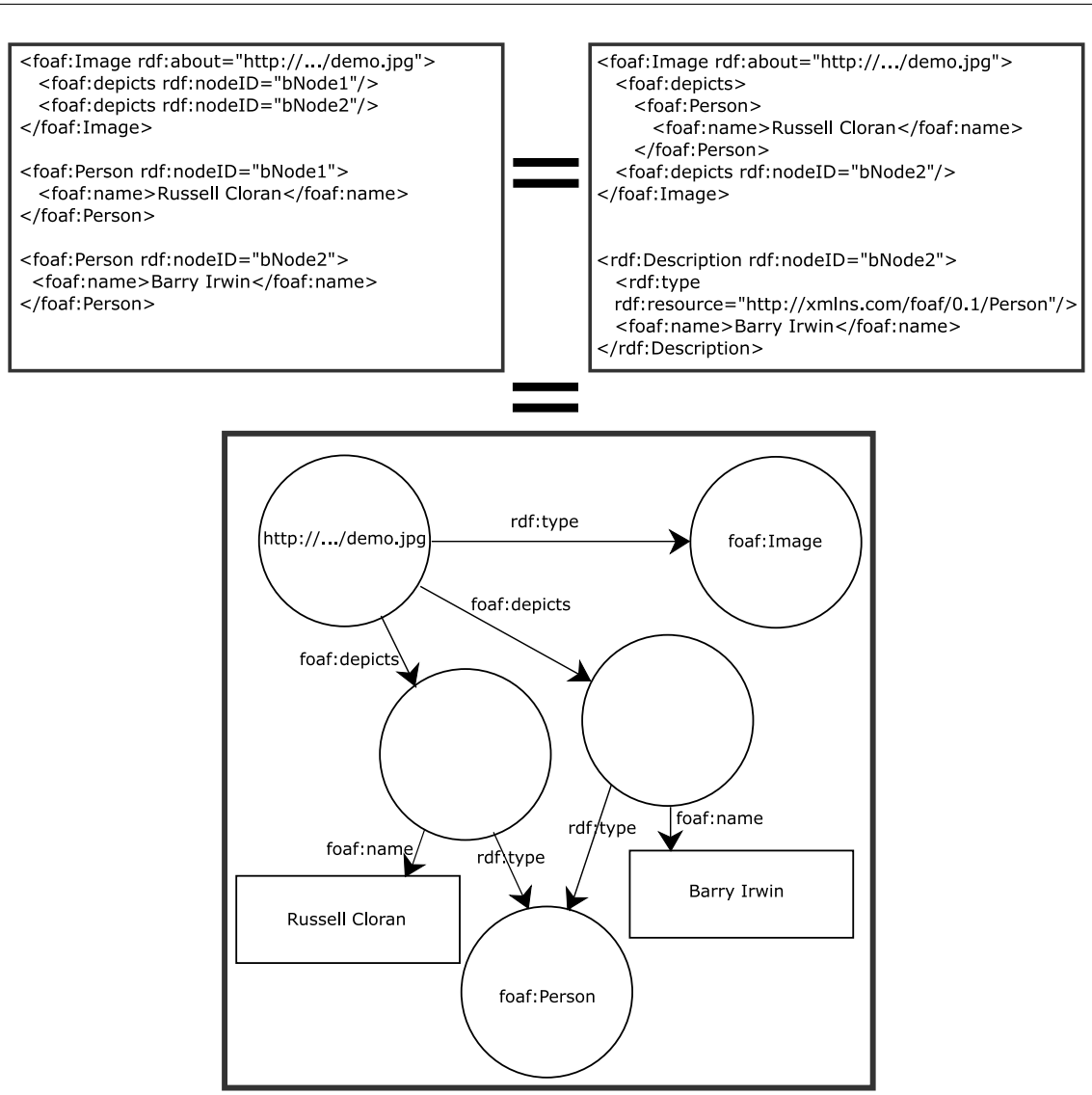

Figure 7.2: An RDF graph serialised as two different RDF/XML documents

Semantic level

The semantic level is the highest level of understanding that the application has, and the model is divorced furthest from its physical representation. At this level RDF is considered as one or more labelled graphs, with some predefined semantics. At this level, issues such as blank node identifiers are not considered, but issues such the structure of the graph are considered instead.

Two graphs are said to be isomorphic if there is a one-to-one correspondence between their vertices, and a one-to-one correspondence between the edges joining corresponding vertices. Two RDF graphs can therefore said to be equivalent if they are found to be isomorphic. Following this, testing RDF graph equality and graph isomorphism have equivalent complexity, as discussed by Carroll in [41].

Testing for graph isomorphism is a problem for which no polynomial time solution has been found, nor has it been proven that it is NP-complete [150]. Carroll [40] notes that it is therefore theoretically possible to verify the signature of an arbitrary RDF graph in polynomial time. Carroll's methods for improving on this are discussed in section 7.2.5.

At this level the RDF data is referred to as a model or a graph, as compared to the syntactic level, where the serialised format is referred to as an RDF document.

Structural level

At the structural level, RDF is considered as a set of triples. The semantics of the graph are broken down into a series of individual node-arc-node relationships. Other than the knowledge that these statements are part of the same RDF model, they are unrelated. At this level, the problem of blank node identifiers arises, as making a link from a labelled node to an unlabelled node and then from the unlabelled node to another node needs to have some sort of link. This link is created using blank node identifiers. The canonicalisation method described by Carroll in [40] operates chiefly at the structural level, although the specific implementation described relies on the N-Triples serialisation described in the RDF Test Cases [84].

Renaming of blank node identifiers and the ordering of triples for use at the syntactic level are operations which occur at the structural level. The first operation requires some knowledge of the semantic level, namely, when renaming a blank node it is necessary to rename all blank nodes with the same identifier in order to retain the semantics. This particular operation is the most difficult part of any of the processes of creating a canonical RDF model or document, and is essentially part of the process of testing for isomorphism. An example of this process is described

in [40], and discussed in Section 7.2.5. Ordering of the triples is also discussed in [40], but is a relatively straightforward process of lexically ordering the triples that make up the graph by subject, predicate and then object.

Syntactic level

At a syntactical level the serialised RDF is considered. In the case of RDF/XML, this would be the XML document that is produced by serialisation. The flexibility of the RDF/XML means that it is likely that only a subset of the RDF/XML features will be used, as is done in R3X [71], discussed in Section 7.2.6. Because the result of an RDF/XML serialisation is an XML document, the XML canonicalisation rules described in 7.2.2 should also be applied in order to produce a canonical XML document, either as part of the serialisation process, or after it.

The process of signing the document also happens at this level, since some sort of serialisation of the RDF model is required in order to sign it. Efforts such as the WOT vocabulary [33] and existing XML signature solve this problem adequately, if a canonical serialisation can be obtained.

7.2.5 Carroll's canonicalisation method

The techniques described by Carroll in [40] are designed to obtain a canonical version of an RDF graph. The N-Triples serialisation is used both for processing and the ultimate output format, as described in the paper. The method arguably operates at all three of the levels discussed above. Most of the difficult work is done at the structural level in order to test that two graphs are the same at the semantic level. The general outline of Carroll's algorithm is:

1. Replace blank nodes (in subject and object positions) with a placeholder, storing the original blank node identifier
2. Sort the triples lexically
3. From top to bottom, replace blank nodes in statement subjects with an incrementing blank node identifier, ensuring all instances of the same blank node get replaced with the same new identifier, if the statement (with placeholder blank node) is not the same as the previous or following statement (with placeholder blank node).

4. Repeat the previous step for blank nodes in statement objects

This process is deterministic for a subset of RDF models. A further "precanonicalisation" step is required to ensure that this process works for all RDF models. This step involves inserting statements which add no meaning to the document between carefully selected nodes. This precanonicalisation arguably changes some of the semantics of the graph.

These algorithms described by Carroll provide a useful step in obtaining a canonical RDF model, although a change from the N-Triples syntax is required in order to harness the power and interoperability introduced by the XML language serialisation of RDF.

7.2.6 R3X

Redland Restricted RDF/XML (R3X) is a subset of the RDF/XML serialisation created by Morten Frederiksen. There are currently two implementations of R3X: one as PHP code which output R3X directly from a stream in the Redland framework, and another as an XSL transform which takes as input an RDF/XML document which is output by the default Redland RDF/XML serialisation component. R3X is, in fact, less restricted than the default output of the Redland library's serialisation library, which only uses a very limited subset of the productions available in RDF/XML. R3X was originally designed as a means of simplifying XSL stylesheets which converted RDF to HTML for user display, and so is not entirely designed for the goal of obtaining a canonical RDF document. It is an interesting observation nonetheless, as it shows some of the key ideas which may be useful in creating a canonical RDF document.

When it is considered that R3X is a serialisation procedure, it is notable that it is a bridge between the structural and syntactic levels described in Section 7.2.4. It does not operate at one of the levels, but rather provides the transition from the higher level to the lower.

Observing the output of R3X provides some insight to the production of an RDF/XML document which Carroll's method doesn't, since Carroll's method is not concerned with RDF/XML at all.

Firstly, it is observed that grouping by subject (obtained automatically by ordering lexically sorting statements) drastically shortens the document, and improves human readability, although this should not be considered an important goal. This has a welcome side effect in that statements always occur at the "top level" of the document, eliminating nesting issues.

Secondly, XML namespaces are declared for every node produced in the R3X serialisation. Since the Canonical XML recommendation [31] does not alter namespace declarations, it is important that a canonical RDF serialisation outputs namespace information in a deterministic manner.

Lastly, the R3X output does not in any way order sub-nodes, unless they are part of an RDF `seq`. Such a result occurs naturally as a side effect of an operation similar to Carroll's algorithm, or is a trivial step which can be performed prior to serialisation.

7.3 User education

A thorough discussion on the nature of the adoption of technologies is beyond the scope of this work, but some brief thoughts on the subject are offered for completeness.

As noted at the beginning of this chapter, actual usage of cryptography to verify authorship of documents on the Web is not high. Creating an environment where cryptography is commonly used to verify an author's authenticity is a chicken and egg problem – if tools do not widely utilise digital signatures to enrich a user's experience, content authors will see little benefit in providing digitally signed content. Without digitally signed content, software which consumes RDF can not rely on digital signature being available to verify authenticity. Bootstrapping the use of digital signature in an environment like the Web seems to be contingent on early adopters publishing signed content, and producing useful tools which use this published content.

With the utility demonstrated to other organisations and implementers, the development of tools which take advantage of digital signature will be much higher than if no utility has been demonstrated. These tools are the pieces of software which will make it easy for average users to easily add strong cryptographic links to the information they publish on the Semantic Web.

7.4 Summary

This chapter has shown that asserting authorship can be achieved by digital signature. XML Digital Signature was discussed as a solution which provides features which are useful when dealing with XML documents, such as RDF/XML. These are:

1. Any data which may be referenced by a URI can be signed. XML document fragments may therefore be signed
2. Signatures are represented in XML
3. A canonicalisation algorithm for allowing bit-wise comparison of XML documents

The combination of the first two features means that signatures and data fragments may be transmitted in the same XML document, and multiple signatures and data fragments may also be transmitted in the same document.

Carroll's canonicalisation method, which is designed to be used with the N-Triples serialisation presents useful techniques for canonicalisation at the semantic and structural level. R3X, a subset of RDF/XML provides useful techniques for canonicalisation at the syntax level. A combination of the two provides a workable solution to canonical RDF/XML.

A brief thought on the take up of such technologies was also presented, with encouragement for early adopters to show the utility provided by digitally signing RDF/XML documents.

Chapter 8

Conclusion

This work has detailed the investigations of trust on the Semantic Web undertaken by the author. The goals, as stated in Chapter 1, were to:

1. Investigate and understand the Semantic Web, and how it relates to other current trends on the Web.
2. Investigate and understand the notion of trust, how it applies to the Semantic Web, and how it may be propagated.
3. Investigate suitable technologies for establishing authorship, and how these relate to the Semantic Web, for the purpose of establishing trust relationships

8.1 Summary of work

Our investigation into the Semantic Web focused on the technology which realises its goals – RDF. The RDF data model, as well as serialisations of RDF were discussed in-depth. The FOAF vocabulary was also introduced, as it formed an important part of many of the sub-projects undertaken in this work. The work focused on parts of RDF and its related technologies which were important to understanding the vocabularies which would be used later in trust related work. The work also focused on those parts of RDF which were useful in building Clique, a social networking site built using RDF technologies.

The work next moved to developing an understanding of trust, and forming a definition applicable to use when used in relation to the Semantic Web. Concepts surrounding trust were examined,

from an investigation into why and how people trust other people in day-to-day interactions, to the investigation of a number of different methods for modelling trust. Trust models and metrics for a number of different areas (distributed systems in a general sense, recommender systems and the Semantic Web) were examined. Much of the discussion on trust is not only applicable to the Semantic Web, and is useful in gaining a broader understanding of trust in relation to information security.

A number of sub-projects were tackled in order to gain a better understanding of trust and the Semantic Web. These were:

1. A case study detailing advanced, manual, checks of a website's author and credibility.
2. The development of a social networking site, Clique, based on RDF technologies.
3. The development of "scutter", or Semantic Web spider to download and process RDF data, and a "smusher" to aggregate the data.
4. Investigate means of linking a creator of RDF content with the content

The first sub-project, a case study on manual means of checking a website's credibility, gave some idea about how advanced Web users would check a website's credibility. The website in question was found to be credible, however the checks used suggested that it was not. The exercise was useful in gaining a better understanding of how people assess a website's credibility, and helped form initial ideas on how machines could do the same.

The second sub-project tackled was to develop a social networking site, named Clique. Clique was designed to be focused around Rhodes University, and almost all of the initial invitees were Rhodes students. Clique experienced some growth, but never caught on, probably due to the lack of a "killer" feature. Many social networking sites already exist, and the benefit of joining yet another social network is minimal for most people. Because of this, one of the goals of Clique (to gather data which would be used in trust investigations) was never realised to a useful extent.

Clique was, however, successful as an investigation into RDF toolkits, and the value of using RDF as a backend data model. It was found that the tools were, at the time, immature. Adapting to the tools took some time, but once the peculiarities were worked around, development was sped up. A brief comparison of RAP (the RDF API for PHP) and Redland was presented. Although RAP was easier to use because no memory management was necessary, Redland had more features, and was much faster. Overall, using the RDF data model and the Redland RDF

toolkit proved to be acceptable. Some newer Web frameworks, based on relational databases, provide rapid development opportunities which show greater opportunity for rapid, effective development than basing development on RDF toolkits.

For the third, smaller sub-project, a scutter was developed using Python and the Redland RDF toolkit. A smusher was used on the resulting database, which linked uniquely identifiable nodes together, thus providing a unified data set on which tests could be performed. A large amount of RDF data was downloaded from the Web, and a brief analysis of the data showed that it was largely FOAF data.

The final sub-project undertaken was the investigation of linking an author with the RDF content produced. In Chapter 3, it was shown that current trust metrics provide an improvement over traditional techniques in areas which involve knowledge transfer. These trust metrics rely on trust in the creator of the information. In a distributed publishing environment, such as the Web, it is necessary to be able to create a strong tie between the creator and the content. Chapter 7 presents digital signature as a solution to this.

There is some existing work in the area, although it is not thorough. The WOT vocabulary allows the description of signatures using PGP, allowing validation of a document's author. The biggest problem with this is that the validation must occur with an unchanged document. Another problem with this is its lack of standardisation and wide acceptance, something which XML Digital Signature has gained. XML Digital Signature also allows more flexibility, such as the ability to embed signed content into XML documents. XML Digital Signature provides a standard for XML canonicalisation, so that any semantically equivalent XML documents may be compared for similarity, also enabling the use of digital signatures.

XML Digital Signature focuses on the semantics introduced by XML, but leaves room for application-specific canonicalisation. No such canonicalisation exists for RDF/XML, and an examination of what exists and what is still required was presented in Section 7.2.4. The concept of canonicalisation at different levels was suggested, and requirements for canonicalisation at each level discussed. The particular problems with canonicalisation of RDF/XML were discussed. Restricted RDF/XML serialisations were discussed as a possible solution to canonicalisation at the syntax level, with particular focus on the R3X serialisation.

The goal of realising a "trusted Semantic Web", introduced in Section 3.9, has been shown to be possible. The two main areas which are needed to provide this – an accurate, attack resistant trust metric, and cryptographic signatures to strongly link a creator with content – have been shown to be possible. A standard means for signing RDF/XML documents which are published

on the Web seems to be necessary to make the second possible. Along with this, educating publishers as to the advantages of proper digital signature, as shown in Chapter 4, is important. Only with digital signature information provided by publishers will trust metrics become useful. A responsibility of creators of tools which publish information on the Semantic Web is to allow users to publish this information easily.

8.2 Further Work

In any work of this nature, a number of questions arise during the work which are outside its scope. The more significant of those that have arisen during our work are noted here.

8.2.1 Credibility assessment using human means

One of the most interesting areas of work for us is the combination of ideas on how humans can assess trust values and how machines commonly propagate trust through social networks. Humans assess the credibility of a web page based on a number of criteria, whereas we have presented only one criterion – trust in the author, possibly in a particular knowledge domain. Increasing the number of criteria on which a machine can gain a level of trust may help to compose an overall better metric. What other criteria are there on the Semantic Web?

Humans often assess credibility through elements such as perceived professionalism of the presentation and website structure. Can a computer be taught to do the same? Is RDF/XML flexible enough that a Bayesian, or similar, filter would be able to determine poor quality information from high quality information?

8.2.2 Enhanced trust metrics

We briefly discussed the use of a value for certainty or deviation, in addition to a level of trust, in Section 3.2.3. This is not common amongst current trust metrics, but it could possibly lead to a trust metric which more accurately models real life, and thus works better for many users. The main benefit would be in the modelling of dishonesty and malice more accurately – where attackers may exhibit a tendency to act in a trustworthy manner until a time most opportune for attack, and then deviate from the usual behaviour.

References

[1] ABDUL-RAHMAN, A., AND HAILES, S. A distributed trust model. In *NSPW '97: Proceedings of the 1997 workshop on New security paradigms* (New York, NY, USA, 1997), ACM Press, pp. 48–60. http://doi.acm.org/10.1145/283699.283739.

[2] ABDUL-RAHMAN, A., AND HAILES, S. Using recommendations for managing trust in distributed systems. In *IEEE Malaysia International Conference on Communication* (November 1997). http://www.cs.ucl.ac.uk/staff/F.AbdulRahman/docs/micc97.ps.

[3] ADLER, S., BERGLUND, A., CARUSO, J., DEACH, S., GROSSO, P., GUTENTAG, E., MILOWSKI, R. A., PARNELL, S., RICHMAN, J., AND ZILLES, S. Extensible stylesheet language (XSL) version 1.0. World Wide Web Consortium, Recommendation REC-xsl-20011015, Oct. 2001. http://www.w3.org/TR/2001/REC-xsl-20011015.

[4] XHTML 1.1 – module-based XHTML. World Wide Web Consortium, Recommendation REC-xhtml11-20010531, May 2001. http://www.w3.org/TR/2001/REC-xhtml11-20010531.

[5] AMAZON. Amazon.com: Online shopping for electronics, apparel, computers, books, dvds & more. http://www.amazon.com/.

[6] ANTI-PHISHING WORKING GROUP. Phishing activity trends report, Oct. 2005. http://www.antiphishing.org/reports/apwg_phishing_activity_report_oct_05.pdf.

[7] ANTI-PHISHING WORKING GROUP. Phishing activity trends report, May 2006. http://www.antiphishing.org/reports/apwg_report_May2006.pdf.

[8] APACHE SOFTWARE FOUNDATION. Apache HTTP server version 2.0 documentation: Apache core features, 2005. http://httpd.apache.org/docs/2.0/mod/core.html#acceptpathinfo.

[9] APACHE SOFTWARE FOUNDATION. Apache HTTP server version 2.0 documentation: Content negotiation, 2005. http://httpd.apache.org/docs/2.0/content-negotiation.html#multiviews.

[10] AXELROD, R. *The Evolution of Cooperation*. Basic Books, Mar. 1984.

[11] XHTML 2.0. World Wide Web Consortium, Working Draft WD-xhtml2-20060726, July 2006. http://www.w3.org/TR/2006/WD-xhtml2-20060726/.

[12] BARNESANDNOBLE.COM LLC. Barnes & Noble.com. http://www.bn.com/.

[13] BARROSO, L. A., DEAN, J., AND HÖLZLE, U. Web search for a planet: The Google cluster architecture. *IEEE Micro 23*, 2 (2003), 22–28. http://dx.doi.org/10.1109/MM.2003.1196112.

[14] BARRY, P. Ljnet: Livejournal social network browser. http://patrickbarry.com/projects/ljnet/.

[15] BEAZLEY, D. M. SWIG : An easy to use tool for integrating scripting languages with C and C++. In *4th Annual Tcl/Tk Workshop Conference* (July 1996), The USENIX Association. See also http://www.swig.org/.

[16] BECHHOFER, S., VAN HARMELEN, F., HENDLER, J. A., HORROCKS, I., MCGUINNESS, D. L., PATEL-SCHNEIDER, P. F., AND STEIN, L. A. OWL web ontology language reference. World Wide Web Consortium, Recommendation REC-owl-ref-20040210, Feb. 2004. http://www.w3.org/TR/2004/REC-owl-ref-20040210.

[17] BECKETT, D. The design and implementation of the redland RDF application framework. In *WWW '01: Proceedings of the 10th international conference on World Wide Web* (New York, NY, USA, 2001), ACM Press, pp. 449–456. http://doi.acm.org/10.1145/371920.372099.

[18] BECKETT, D. RDF/XML syntax specification (revised). World Wide Web Consortium, Recommendation REC-rdf-syntax-grammar-20040210, Feb. 2004. http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210.

[19] BECKETT, D. Turtle – Terse RDF triple language, Aug. 2005. http://www.dajobe.org/2004/01/turtle/.

[20] SPARQL query results XML format. World Wide Web Consortium, Working Draft WD-rdf-sparql-XMLres-20050801, Aug. 2005. http://www.w3.org/TR/2005/WD-rdf-sparql-XMLres-20050801.

[21] BERNERS-LEE, T. Information management: A proposal, Mar. 1989. http://www.w3.org/History/1989/proposal.html.

[22] BERNERS-LEE, T. Cool URIs don't change, 1998. http://www.w3.org/Provider/Style/URI.

[23] BERNERS-LEE, T. CWM – a general purpose data processor for the Semantic Web, Oct. 2000. http://www.w3.org/2000/10/swap/doc/cwm.html.

[24] BERNERS-LEE, T. Notation 3, Nov. 2001. http://www.w3.org/DesignIssues/Notation3.html.

[25] BERNERS-LEE, T. Semantic web status and direction, 2003. http://www.w3.org/2003/Talks/1023-iswc-tbl/all.htm.

[26] BERNERS-LEE, T., FIELDING, R., AND MASINTER, L. Uniform Resource Identifiers (URI): Generic syntax. Tech. Rep. Internet RFC 2396, IETF, 1998. http://www.ietf.org/rfc/rfc2396.txt.

[27] BERNERS-LEE, T., FISCHETTI, M., AND DERTOUZOS, M. *Weaving the Web*. HarperCollins, San Francisco, California, Sept. 1999.

[28] BERNERS-LEE, T., HENDLER, J., AND LASSILA, O. The Semantic Web. *Scientific American 284*, 5 (May 2001), 34–43. http://www.sciam.com/2001/0501issue/0501berners-lee.html.

[29] BOARD, D. U. DCMI metadata terms. Dublin Core Metadata Initiative Recommendation, June 2005. http://dublincore.org/documents/2005/06/13/dcmi-terms/.

[30] BOS, B., ÇELIK, T., HICKSON, I., AND LIE, H. W. Cascading style sheets, level 2 revision 1: CSS 2.1 specification. World Wide Web Consortium, Working Draft WD-CSS21-20050613, June 2005. http://www.w3.org/TR/2005/WD-CSS21-20050613.

[31] BOYER, J. M. Canonical XML version 1.0. World Wide Web Consortium, Recommendation REC-xml-c14n-20010315, Mar. 2001. http://www.w3.org/TR/2001/REC-xml-c14n-20010315.

[32] BRAY, T., HOLLANDER, D., AND LAYMAN, A. Namespaces in XML. World Wide Web Consortium, Recommendation REC-xml-names-19990114, Jan. 1999. http://www.w3.org/TR/1999/REC-xml-names-19990114.

[33] BRICKLEY, D. Web of Trust RDF ontology. http://xmlns.com/wot/0.1/.

[34] BRICKLEY, D. Semantic Web history: Nodes and arcs 1989-1999, Nov. 1999. http://www.w3.org/1999/11/11-WWWProposal/.

[35] BRICKLEY, D. RDF: Understanding the striped RDF/XML syntax, Aug. 2002. http://www.w3.org/2001/10/stripes/.

[36] BRICKLEY, D., AND GUHA, R. V. RDF Vocabulary Description Language 1.0: RDF Schema. World Wide Web Consortium, Recommendation REC-rdf-schema-20040210, Oct. 2004. http://www.w3.org/TR/rdf-schema-20040210/.

[37] BRICKLEY, D., AND MILLER, L. The Friend of a Friend vocabulary, July 2005. http://xmlns.com/foaf/0.1/.

[38] BRIN, S., AND PAGE, L. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems 30*, 1–7 (1998), 107–117. http://citeseer.ist.psu.edu/brin98anatomy.html.

[39] CANNY, J. Collaborative filtering with privacy. In *SP '02: Proceedings of the 2002 IEEE Symposium on Security and Privacy* (Washington, DC, USA, 2002), IEEE Computer Society, p. 45.

[40] CAROLL, J. J. Signing RDF graphs. Tech. Rep. HPL-2003-142, Hewlett-Packard, Jul 2003. http://www.hpl.hp.com/techreports/2003/HPL-2003-142.html.

[41] CARROLL, J. Matching RDF graphs. In *Proceedings of the First International Semantic Web Conference* (2002), I. Horrocks and J. Hendler, Eds., pp. 5–15.

[42] CCITT (CONSULTATIVE COMMITTEE ON INTERNATIONAL TELEGRAPHY AND TELEPHONY). The directory – authentication framework. CCITT Recommendation X.509, 1988.

[43] CLARK, J., AND DEROSE, S. J. XML Path Language (XPath) version 1.0. World Wide Web Consortium, Recommendation REC-xpath-19991116, Nov. 1999. `http://www.w3.org/TR/1999/REC-xpath-19991116`.

[44] SPARQL protocol for RDF. World Wide Web Consortium, Working Draft WD-rdf-sparql-protocol-20060125, Jan. 2006. `http://www.w3.org/TR/2006/WD-rdf-sparql-protocol-20060125/`.

[45] CLORAN, R., AND IRWIN, B. Drawing data on the semantic web: Presenting RDF to end users. In *Proceedings of the 7th Annual Conference on WWW Applications* (2005).

[46] CLORAN, R., AND IRWIN, B. Trust on the web. In *Proceedings of the fifth annual ISSA Information Security Conference* (2005).

[47] COATES, T. Cal Henderson on "How We Built Flickr"..., June 2005. `http://www.plasticbag.org/archives/2005/06/cal_henderson_on_how_we_built_flickr.shtml`.

[48] CONSUMERS UNION OF U.S., I. Consumer reports webwatch. `http://www.consumerwebwatch.org/`.

[49] CRAIGSLIST, INC. craigslist. `http://www.craigslist.com/`.

[50] DATE, C. J. *A guide to the SQL standard*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1986.

[51] DAVIS, I., AND VITIELLO, E. RELATIONSHIP: A vocabulary for describing relationships between people, 2005. `http://purl.org/vocab/relationship/`.

[52] DAWSON, F., AND HOWES, T. vCard MIME directory profile. Tech. Rep. 2426, IETF, Sept. 1998. `http://www.ietf.org/rfc/rfc2426.txt`.

[53] DAWSON, F., AND STENERSON, D. Internet calendaring and scheduling core object specification (icalendar). Internet RFC 2445, Nov. 1998. `http://www.ietf.org/rfc/rfc2445.txt`.

[54] DCMI USAGE BOARD. DCMI grammatical principles, Nov. 2003. http://dublincore.org/usage/documents/2003/11/18/principles/.

[55] DIERKS, T., AND ALLEN, C. The TLS protocol. Tech. Rep. Internet RFC 2246, IETF, Jan. 1999. http://www.ietf.org/rfc/rfc2246.txt.

[56] DOUBLECLICK INC. DoubleClick: Digital advertising. http://www.doubleclick.com/.

[57] DUERST, M., AND SUIGNARD, M. Internationalized resource identifiers (IRIs). Tech. Rep. Internet RFC 3987, IETF, Jan. 2005. http://www.ietf.org/rfc/rfc3987.txt.

[58] DUMBILL, E. PGP signing FOAF files, 2002. http://usefulinc.com/foaf/signingFoafFiles.

[59] EACCELERATOR PROJECT. eAccelerator, 2006. http://eaccelerator.net/.

[60] EASTLAKE, D., REAGLE, J., AND SOLO, D. XML-Signature syntax and processing. Tech. Rep. Internet RFC 3275, IETF, Mar. 2002. http://www.ietf.org/rfc/rfc3275.txt.

[61] ENGELS, R., AND LECH, T. Generating ontologies for the Semantic Web: OntoBuilder. In *Towards the Semantic Web: Ontology-Driven Knowledge Management*, J. Davies, D. Fensel, and F. van Harmelen, Eds. John Wiley and Sons, 2003.

[62] FIELDING, R., GETTYS, J., MOGUL, J., FRYSTYK, H., MASINTER, L., PEACH, P., AND BERNERS-LEE, T. Hypertext Transfer Protocol – HTTP/1.1. Tech. Rep. Internet RFC 2616, IETF, June 1999. http://www.ietf.org/rfc/rfc2616.txt.

[63] FITZPATRICK, B. LiveJournal. http://www.livejournal.com/.

[64] FOGG, B., MARSHALL, J., LARAKI, O., OSIPOVICH, A., VARMA, C., FANG, N., PAUL, J., RANGNEKAR, A., SHON, J., SWANI, P., AND TREINEN, M. What makes a web site credible? A report on a large quantitative study. In *Proceedings of ACM CHI 2001 Conference on Human Factors in Computing Systems* (New York, NY, USA, 2001), vol. 1, ACM Press, pp. 61–68. http://captology.stanford.edu/pdf/p61-fogg.pdf.

[65] FOGG, B., SOOHOO, C., DANIELSON, D., MARABLE, L., STANFORD, J., AND TAUBER, E. R. How do people evaluate a web site's credibility?, Oct. 2002. http://www.consumerwebwatch.org/dynamic/web-credibility-report-evaluate.cfm.

[66] FOGG, B., AND TSENG, S. The elements of computer credibility. In *Proceedings of ACM CHI 99 Conference on Human Factors in Computing Systems* (New York, NY, USA, 1999), vol. 1, ACM Press, pp. 80–87. http://captology.stanford.edu/pdf/p80-fogg.pdf.

[67] FOGG, B. J. Stanford guidelines for web credibility, May 2002. http://www.webcredibility.org/guidelines/.

[68] FOURTHOUGHT, INC. 4suite: an open-source platform for xml and rdf processing. http://4suite.org/.

[69] FREDERIKSEN, M. librdfutil, Sept. 2004. http://www.wasab.dk/morten/blog/archives/2004/09/19/librdfutil.

[70] FREDERIKSEN, M. RDF query rewriter, Apr. 2004. http://www.wasab.dk/morten/2004/04/rewrite/.

[71] FREDERIKSEN, M. Transforming RDF/XML with XSLT, May 2004. http://www.wasab.dk/morten/blog/archives/2004/05/30/transforming-rdfxml-with-xslt.

[72] FRIEDL, J. E. F. *Mastering Regular Expressions*. O'Reilly & Associates, Sebastopol, California, Jan. 1997.

[73] GAMBETTA, D. Can we trust trust? In *Trust: Making and Breaking Cooperative Relations (electronic edition)*, D. Gambetta, Ed. Department of Sociology, University of Oxford, 2000, ch. 13, pp. 213–237. http://www.sociology.ox.ac.uk/papers/gambetta213-237.pdf.

[74] GOLBECK, J. The trust ontology. http://trust.mindswap.org/trustOnt.shtml.

[75] GOLBECK, J. *Computing and Applying Trust in Web-Based Social Networks*. PhD thesis, University of Maryland, College Park, 2005.

[76] GOLBECK, J., AND HENDLER, J. Accuracy of metrics for inferring trust and reputation in Semantic Web-based social networks. In *Proceedings of EKAW'04* (2004). http://www.mindswap.org/papers/GolbeckEKAW04.pdf.

[77] GOLBECK, J., AND HENDLER, J. Inferring reputation on the Semantic Web. In *Proceedings of the Thirteenth International World Wide Web Conference* (2004). http://www.mindswap.org/papers/GolbeckWWW04.pdf.

[78] GOLBECK, J., AND HENDLER, J. Reputation network analysis for email filtering. In *Proceedings of the First Conference on Email and Anti-Spam* (2004). http://www.mindswap.org/papers/Email04.pdf.

[79] GOLBECK, J., HENDLER, J., AND PARSIA, B. Trust networks on the Semantic Web. In *Proceedings of Cooperative Intelligent Agents 2003* (Helsinki, Finland, 2003), University of Maryland. http://www.mindswap.org/papers/CIA03.pdf.

[80] GOLDBERG, D., NICHOLS, D., OKI, B. M., AND TERRY, D. Using collaborative filtering to weave an information tapestry. *Communications of the ACM 35*, 12 (1992), 61–70. http://doi.acm.org/10.1145/138859.138867.

[81] GOOGLE. Google AdSense. http://www.google.com/adsense.

[82] GOOGLE. Google help : Search features : Definitions. http://www.google.com/help/features.html#definitions.

[83] GOOGLE. Google maps. http://maps.google.com/.

[84] GRANT, J., AND BECKETT, D. RDF test cases. World Wide Web Consortium, Recommendation REC-rdf-testcases-20040210, Feb. 2004. http://www.w3.org/TR/2004/REC-rdf-testcases-20040210.

[85] HAMMERSLEY, B. What is Atom?, Oct. 2005. http://www.xml.com/pub/a/2005/10/26/what-is-atom.html.

[86] HARRENSTIEN, K., STAHL, M., AND FEINLER, E. NICNAME/WHOIS. Tech. Rep. Internet RFC 954, IETF, Oct. 1985. http://www.ietf.org/rfc/rfc954.txt.

[87] HAUSTEIN, S., AND PLEUMANN, J. Easing participation in the Semantic Web. In *International Workshop on the Semantic Web at WWW2002* (2002). http://citeseer.ist.psu.edu/haustein02easing.html.

[88] HAUSTEIN, S., AND PLEUMANN, J. Is participation in the semantic web too difficult? In *ISWC '02: Proceedings of the First International Semantic Web Conference on The Semantic Web* (London, UK, 2002), Springer-Verlag, pp. 448–453. http://citeseer.ist.psu.edu/haustein02is.html.

[89] HAYES, P. RDF semantics. World Wide Web Consortium, Recommendation REC-rdf-mt-20040210, Feb. 2004. http://www.w3.org/TR/2004/REC-rdf-mt-20040210.

[90] HAZAËL-MASSIEUX, D., AND CONNOLLY, D. Gleaning Resource Descriptions from Dialects of Languages (GRDDL). World Wide Web Consortium, W3C Team Submission SUBM-grddl-20050516, May 2005. http://www.w3.org/TeamSubmission/2005/SUBM-grddl-20050516.

[91] HERMAN, I. Semantic web activity statement, May 2006. http://www.w3.org/2001/sw/Activity.

[92] HEWLETT-PACKARD DEVELOPMENT COMPANY, LP. Jena – a semantic web framework for Java. http://jena.sourceforge.net/.

[93] HOUSLEY, R., FORD, W., POLK, W., AND SOLO, D. Internet X.509 Public Key Infrastructure Certificate and CRL Profile. RFC 2459 (Proposed Standard), Jan. 1999. http://www.ietf.org/rfc/rfc2459.txt.

[94] HOWES, T., SMITH, M., AND DAWSON, F. A MIME Content-Type for directory information. Tech. Rep. Internet RFC 2425, IETF, Sept. 1998. http://www.ietf.org/rfc/rfc2425.txt.

[95] IANNELLA, R. Representing vCard objects in RDF/XML. World Wide Web Consortium, Note NOTE-vcard-rdf-20010222, Feb. 2001. http://www.w3.org/TR/2001/NOTE-vcard-rdf-20010222.

[96] ICQ INC. ICQ. http://www.icq.com/.

[97] INTERNATIONAL TELECOMMUNICATION UNION. The directory – authentication framework. ITU-T Recommendation X.509, Nov. 1993.

[98] INTERNATIONAL TELECOMMUNICATION UNION. The directory – authentication framework. ITU-T Recommendation X.509, 1997.

[99] INTERNATIONAL TELECOMMUNICATION UNION. The directory – authentication framework. ITU-T Recommendation X.509, 2000.

[100] JEEN BROEKSTRA, A. K., AND VAN HERMELEN, F. Sesame: A generic architecture for storing and querying RDF and RDF Schema. In *Towards the Semantic Web: Ontology-Driven Knowledge Management*, J. Davies, D. Fensel, and F. van Harmelen, Eds. John Wiley and Sons, 2003.

[101] JØSANG, A. The right type of trust for distributed systems. In *Proceedings of the 1996 New Security Paradigms Workshop* (1996), C. Meadows, Ed., ACM. `http://sky.fit.qut.edu.au/~josang/papers/Jos1996-NSPW.pdf`.

[102] KLYNE, G., AND CARROLL, J. J. Resource Description Framework (RDF): Concepts and abstract syntax. World Wide Web Consortium, Recommendation REC-rdf-concepts-20040210, Feb. 2004. `http://www.w3.org/TR/2004/REC-rdf-concepts-20040210`.

[103] KOKKELINK, S., AND SCHWAZL, R. Expressing qualified Dublin Core in RDF/XML. Dublin Core Metadata Initiative Recommendation, May 2002. `http://dublincore.org/documents/2002/05/15/dcq-rdf-xml/`.

[104] KOSTER, M. A standard for robot exclusion, 1994. `http://www.robotstxt.org/wc/norobots.html`.

[105] KOSTER, M. Evaluation of the standard for robots exclusion, 1996. `http://www.robotstxt.org/wc/eval.html`.

[106] KRECH, D. RDFLib. `http://rdflib.net/`.

[107] KUMAR, R., NOVAK, J., RAGHAVAN, P., AND TOMKINS, A. Structure and evolution of blogspace. *Commun. ACM 47*, 12 (2004), 35–39.

[108] LAST.FM LTD. Last.fm – the social music revolution. `http://www.last.fm/`.

[109] LEVIEN, R. Attack-resistant trust metrics for public key certification. In *Proceedings of the 7th USENIX Security Symposium* (Jan. 1998), pp. 229–242. `https://www.usenix.org/publications/library/proceedings/sec98/full_papers/levien/levien_html/levien.html`.

[110] LEVIEN, R. Advogato's trust metric, 2000. http://www.advogato.org/trust-metric.html.

[111] LEVIEN, R., AND AIKEN, A. An attack-resistant, scaleable name service, 2000. http://www.levien.com/fc.ps.

[112] LEY, J. foafnaut FOAF browser. http://www.foafnaut.org/.

[113] LUCAS, G. Star Wars: Episode V – The Empire Strikes Back. Film, 1980. See also: http://www.imdb.com/title/tt0080684/.

[114] MANOLA, F., AND MILLER, E. RDF primer. World Wide Web Consortium, Recommendation REC-rdf-primer-20040210, Feb. 2004. http://www.w3.org/TR/rdf-primer/.

[115] MAPQUEST INC. MapQuest.com: Maps, directions and more. http://www.mapquest.com/.

[116] MARON, M. the world as a blog. http://www.brainoff.com/geoblog/.

[117] MASSA, P., AND AVESANI, P. Trust-aware collaborative filtering for recommender systems. In *CoopIS/DOA/ODBASE* (2004), pp. 492–508. http://dblp.uni-trier.de/db/conf/coopis/coopis2004-1.html#MassaA04.

[118] MASSA, P., AND BHATTACHARJEE, B. Using Trust in Recommender Systems: An Experimental Analysis. In *Proceedings of iTrust 2004* (2004). http://doi.acm.org/10.1145/223904.223929.

[119] MCBRIDE, B. Jena: Implementing the rdf model and syntax specification. In *The Second International Workshop on the Semantic Web* (May 2000). http://www.hpl.hp.com/personal/bwm/papers/20001221-paper/.

[120] MICROFORMATS WIKI. Semantic XHTML, 2006. http://microformats.org/wiki/semantic-xhtml.

[121] MICROFORMATS WIKI. semantic-xhtml-design-principles, 2006. http://microformats.org/wiki/semantic-xhtml-design-principles.

[122] MICROSOFT CORPORATION. MSN maps & directions. http://maps.msn.com/.

[123] MILLER, E., AND BRICKLEY, D. Expressing simple dublin core in RDF/XML. Dublin Core Metadata Initiative Recommendation, July 2002. http://dublincore.org/documents/2002/07/31/dcmes-xml/.

[124] MILLER, L., AND BRICKLEY, D. The friend of a friend (foaf) project. http://www.foaf-project.org/.

[125] MILLS, D. L. Internet name domains. Internet RFC 799, Sept. 1981. http://www.ietf.org/rfc/rfc799.txt.

[126] MOCKAPETRIS, P. Domain names — concepts and facilities. Internet RFC 1034 (Standard), Nov. 1987. http://www.ietf.org/rfc/rfc1034.txt.

[127] MOCKAPETRIS, P. Domain names — implementation and specification. Internet RFC 1035 (Standard), Nov. 1987. http://www.ietf.org/rfc/rfc1035.txt.

[128] O'DONOVAN, J., AND SMYTH, B. Trust in recommender systems. In *Proceedings of the 10th international conference on Intelligent user interfaces* (2004), J. Riedl, A. Jameson, D. Billsus, and T. Lau, Eds., pp. 167–174. http://doi.acm.org/10.1145/1040830.1040870.

[129] OIKARINEN, J., AND REED, D. Internet relay chat protocol. Tech. Rep. Internet RFC 1459, IETF, May 1993. http://www.ietf.org/rfc/rfc1459.txt.

[130] OLDAKOWSKI, R., BIZER, C., AND WESTPHAL, D. RAP: RDF API for PHP. In *1st International Workshop on Interpreted Languages* (2004).

[131] O'REILLY, T. Web 2.0: Compact definition?, Oct. 2005. http://radar.oreilly.com/archives/2005/10/web_20_compact_definition.html.

[132] O'REILLY, T. What is web 2.0, Sept. 2005. http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html.

[133] O'REILLY NETWORK WIKI. Welcome to foo camp, 2005. http://wiki.oreillynet.com/foocamp05/.

[134] PAGE, L., BRIN, S., MOTWANI, R., AND WINOGRAD, T. The PageRank citation ranking: Bringing order to the web. Tech. Rep. SIDL-WP-1999-0120, Stanford University, Nov. 1999. http://dbpubs.stanford.edu/pub/1999-66.

[135] PALMER, S. B. RDF in HTML: Approaches, May 2002. http://infomesh.net/2002/rdfinhtml/.

[136] PAOLILLO, J. C., MERCURE, S., AND WRIGHT, E. The social semantics of livejournal foaf: Structure and change from 2004 to 2005. In *Proceedings of the 1st Workshop on Semantic Network Analysis at the ISWC 2005 Conference* (Galway, Ireland, November 2005), G. Stumme, B. Hoser, C. Schmitz, and H. Alani, Eds., pp. 69 – 80.

[137] PEMBERTON, S. XHTML 1.0: The Extensible HyperText Markup Language. World Wide Web Consortium, Recommendation REC-xhtml1-20000126, Jan. 2000. http://www.w3.org/TR/2000/REC-xhtml1-20000126.

[138] PILGRIM, M. What is RSS?, Dec. 2002. http://www.xml.com/pub/a/2002/12/18/dive-into-xml.html.

[139] POSTEL, J. B. Domain name system structure and delegation. Internet RFC 1591 (Informational), Mar. 1994. http://www.ietf.org/rfc/rfc1591.txt.

[140] POWELL, A. Expressing Dublin Core in HTML/XHTML meta and link elements. Dublin Core Metadata Initiative Recommendation, Nov. 2003. http://dublincore.org/documents/2003/11/30/dcq-html/.

[141] POWELL, A., AND JOHNSTON, P. Guidelines for implementing Dublin Core in XML. Dublin Core Metadata Initiative Recommendation, Apr. 2003. http://dublincore.org/documents/2003/04/02/dc-xml-guidelines/.

[142] PRUD'HOMMEAUX, E., AND SEABORNE, A. SPARQL query language for RDF. World Wide Web Consortium, Working Draft WD-rdf-sparql-query-20051123, Nov. 2005. http://www.w3.org/TR/2005/WD-rdf-sparql-query-20051123.

[143] RADEMACHER, P. HousingMaps. http://www.housingmaps.com/.

[144] RAGGETT, D., LE HORS, A., AND JACOBS, I. HTML 4.01 specification. World Wide Web Consortium, Recommendation REC-html401-19991224, Dec. 1999. http://www.w3.org/TR/1999/REC-html401-19991224.

[145] REAGLE, J. M. Finding Bacon's Key: Does Google Show How the Semantic Web Could Replace Public Key Infrastructure?, 2002. http://www.w3.org/2002/03/key-free-trust.html.

[146] RESCORLA, E. HTTP over TLS. Tech. Rep. Internet RFC 2818, IETF, May 2000. http://www.ietf.org/rfc/rfc2818.txt.

[147] RESNICK, P., AND VARIAN, H. R. Recommender systems. In *Communications of the ACM* (New York, NY, USA, Mar. 1997), vol. 40, ACM Press, pp. 56–58. http://doi.acm.org/10.1145/245108.245121.

[148] SEABORNE, A. RDQL – a query language for RDF. World Wide Web Consortium, Member Submission SUBM-RDQL-20040109, Jan. 2004. http://www.w3.org/Submission/2004/SUBM-RDQL-20040109.

[149] SIMPSON, J. A., AND WEINER, E. S. C., Eds. *The Oxford English Dictionary*, second ed. Oxford University Press, 1989.

[150] SKIENA, S. *Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica*. Addison-Wesley, July 1990, pp. 181–187.

[151] STALLMAN, R. The GNU project. In *Open Sources – Voices from the Open Source Revolution*, C. Dibona, S. Ockman, and M. Stone, Eds. O'Reilly & Associates, 1999. http://www.gnu.org/gnu/the-gnu-project.html.

[152] TECHNORATI. Technorati. http://www.technorati.com/.

[153] THE UNICODE CONSORTIUM. *The Unicode Standard, Version 5.0*. Addison-Wesley, Reading, MA, USA, 2006.

[154] TURCK SOFTWARE ST. PETERSBURG. Turck MMCache for PHP, 2002. http://turck-mmcache.sourceforge.net/index_old.html.

[155] WHITE, E. B., AND HAMNER, E. Charlotte's web. Film, 1973. See also: http://www.imdb.com/title/tt0070016/.

[156] WIKIPEDIA. Eaccelerator — wikipedia, the free encyclopedia, 2006. http://en.wikipedia.org/w/index.php?title=EAccelerator&oldid=54663754.

[157] WIKIPEDIA. Web 2.0 — wikipedia, the free encyclopedia, 2006. http://en.wikipedia.org/w/index.php?title=Web_2.0&oldid=66101697.

[158] WIKIPEDIA. Ajax (programming) — wikipedia, the free encyclopedia, 2007. [Online; accessed 27-March-2007].

[159] WIKIPEDIA. Digital signature — wikipedia, the free encyclopedia, 2007. [Online; accessed 30-March-2007].

[160] WIKTIONARY. Definition: canonical, 2006. http://en.wiktionary.org/w/index.php?title=canonical&oldid=1175162.

[161] WIKTIONARY. Definition: credibility, 2006. http://en.wiktionary.org/w/index.php?title=credibility&oldid=1177659.

[162] WILLIAMSON, S., KOSTERS, M., BLACKA, D., SINGH, J., AND ZEILSTRA, K. Referral whois (RWhois) protocol V1.5. Tech. Rep. Internet RFC 2167, IETF, June 1997. http://www.ietf.org/rfc/rfc2167.txt.

[163] YAHOO! INC. Bloglines. http://www.bloglines.com/.

[164] YAHOO! INC. Yahoo! maps, driving, and traffic. http://maps.yahoo.com/.

[165] YAO, W. T.-M. *Trust Management for Widely Distributed Systems*. PhD thesis, Jesus College, University of Cambridge, Feb. 2003. http://www.cl.cam.ac.uk/Research/SRG/opera/publications/Theses/wtmy2.pdf.

[166] YERGEAU, F. UTF-8, a transformation format of Unicode and ISO 10646. Tech. Rep. Internet RFC 2044, IETF, Oct. 1996. http://www.ietf.org/rfc/rfc2044.txt.

[167] ZIMMERMAN, P. R. The official PGP user's guide, 1995.

[168] ÇELIK, T. The elements of meaningful XHTML, 2006. http://tantek.com/presentations/2005/09/elements-of-xhtml/.

Appendix A

Glossary

AJAX Asynchronous JavaScript and XML, a technique which allows a web page to be updated without reloading the entire page from the web server.

Atom The Atom Syndication Format aims to bring together the fragmented RSS standards for web feeds, and is now a proposed Internet standard. The name Atom also applies to the Atom Publishing Protocol.

CSS Cascading StyleSheets, a standard which allows the separation of style and content in HTML or XML.

DC Dublin Core.

Dublin Core A set of metadata elements with a set of conventions for describing resources (especially online) in a number of different formats. This metadata is hoped to make specific resources easier to find.

FOAF Friend of a Friend, an RDF vocabulary which allows the description of personal attributes and relationships between people.

GRDDL Gleaning Resource Descriptions from Dialects of Languages, a technique for transforming subsets of other languages (such as HTML) into RDF.

HTML HyperText Markup Language, the markup language designed for the Web, and the language in which most web pages are written. HTML allows the description of a document's structure (paragraphs, headings, etc) and links between documents. HTML has gone through a number of revisions (currently HTML 4.01), and its future is in XHTML.

IRI Internationalised Resource Identifier, a generalisation of URI which allows the use of Unicode characters, as opposed to the limit to ASCII (American Standard Code for Information Interchange) characters which URIs have.

N3 Notation 3, a readable RDF syntax. N3 provides a number of shorthand features to make reading and writing RDF by humans easier.

N-Triples A very constrained subset of N3 optimised for use by simple scripts.

Ontology On the Semantic Web, a specification or description of terms which may be used to describe data within a specific knowledge domain. Ontologies are usually described using RDFS and OWL.

OWL Web Ontology Language, an RDF vocabulary used to specify semantics of vocabularies. OWL introduces new semantics to RDF, and is used to describe the meaning of terms, and the relationships between them in a way which is meaningful to computers.

Phishing Fraud designed to entice the victim into divulging sensitive information to an untrusted source. This is usually achieve by misleading the victim into thinking they are logging in or otherwise providing information to a trusted site, such as their bank.

R3X Redland Restricted RDF/XML, a subset of RDF/XML by Morten Frederiksen

RAP RDF API for PHP, a programming interface for dealing with RDF concepts and constructs, written purely in PHP.

RDF Resource Description Framework, a data model to realise the Semantic Web.

RDF Schema Provides the basics required for describing a new RDF vocabulary. As the main constructs in RDFS, it supports a simple class system for objects, and the basic description of properties.

RDFS RDF Schema

RDF/XML The XML serialisation of RDF, which allows the exchange of RDF data.

Redland An RDF application framework which provides APIs for dealing with various RDF constructs. Redland provides a number of different storage backends, can parse and serialise different RDF serialisations, is object oriented and has language bindings in a number of popular languages.

RSS One of a number of related specifications designed for publishing collections of links, commonly used for syndication of personal weblog (journal), news information and other frequently updated material, such as podcasts. Common RSS specifications include the 0.9 series (Rich Site Summary), 1.0 (RDF Site Summary) and 2.0 (Really Simple Syndication).

Scutter A scutter is a piece of software which parses RDF data downloaded from the Semantic Web and follows links in that data. This is parallel to the Web terms "robot" and "spider".

Scutterplan An initial document with which a scutter starts. A scutterplan usually simply contains a number of links to other documents on the Semantic Web.

Semantic Web A vision of the creator of the World Wide Web to create a "web of knowledge". The Semantic Web allows rich descriptions of resources which will support advanced features such as reasoning across the data.

Smush Data in RDF can be merged based on uniquely identifying properties. Two entities which are described separately may in fact be the same entity, and can be linked in an aggregated database. This linking process is known as "smushing".

SPARQL The Simple Protocol and RDF Query Language, a query language for RDF data.

SWIG Simplified Wrapper and Interface Generator, a tool to interface programmes and libraries written in C and C++ with scripting languages such as PHP or Python.

TLS Transport Layer Security, a standard similar to SSL.

Turtle The Terse RDF Triple Language, a simple RDF serialisation which which extends N-Triples by taking useful features of N3, while ensuring that the syntax may only describe RDF graphs. Turtle is a subset of N3.

URI Uniform Resource Identifier, a sequence of characters which is used to identify a resource. A URI consists of two parts: the URI scheme, and the scheme-specific section. These two parts are separated by a colon.

URL Uniform Resource Locator, a type of URI which can be used as an address to retrieve a representation of the resource. For example, all `http:` URIs are URLs.

Vocabulary Used interchangeably with "ontology" in the context of the Semantic Web.

Web A common contraction for World Wide Web.

Web 2.0 A new way of looking at Web technologies and practices, which encourages using the strengths of the Web. It includes ideas such as harnessing the network effect, promoting user participation rather than centralised publishing and providing software as a service.

World Wide Web An information space envisioned by Tim Berners-Lee in 1989. The basis of the Web is the concept of the URI to create a uniform information space, HTML to provide a standard means of marking up documents for display to users, and HTTP as a simple protocol for file transfer.

WWW World Wide Web.

XHTML eXtensible HyperText Markup Language, a reformulation of HTML in XML.

XML eXtensible Markup Language, a simplified subset of SGML (Standard Generalised Markup Language), designed for creating special-purpose markup languages, in turn used to describe data.

Appendix B

FOAF to XHTML XSL

This XSL stylesheet was used to transform an RDF/XML document using the FOAF vocabulary into an XHTML document, displayed as a list of images. Only properties relevant to display (image URL and names of people depicted in the picture) are used in this stylesheet.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns="http://www.w3.org/1999/xhtml"
   xmlns:foaf="http://xmlns.com/foaf/0.1/"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
<xsl:template match="/">
<html>
<head>
<title>Test RDF to XHTML</title>
</head>
<body>
<h1>FOAF (RDF/XML) to XHTML test</h1>
<xsl:for-each select="//foaf:Image">
<p style="text-align: center">
<img>
  <xsl:attribute name="src">
    <xsl:value-of select="@rdf:about" />
  </xsl:attribute>
</img>
</p>
  <xsl:if test="foaf:depicts">
```

```
      <p style="text-align: center">
      <strong>Depicts: </strong>
      <xsl:for-each \
           select="foaf:depicts[@rdf:nodeID]|foaf:depicts/foaf:Person|\
                   foaf:depicts/rdf:Description/rdf:type\
                   [@rdf:resource='http://xmlns.com/foaf/0.1/Person']/..">
        <xsl:if test="position()!=1">, </xsl:if>
        <xsl:choose>
          <xsl:when test="foaf:name">
            <xsl:value-of select="foaf:name" />
          </xsl:when>
          <xsl:otherwise>
            <xsl:variable name="thename" select="@rdf:nodeID" />
            <xsl:value-of \
              select="//foaf:Person[@rdf:nodeID=$thename]/foaf:name|\
              //rdf:Description[@rdf:nodeID=$thename]/rdf:type\
              [@rdf:resource='http://xmlns.com/foaf/0.1/Person']/../foaf:name" />
          </xsl:otherwise>
        </xsl:choose>
      </xsl:for-each>
      </p>
    </xsl:if>
    <hr width="50%"/>
  </xsl:for-each>
  <xsl:for-each select="//foaf:Person|//rdf:Description/rdf:type\
                        [@rdf:resource='http://xmlns.com/foaf/0.1/Person']/..">
    <xsl:if test="position()!=1">, </xsl:if>
    <xsl:value-of select="foaf:name" />
  </xsl:for-each>
  </body>
  </html>
  </xsl:template>
  </xsl:stylesheet>
```

Appendix C

Scutter/smusher code

The following code, discussed in Chapter 6, gathers information from the Semantic Web, parses it and stores it in an RDF triple store, and saves the file (for possible future use) on the filesystem. The scutterplan is contained in the first URL, and is presented in Appendix D.

```
#!/usr/bin/python
import RDF
import urllib
import re
import urlparse
import os
seenURIs = []
newURIs = []
newURIs.append(RDF.Uri(string="http://russell.rucus.net/foaf.rdf"))
parser=RDF.Parser("raptor")
if parser is None:
   raise "No rdf/xml parser"
def httperrcodehandler(url, fp, errcode, errmsg, headers, data=None):
   if errcode == 401:
      print " Not authorized..."
   if errcode == 404:
      print " Not found..."
   if errcode < 400 and errcode >= 300:
      "some sort of redirect, ignoring"
_sub = re.compile("([^\\-/a-zA-Z0-9\\.])").sub
def _sc(ch):
   return "_%02x" % ord(ch.group(1))
```

```
def normalize_path(path, sub=_sub, sc=_sc):
   if "/" not in path:
      path += "/"
if path.endswith("/"):
      path += "index"
   if path and path[0] == "/":
      path = path[1:]
   path = sub(sc, path)
   return path
def localize_path(url):
   splitd = urlparse.urlsplit(url)
   if splitd[3]:
      return splitd[1] + "%s?%s" % splitd[2:4]
   else:
      return splitd[1] + splitd[2]
seeAlso=RDF.Statement(subject=None,
                      predicate=RDF.Node(
                       uri_string="http://www.w3.org/2000/01/rdf-schema#seeAlso"),
                      object=None)

while len(newURIs) > 0:
   storage=RDF.Storage(storage_name="mysql",
                    name="russmush",
                    options_string="host='localhost',
                    database='russmush',user='russmush',password='smushit'")
   if storage is None:
      raise "new RDF.storage failed"
   model=RDF.Model(storage)
   if model is None:
      raise "new RDF.model failed"
   parser=RDF.Parser(name="rdfxml",mime_type="application/rdf+xml")
   if parser is None:
      raise "Could not find any rdf/xml parser"
   mstorage=RDF.Storage(storage_name="memory", name="test",
                        options_string="new='yes',hash-type='memory',dir='.'")
   if mstorage is None:
      raise "No Storage"
   mmodel=RDF.Model(mstorage)
   if model is None:
      raise "No model"
```

```
uri=newURIs.pop(0)
print "Doing stuff with: ",uri
fuo = urllib.FancyURLopener()
fuo.http_error_default = httperrcodehandler
try:
   readfile = fuo.open(uri.__str__())
except IOError, err:
   print " Something went wrong: ", err
   readfile = None
if readfile:
   data = readfile.read()

   path = os.path.normpath(normalize_path(localize_path(uri.__str__())))
   while os.path.isdir(path):
      path = normalize_path(path + "/")
   if os.path.exists(path):
      print " Weird, the file already exists... overwriting"
   head, tail = os.path.split(path)
   if not os.path.exists(head):
      os.makedirs(head)
      print "Creating ", head
   savefile = open(path, "wb")
   savefile.write(data)
   savefile.close()

   try:
      parser.parse_string_into_model(mmodel, data, uri)
      contextnode = RDF.Node(uri);
      model.add_statements(mmodel.as_stream(), context=contextnode)
      model.sync()
   except Exception, err:
      print " Failed?: ", err
      print " Ignoring ", uri, " for the rest of eternity"
   else:
      for statement in mmodel.find_statements(seeAlso):
         if statement.object.is_resource()
             and statement.object.uri not in seenURIs
             and statement.object.uri not in newURIs:
            newURIs.append(statement.object.uri)
            print "   Adding ",statement.object.uri
else:
```

```
print
 " Something went wrong opening ",uri,", I'm going to forget about it"
```

Appendix D

Scutterplan

The following scutterplan was used to gather FOAF data for further experimentation.

1. http://www.w3.org/2000/06/webdata/xslt?xslfile=http://www.ldodds.com/foaf/bulletin.xsl&xmlfile=http://cgi.w3.org/cgi-bin/tidy?docAddr=http%3A%2F%2Frdfweb.org%2Frweb%2Fwiki%2Fwiki.pl%3FFOAFBulletinBoard
2. http://xml.mfd-consult.dk/foaf/scutter.rdf
3. http://swordfish.rdfweb.org/discovery/2001/08/codepict/scutterplan.rdf
4. http://www.perceive.net/xml/googlescutterNoChatlogs.rdf
5. http://jibbering.com/2002/10/rdf-wiki.1
6. http://www.fotothing.com/foaf.php?id=dom
7. http://www.23pools.com/~bandri.xrdf
8. http://clique.ru.ac.za/foaf2
9. http://dougal.gunters.org/dougal.foaf
10. http://affy.blogspot.com/david_mark_medinets.foaf
11. http://search.cpan.org/src/BTROTT/XML-FOAF-0.03/t/samples/person.foaf

12. http://search.cpan.org/src/BTROTT/XML-FOAF-0.03/t/samples/person-lower-case.foaf
13. http://www6.vc-net.ne.jp/~ainosato/valid-chan.foaf
14. http://www.mindswap.org/rdf/instance/?inst=http://www.mindswap.org%2F~katz%2F2002%2F11%2Fjordan.foaf
15. http://www.wosio.flnet.org/foaf/wosio.foaf
16. http://caprahircus.ws/files/adrian.foaf
17. http://www.gnu.org/projects/dotgnu/xml/src/people.foaf
18. http://dave.org.uk/dc.foaf
19. http://seanmcgrath.blogspot.com/SeanMcGrath.foaf
20. http://markpasc.org/index.foaf
21. http://interalia.org/ulrichp.foaf
22. http://moins.de/~ny/stuff/ny.foaf
23. http://odin.himinbi.org/will_holcomb.foaf
24. http://planet.volity.net/feeds.foaf
25. http://www.asynchronous.org/jsled.foaf
26. http://www.goodner.us/jam/marc/marcg.foaf
27. http://www.dwerg.net/ontology/me.foaf
28. http://www.boxuk.com/upload/rdf/dan_zambonini.foaf
29. http://bshanksserver.dyndns.org/trusttest/david.foaf
30. http://bshanksserver.dyndns.org/trusttest/cara.foaf
31. http://introspector.sourceforge.net/daml/mdupont777.foaf
32. http://msdneventsbloggers.net/Bloggers.foaf
33. http://msdneventsbloggers.net/Bloggers/Microsoft.foaf
34. http://technneteventsbloggers.net/Bloggers.foaf
35. http://technneteventsbloggers.net/Bloggers/Staff.foaf

36. http://valid-chan.m78.com/andnow/valid-chan.foaf
37. http://techedbloggers.net/Bloggers/TechEd2005/Orlando/Press.foaf
38. http://techedbloggers.net/Bloggers/TechEd2005/Orlando/Staff.foaf