UNIVERSITY OF SOUTHAMPTON

Trusted Collaboration in Distributed Software Development

by

Ellis Rowland Watkins

A thesis submitted in partial fulfillment for the degree of Doctor of Philosophy

in the

Faculty of Engineering, Science and Mathematics

School of Electronics and Computer Science

June 2007

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING, SCIENCE AND MATHEMATICS
SCHOOL OF ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

by Ellis Rowland Watkins

Distributed systems have moved from application-specific, bespoke and mutually incompatible network protocols to open standards based on TCP/IP, HTTP, and SGML - the foundations of the World Wide Web (WWW). The emergence of the WWW has brought about a revolution in computer resource discovery and exploitation across organisational boundaries. Examples of this can be seen with recent advances in Security and Service Orientated Architectures such as Web Services and Grid middleware. Expansion of the WWW has seen the development of the Semantic Web, a layer on top of the WWW where content is enriched and made interoperable through standards such as RDF and OWL.

Our work in these fields has brought together different ideas to further the advancement of version control; the Semantic Web, Service Orientated Architectures, strong cryptography and the highly dynamic and collaborative WikiWikiWeb. Our online collaborative tool takes advantage of Description Logics, Named Graphs, digital signatures and Grids technologies, to improve collaboration for software engineers working in distributed software development, using semantic knowledge federation and inference rules. Such a system goes well beyond any current version control technology and demonstrates the value and future potential of Semantic Web technologies over traditional Relational Database Management Systems and overly expressive logics such as Prolog.

Contents

List of Figures

List of Tables

Acknowledgements

I owe considerable thanks to Dr. Denis Nicole for his enthusiasm, thoughts, opinions and supervision during my research as well as feedback on my thesis. I would like to thank Dr. Jeremy Carroll for inspiring me to develop an RDF digital signature mechanism based on Named Graphs. Chris Bizer and Richard Cyganiak also deserve special thanks for their encouragement and support as well as for accepting my RDF digital signature implementation into NG4J.

The drafting of this thesis was made possible by the Ecletex LaTeX 2_ε Eclipse plugin by Ian Hartney. I have found this plugin extremely useful and thank Ian for taking the time to write it. I also thank Dr. Steve Gunn for his ECS LaTeX 2_ε Postgraduate Thesis style class and BibTeX style.

Family members have been a significant source of morale during both my research and write up. I would like that thank both my mum, Rae Watkins and my sister, Rebekah Watkins for not only checking my thesis for hideous split infinitives which I am rather fond of, but also the odd hyperbole.

Last but not least I would also like to thank my wife, Junjie Watkins, for her constant support.

To my wife, who is the sunshine in my life.

Chapter 1

Introduction

1.1 Distributed Systems and the Birth of the Web

Over the past twenty years, distributed systems have matured from government and university-run research projects over bespoke and incompatible communication protocols to interoperable, open standard (TCP/IP)[1] and scalable architectures used in heterogeneous environments [Foster (2002)]. Much of this has been due to the emergence of the World Wide Web (WWW), a global network of servers connected using the HyperText Transport Protocol (HTTP) [Fielding et al. (1999)].

Started as a way for scientists and researchers to access information efficiently at the European Organisation for Nuclear Research (CERN), Professor Sir Tim Berners-Lee [Berners-Lee (1989)][2] and contributors have developed the WWW [Berners-Lee (2000)] from a small project into a global communication system used for everything from online web logs, so-called *blogs* [Fujiki (2005)] and secure e-commerce websites[3], to advanced distributed systems based on the eXtensible Markup Language (XML) [Bray et al. (2004)]. Current research on the WWW has been expanded to investigate the use of knowledge representation languages in the form of the Semantic Web.

[1] http://faqs.org/rfcs/rfc793.html.

[2] Professor of Computer Science, Intelligent Agents and Multimedia Group, School of Electronics and Computer Science, University of Southampton.

[3] For example, http://ebay.co.uk and http://www.amazon.co.uk, two of the few companies to survive the dot-com bubble [Chapman (2003)].

1.1.1 The Semantic Web

The Semantic Web [Berners-Lee et al. (2001)] is an attempt to enrich the current and future content of the WWW with standards-based semantic markup where resources can be aggregated and reasoned about using languages such as the Resource Description Framework (RDF) [Klyne and Carroll (2004)] and expressive Description Logic (DL) [Nardi and Brachman (2002); Calvanese and Giacomo (2003)] languages, for example, sub-languages of the Web Ontology Language (OWL) [Bechhofer et al. (2004)]. The Semantic Web is designed to be layered on top of the existing WWW and not as a replacement. The advantage of ontology-based semantic markup is seen as a way to achieve interoperability between different problem domains, in a machine readable manner. Different software agents are able to aggregate information from various sources, then use DL tools to increase their own knowledge-base [Berners-Lee et al. (2001)]. OWL-S [Martin et al. (2005)] is a good example of semantic annotation increasing interoperability with some users even developing methods for additional web service descriptions using domain ontologies [Sabou et al. (2005)]. We envisage that as semantic information increases on the WWW, so will interoperability and inter-domain understanding.

There are doubts, however, as to the suitability and potential benefits of the Semantic Web to developers and users. Unlike Relational Database Management Systems (RDBMS), which are fairly mature and widely deployed, the use of DL in any real application is severely limited beyond academia. Some researchers believe the Semantic Web and DL to be no better than Prolog and other past-generation Artificial Intelligence (AI) approaches. A thorough investigation of DL, its comparative advantages in a real world problem domain would be advantageous for future research.

1.2 Research Motivation

Current Open Source version control repositories, such as Subversion (SVN) [Collins-Sussman et al. (2004)], Git[4] and GNU Arch [Moffitt (2004)], provide frameworks that track the evolution of documents, managing them in logical structures such as projects and releases. Such systems have matured over the past decades to include management techniques such as branching and merging, versioning of metadata,

[4] http://git.or.cz/.

and even limited cryptographic validation. While older systems relied on custom protocols, the vast majority of newer repositories are accessible with standardised protocols such as HTTP or WebDAV, and use relational databases to optimise performance. Despite all this, these frameworks lack critical features necessary for effective auditing and management in distributed software development.

1.2.1 Server Integrity

SVN, which claims to be more advanced and reliable than its predecessor, CVS, still maintains a trusted server model. Developers will generally trust the server to adequately protect the underlying files and metadata from modification, malicious or otherwise. While this is relatively safe in within a single trusted domain (corporate Intranet), it is inherently problematic when considering an inter-domain deployment across insecure networks (Internet).

In an inter-domain deployment it is inadequate to rely on the provenance of the server itself; once an attacker has access to the machine hosting the repository, the repository as a whole is no longer reliable. Server logs, metadata, and files can easily be modified since integrity is not part of the underlying system. Free Open Source software repositories such as Source Forge[5] act as centralised hubs for software development to a point where trust in the hoster and thus server is *implicit* rather than *explicit*.

1.2.2 Audit Logs

Audit logs are an important component in maintaining computer security, especially in a truster server model. Audit logs can be used in several ways, including[6]:

- Accountability
- Reconstruction
- Intrusion Detection
- Problem Detection

[5] http://www.sourceforge.net/.
[6] http://itmanagement.earthweb.com/columns/article.php/3578916/.

Care must be taken with *what* is recorded to support the above business processes as well as the measures taken to maintain the security and integrity of audit logs. Schneier and Kelsey (1999a,b) also note the importance of audit log security, suggesting the use of a Public Key Infrastructure (PKI) [Rivest et al. (1978)] for maintaining the confidentiality and integrity of audit logs.

The vast majority of version control systems keep log files as records in the event something goes wrong during operation. SVN keeps log messages in repositories as properties of a revision which can be modified (by an administrator) at a later date. Such properties are known as *unversioned properties*; modifications overwrite previous values permanently.

If it is possible for an administrator to modify *unversioned properties* in a trusted server environment, then SVN logs cannot be used to audit repositories for malicious behaviour after the server has been compromised. It is possible for authorship information to be manipulated, leading to a loss of accountability, while the modification of timestamps make reconstruction and intrusion detection more difficult. This means that users cannot rely on the provenance of the server since there is no way to verify the integrity of the metadata recorded by the server.

1.2.3 Management

Management in version control can take many forms; developers can perform simple functions such as branching, merging, expunging deprecated or refactored files from a repository. Administrators typically perform backups, manage access and fix faults in the event of repository failure. The tools that support this functionality are, at best, version control system dependent (command line tools), and at worst data storage dependent (database or files). Data storage is a particular concern since in the event a repository's metadata becomes corrupted. SVN amongst others use a proprietary metadata format that cannot be easily copied without special tools to read a database (Berkeley DB) or a set of files (SVN FSFS).

Along with metadata portability issues, there are issues that relate to *useful* statistics that can be curated from a software repository. Project managers should be able to get an overview of the repository and have facilities that warn when patterns of behaviour in commits reveal problems in the software development team. Source Forge, a popular Open Source Software project host provides trivial statistics of hosted projects based on CVS and SVN commit activity; however, these

features are part of Source Forge software system, not the underlying version control systems and therefore host dependent. The RDBMS-level provides a certain amount of functionality for queries using SQL, however, such queries cannot *infer* new knowledge based on existing information. More complex pattern recognition is only possible with algorithms used in rule-based systems and neural networks.

Rule-based logic languages such as Prolog and Logic Programming [Nilsson and Mabarluszyńnski (2000)] offer one approach to the curation of new knowledge. Such logics are, however, known in general to be undecidable. Description Logics, on the other hand, are decidable [Smith et al. (2004)] and are thus uniquely placed between relational databases and Prolog in terms of expressivity and computational completeness. Decidability is held by Tim Berners-Lee [Berners-Lee (2001)] to be an important feature of the Semantic Web.

When software development is distributed between organisations, curation of statistics becomes more difficult. Repository federation would allow project managers to discover new knowledge across organisation boundaries. This information could be used to help improve the performance of the software development lifecycle by removing inefficient business processes discovered during federation. While many RDBMSs can be accessed remotely, most version control systems do not permit remote access to repository metadata; the result of this is that it is not possible to federated multiple repository sources.

1.3 Research Statement

The purpose of this thesis is to investigate new and novel strategies to improve version control in distributed software development. Firstly, we consider the use of Semantic Web technology as an alternative to the traditional relational database used in Subversion. The merits of using Semantic Web technology as a viable substitute to Subversion should be based on the ability to federate knowledge, the need to *explicitly* not *implicitly* trust servers, interoperability with other Semantic Web data structures, and new facilities that are beneficial to developers.

Secondly, we want to discover whether we can reliably bind provenance to source knowledge contained within a semantic version control repository and use it to infer new knowledge. While Description Logic is in principle a stronger reasoning logic than SQL, there is a need to clarify exactly how useful it is in practical applications that go beyond type classification and consistency validation.

As realistic examples of the types of use of version control systems we have considered two distinct approaches of distributed software development: Free, Libre, Open Source Software (FLOSS) [DiBona et al. (1999, 2005)] development and European Community (EC) Information Society Technology (IST) Framework projects [EC-IST (2006b,a)]. Both types of software project development are useful as case studies since they differ in their philosophy, motivation, funding, management, organisation, and involvement of industry amongst other considerations. These differences are reflected in the way developers develop software and use version control systems. IST projects invariably include industrial partners keen to gain early access to experimental technology and are, therefore, concerned with Intellectual Property Rights (IPR) [Gowers (2006); Miller and Davis (2000)] and licencing issues; project partners are likely to restrict access to version control repositories without Non-Disclosure Agreements (NDA) between third-parties [UKPO]. FLOSS projects take a more liberal philosophy, relying on the community to supply developers to improve software. FLOSS projects unless sponsored by industry tend to progress sporadically on a *best effort* basis, with core developers committing to a common repository; the general public can, in general, have read access to the repository. A common example where the FLOSS approach has succeeded is the GNU[7]/Linux Kernel[8] and X.org X Window System[9] [Scheifler and Gettys (1996)]. A recent report by the European Commission has also shown the FLOSS approach to save businesses money over time [Aigrain et al. (2006)].

The premise behind using the Semantic Web in version control is that although it is an emerging technology, it is based on well established logics, including Description Logic, that provides a suitable semantic foundation. Its semi-structured design, based firmly around RDF syntax [Klyne and Carroll (2004)] and semantics [Hayes (2004)], means that Semantic Web data structures are extremely flexible, and more importantly, can be easily *merged* with other RDF data sources. Graph merging becomes important when we consider the potential of federating the data sources of multiple software repositories, something not currently possible with an RDBMS.

Knowledge federation and ontology reuse are crucial aspects of Semantic Web development, both being considered *best practice*. Ontology reuse is important because to define a new ontology does not help in shared understanding across

[7] GNU is not UNIX®.
[8] http://kernel.org/.
[9] http://x.org/.

problem domains; by leveraging existing work, the semantic content of a system is immediately accessible to tools built for pre-existing ontologies. It then becomes straightforward to federate knowledge scattered over the Semantic Grid [Roure et al. (2003)]. Ontologies including Dublin Core [Watanabe (2001)], Friend of a Friend (FOAF) [Miller and Brickley (2004)], and Description of a Project (DOAP) [Dumbill (2004)] are popular ontologies that can be trivially extended.

Our ontology work has leverages existing, popular OWL-based ontologies to introduce a minimal set of extensions to track the provenance of documents. This Document Provenance (DP) ontology forms part of our approach to recording provenance. The ontology acts as a flexible version control model, based on Delta-V [Whitehead (2001); Hunt and Reuter (2001)]. Rather than taking a typical logging approach found in most version control systems where logs are separate from version metadata, accessible only on the server, and only *side-effects* of the commit process, DP takes a more open approach, representing version and log information in a single RDF graph that can be remotely queried using RDF query languages such as RDQL [Seaborne (2004)] or SPARQL [Prud'hommeaux and Seaborne (2007)]. The information contained in the DP graph includes readily extractable from the source code document being placed under version control. It is important that the information represented in the DP graph is sufficient to reconstruct the commit event, e.g., include the *who*, *when*, *what*, *where*, *why* provenance of the commit.

The second part of our approach to recording provenance is the ability to create relationships between RDF graphs which becomes necessary to facilitate intrusion detection and enforce accountability. We have used Named Graphs [Carroll et al. (2005)] to label RDF graphs, create relationships between graphs, and hence make provenance statements. While simple statements using this provenance mechanism might include the assertion of authorship, such assertions can be modified quite easily due to the flexible nature of RDF [Watkins and Nicole (2006)]. A more robust method of assertion of authorship should include a *digital signature* [NIST (1993)] that signs over all RDF in a commit. A digital signature guarantees the *integrity* of the signed data; any modification of the data would break the signature and thus show intrusion in the repository server. Developers are made accountable for their actions during commits, because digital signatures support *non-repudiation* [McCullagh and Caelli (2000)]. A developer cannot deny creating the digital signature since each signature is uniquely linked to the private key that made it; as such it is impossible to fake a digital signature with a different private key. Authorship of a digital signature also provides IPR attribution; coupled with

non-repudiation, a developer can prove that they hold the IPR on a particular document.

Digital signatures are an important component in our approach to provenance and version control. Digital signatures and non-repudiation are the first steps to trust on the Semantic Web[10]. By making developers sign metadata that enforces accountability, the developer becomes a stakeholder in the integrity of the software repository. The fact the developer is an active participant in the generation of metadata for the commit process means the information recorded by the server cannot be considered *logging* as is the case with SVN. While confidentiality of metadata is an important issue, encryption will hamper the federation of repositories; digital signatures pose a unique challenge on the Semantic Web since RDF does not have a canonical form (see Section 3.4).

It is important to note that our approach to trusted collaboration does not rely on any form of trust metrics [Golbeck and Hendler (2004a); Bizer et al. (2005b)], which we consider beyond the scope of our research. Instead, our use of PKI is based upon well established business-to-business (B2B) trust relationships through the use of standards such as SSL/TLS and WS-Security.

As a practical example of how our research improves version control in software development, we developed a client/server-based Web application that acts as an *online collaborative tool*. Our tool takes advantage of a WikiWikiWeb [Leuf and Cunningham (2001)] interface that makes for simple annotation of developer creativity. Our system automatically generates a single wikipage per file (Class); these can easily link to and from pages devoted to more generic ideas. Each wikipage provides hyperlinks between packages and classes to support routine navigation. This and other information is automatically parsed from Java™ source codes. We have enhanced the JSPWiki[11] implementation to track online resources kept in a WebDAV [Whitehead, Jr. and Goland (1999); Goland et al. (1999)] repository and, more importantly, added a SPARQL interface for distributed data federation based on the GRIA grid middleware. We have developed federation scenarios that show how our online collaborative tool goes beyond the capabilities of the RDBMS, and includes an inference mechanism to help in metadata integrity recovery.

To maintain compatibility with browser-based clients, we have built a small Java™ applet to facilitate secure signing of the metadata in a browser environment, including secure hashes of source files [Watkins and Nicole (2005a)]. This applet

[10] Available at http://www.w3.org/2000/10/swap/doc/Trust.html.
[11] http://www.jspwiki.org/

digitally signs RDF [Carroll (2003)] and stores the result in a Named Graph, and thus validates inputs and detects corruption in the knowledge-base. Our work on RDF digital signatures is now a core component of the Semantic Web Publishing framework (SWP), which is an extension to the Named Graphs for Jena (NG4J) [Bizer et al. (2005a)] project[12].

As part of the evaluation of our implementation we created several federation scenarios that explore trusted data federation to show the value of the Semantic Web beyond OWL entailment. Performance experiments were also conducted to determine the extent to which Semantic Web technology lags behind native RDBMSs. Results from these experiments show that the performance of Semantic Web technology is considerably less than a modern RDBMS (HSQLDB). This is in part due to the immaturity of available tools, and crucially, support from commercial companies. Despite this, the fact that our federation scenarios have been implemented using Semantic Web technology is important since it opens up access to the software repository for more complex analysis, something not possible in RDBMS-based systems.

The original contributions of this thesis lie in three main areas. Firstly, our investigation to determine the viability of using DL as the basis for a version control system rather than an RDBMS. Secondly, the successful binding of trusted provenance to source knowledge using Named Graphs that can be subsequently reasoned over, published in Watkins and Nicole (2006). Thirdly, the development of an online collaborative tool that enables distributed collaborative software development, published in Watkins and Nicole (2005a) and Watkins and Nicole (2005b). This tool further supports two different case studies with a set of federation and digital signature recovery scenarios.

1.4 Thesis Structure

Chapter 2 provides a background review of important topics and technologies used in this thesis, including relational database management systems, current grid technology, public key cryptography, Semantic Web technologies, and the concept of the WikiWikiWeb. Chapter 3 describes our case studies and goes on to analyse the potential benefits of using Semantic Web technology as an alternative to the RDBMS, based on our case studies, and our motivation for RDF digital

[12] http://ng4j.sourceforge.net/.

signatures. Chapter 4 details our design choices for our online collaborative tool and its implementation. Chapter 5 evaluates our research based on quantitative and qualitative results. We conclude in Chapter 6 with a self-evaluation and list our key achievements.

1.5 Declaration

This thesis is based on work done by the author within a collaborative research environment. It is all original work by the author unless explicitly stated otherwise.

Chapter 2

Background

This chapter gives a brief overview of research topics that are relevant to this thesis. These topics cover a broad range of disciplines in Computer Science, including version control, database technologies, grid middleware, cryptography, web technologies and formal logics.

2.1 The WikiWikiWeb: Lightweight Collaboration

In the past, online collaboration was either via email or static web pages that had to be updated manually. Dynamic webpage generation with the PHP Hypertext Preprocessor[1] and Common Gateway Interface technologies not only improved webpage design, but also allowed for content that could be stored in a backend database. As a result we now have weblogs, online forums and the WikiWikiWeb [Leuf and Cunningham (2001)].

The term "Wiki-wiki" is derived from the Hawaiian word for 'quick' [Taylor (2003)]. The WikiWikiWeb's creator, Ward Cunningham[2] describes it as "the simplest online database that can possibly work". Modern relational database management systems (RDBMS) for example, PostgreSQL[3], provide recoverability, transactions, and referential integrity. Perhaps it might be wiser to use an RDBMS to store a Wiki's content? We can describe the WikiWikiWeb as, "an

[1] http://www.php.net/.
[2] http://c2.com/cgi/wiki?WardCunningham.
[3] http://www.postgresql.org/.

interconnected collection of webpages that can be edited by anyone, at any time, from anywhere"[4]. While Tim Berners-Lee argues web logs to be the realisation of the *read-write* Web[5], it can also be argued that Wikis are part of this future.

The WikiWikiWeb has become popular as a collaborative hypermedium due to its simplicity. It is this collaborative [Gillmoor (2004)] philosophy that allows multiple users to efficiently publish document-based material online. This means user authentication is relatively rare; Ward's original WikiWikiWeb does not impose authentication restrictions. Frequently asked questions (FAQ), installation instructions and other documentation are increasingly hosted on WikiWikiWebs. Examples include the Free Desktop Project[6], Wordpress[7] to name but a few. Mayfield (2003) has argued the case for commercial organisations to use WikiWikiWebs for content management.

Because anyone can edit pages and add their own content, people gain a sense of responsibility and a voice, which they might not have elsewhere[8]. This can be a disadvantage when we consider malicious users[9]. The WikiWikiWeb, on the whole, trusts its user base, allowing them to shape the way the WikiWikiWeb develops. If we consider the free community built encyclopedia, Wikipedia[10], we find numerous definitions and articles from contributors all over the world, although it lacks adequate mechanisms to control accountability and content quality. This problem has been noted and argued over in recent years and is summarised in Mayfield (2004b). Indeed, librarians and reporters such as Fasoldt (2004), and online commentators like Ito (2004) have discussed the trustworthiness and validity of Wikipedia's contents. They all agree that it is not a source to use with confidence; even Wikipedia acknowledges that it is not an authoritative source of information[11]. Confidence in the source of a definition or article is what most people want. Commercial encyclopedias, like Encyclopedia Britannica, provide this assurance; Wikipedia does not.

Lack of trust and accountability in the WikiWikiWeb context can be alleviated by registering users who contribute content. In the case of Wikipedia, users are encouraged to register for reasons of intellectual property management. This

[4]Wiki Getting Started FAQ, http://c2.com/cgi-bin/wiki?WikiGettingStartedFaq.
[5]http://news.bbc.co.uk/1/hi/technology/4132752.stm.
[6]http://www.freedsktop.org/.
[7]http://www.wordpress.org/.
[8]http://c2.com/cgi/wiki?WhyWikiWorks.
[9]http://c2.com/cgi/wiki?WhyWikiWorksNot.
[10]http://www.wikipedia.org/.
[11]http://en.wikipedia.org/wiki/Wikipedia:General_disclaimer.

does not necessarily improve the quality of content; users can, however, be made more accountable by other users who might review their contribution. Neither does it completely inhibit those determined enough to undermine the purpose of Wikipedia [Davis (2006); Frommer (2006)]. Moves by the Wikimedia Foundation as reported by Mayfield (2004a) towards a printed edition of Wikipedia have had to consider the introduction of a formal editorial process prior to publication.

Despite criticisms of high profile examples of the WikiWikiWeb it is clearly a powerful tool that is causing much debate. Its collaborative capabilities, while *informal* in nature, nonetheless show promise in both the public domain and business.

2.2 Technology

2.2.1 Relational Database Management Systems (RDBMS)

The Relational Database Management System (RDBMS) is a class of database management systems based on relational model theory by outlined by Codd (1970, 1990). An RDBMS presents data to the user as relations as well as providing a set of operators to manipulate the data. More complex *queries* can be performed on an RDBMS using standardised query languages such as the Structured Query Language (SQL) [Chamberlin and Boyce (1974)].

One of the novel features of the RDBMS is the ability to be accessed remotely with many Open Source and commercial RDBMS implementations follow the client/server model. While remote access is desirable in enterprise environments, there are issues with security and concurrent access. The vast majority of RDBMS implementations support some kind of discretionary and mandatory access control mechanisms [Krause and Tipton (1998)], together with advanced transaction support [X/Open-Group (1992)]. MySQL[12] and PostgreSQL[13] are arguably the most successful Open Source RDBMS products.

[12] http://www.mysql.com/.
[13] http://www.postgresql.org/.

Other RDBMSs are said to be *embedded*, i.e. the database is internal to the application accessing it. Popular examples of embedded RDBMSs include Oracle Berkeley DB[14], SQLite[15], HSQLDB[16] and Apache Derby[17].

2.2.2 Version Control Systems

Version control systems provide a formal and structured approach to handling how documents change over time. Documents under version control each have a history of all changes made. In the event something goes wrong in the build process, developers can revert to previous versions in a version control repository that are known to work. While designed by and for programmers, version control systems are not limited simply to source code; it possible to place any files under version control.

2.2.2.1 Concurrent Versioning System

The Concurrent Version Control System (CVS)[18] is one of the most well known source code management systems to date. Still used in hundreds of Open Source projects, it is capable of simple, yet efficient version control. It replaces the conservative locking strategy of RCS with an optimistic strategy of merging clashes and changes.

CVS supports basic operations such as version branching and merging in a local or client/server environment. Both text and binary files can be placed under version control; text files hold the changes between each version on the same file. The repository does not use an RDBMS.

2.2.2.2 Subversion

Subversion (SVN) aims to be a "compelling replacement for CVS."[19] Subversion improves on CVS by versioning not only files but also directories, meta-data, copy and rename information. Commits to the repository are truly atomic so that the

[14] http://www.oracle.com/database/berkeley-db.html.
[15] http://www.sqlite.org/.
[16] http://hsqldb.org/.
[17] http://db.apache.org/derby/. Originally developed as IBM Cloudscape.
[18] http://www.nongnu.org/cvs/.
[19] http://subversion.tigris.org/.

entire commit must succeed for a commit to take effect. SVN supports two types of repository data stores: Berkeley DB and FSFS[20]. Berkeley DB requires file locking and should therefore not be used on file systems that do not support them, e.g., NFS.

FSFS is Subversion's file-based data store. FSFS approach appears to be a step back to CVS rather than something more revolutionary. Developers claim SVN to be more scalable, however, it appears to be rather similar to CVS, the only difference being the use of standardised interface. It might have been more suitable for the developers of SVN to have used MySQL or similar as a robust and scalable data store alternative.

Another important difference between CVS and Subversion is that Subversion has taken some effort to support W3C standards such as WebDAV for access to Subversion repositories.

2.2.2.3 Git

Git[21] is a "directory content manager" design to handle very large projects like the Linux kernel[22] [Loeliger (2006b,a)]. Git falls into the category of *distributed* source code management, similar to Arch and Darcs.

Git uses two different persistent storage formats based on bandwidth availability, known as *packed* and *unpacked* storage. Version histories use an directed acyclic graph (DAG) structure so that long-lived branches and repeated merging become more simple to perform. Both GNU Arch and Darcs have taken some of the core concepts from Git and integrated them for future releases.

2.2.2.4 GNU Arch

GNU Arch[23] is a version control system for distributed source code management. It claims to be easy to use and geared towards Open Source development, including GNU/Linux Kernel development. GNU Arch and Git share similar aims, since

[20] http://web.mit.edu/ghudson/info/fsfs/. Not to be confused with the Fast Secure File System (FSFS), available at http://fsfs.sourceforge.net/.
[21] http://git.or.cz/.
[22] http://www.kernel.org/.
[23] http://www.gnu.org/software/gnu-arch/.

the public domain became aware that kernel developers were using BitKeeper[24], a proprietary version control system, for kernel development.

2.2.2.5 Darcs

Darcs[25] takes a different approach to other version control systems, claiming to base itself on the fundamentals of quantum mechanics. Instead of files being stored and versioned on a remote server, patches similar to diff[26] files are versioned and re-applied to files contained on the local file system. This approach is known as the *theory of patches*[27] [Roundy (2006)]. Darcs theory of patches concept only works at the syntactic level, it is not able to track semantic changes to code.

2.2.3 Service Orientated Architecture and Web Services

Service Orientated Architecture (SOA) [Erl (2005)] is an implementation-agnostic distributed systems paradigm significantly different from earlier distributed object systems such as Sun Microsystems' Remote Procedure Call (RPC). SOA keeps the client and service as loosely coupled as possible without impairing communication. Web services in particular exemplify this approach with Simple Object Access Protocol (SOAP), an XML protocol. SOAP is one of the most widely used standards adopted for Web Service interoperability and overcomes the class dependency problem found in distributed object systems in several ways:

- Only supports interfaces taking published, possibly composite, datatypes
- Abstraction from underlying transport
- Provides a standard schema language (WSDL) for signatures of functions
- Extra wrapper information in SOAP envelope (for intermediaries)

While SOAP has been accepted as an industry standard, there are those who believe it to be cumbersome and over-complicated and advocate a less abstract and unlayered approach that is directly tied to HTTP, that restricts methods on

[24] http://kerneltrap.org/node/[4966, 444]/.
[25] http://darcs.net/.
[26] DiffUtils, http://www.gnu.org/software/diffutils/diffutils.html.
[27] http://darcs.net/DarcsWiki/WhyYouWantPatchTheory.

documents to those provided by HTTP implemented as CRUD (Create Retrieve Update Delete) semantics. This is known as the *ReST* approach.

Representational State Transfer (ReST) is an alternative approach to web service design that uses a small fixed set of *untyped* methods, found in HTTP [Fielding (2000)]. ReST can be seen as a distributed object system, however, it avoids the class dependency problem of RPC by having only one class, the Document. ReST advocates resources represented by Universal Resource Identifiers (URI). Data is commonly encoded as parameters in a URI, for example:

http://example.org/ReST/Shop?type=car&order=ascending

Typing the above URI in a browser would typically use the HTTP GET method, whilst sending data in a web form would use POST. The server might then perform a database lookup and return the contents of that URI back to the browser. The importance of the ReST approach to web service design is that data is sent to a URI, **transformed**, then returned.

More recently, Grid frameworks have begun to reap the benefits of advances in SOA and Web Service technologies, including WS-Security, WS-Addressing, WS-Policy, WS-SecurityPolicy, etc.. As web services do not pass objects a mechanism has to be found to represent instances (conversations) if we wish to allow for stateful services. WS-Addressing [Box et al. (2004)] is an attempt to identify web service endpoints and conversations. The service passes an *opaque* object reference (conversion identifier) to the client which can only be resolved by the service. It is vital that the opaqueness of the reference is maintained.

Several Grid frameworks make use of several WS-* standards and drafts[28]. These include the WSRF [Czajkowski et al. (2004b); Foster et al. (2005); Humphrey et al. (2005)], as well as many DTI[29] and EC projects[30] such as the Open Middleware Infrastructure Institute (OMII)[31] and Grid Resources for Industrial Applications (GRIA)[32]. Other researchers in this domain have begun enriching grid and web service technology with the Semantic Web to develop semantic grid services [Roure et al. (2005)]. All these systems are open source, and most conform to published standards; interoperability is, nonetheless, problematic.

[28]The vast majority of these specifications remain as drafts.
[29]http://www.dti.gov.uk/.
[30]http://www.cordis.lu/ist/.
[31]http://omii.soton.ac.uk/.
[32]http://www.gria.org/.

2.2.4 The Grid

The Grid is an emerging computing model for high throughput computing in heterogeneous networks. Using a set of open standards and protocols it enables access to disparate resources whether they be processing power, data, or storage capacity. In many ways the grid is a new type of parallel and distributed system that can take advantage of commodity off-the-shelf components and uses *middleware* to coordinate the allocation of resources.

2.2.4.1 Globus Toolkit

The Globus Toolkit [Foster and Kesselman (1997)] is the main product of this research, a collection of software services and libraries which are supposed to allow organisations around the world to build computational grids and develop applications that are described as "grid enabled", that is to say, usable in a grid environment. Like UNICORE, it uses a Public Key Infrastructure (PKI) for security and authentication, although in a less standard manner[33].

Since there are many technical challenges with respect to the development and deployment of computational grids the Globus Project has focused on the following research areas to satisfy its aims:

- Resource Management
- Data Management and Access
- Application Development Environments
- Information Services
- Security

It is important to note here, that earlier versions of Globus (2.x) did not support Business-to-business (B2B) applications. This was partly due to Globus' architecture that relied on conventional sockets and LDAP-based distributed computing, coupled to an ad hoc job submission language and the novel use of "Proxy Certificates" to support personalisation in a modified X.509 environment. Modern grid service architectures, based on web services are becoming popular to support B2B applications as well as scientists and engineers.

[33] A non-standard Object Identifier (OID) is used to distinguish a Globus Proxy Certificate from standard X.509v3.

2.2.4.2 Grid Services

The Open Grid Services Architecture (OGSA) defines the structure and standard methods for a grid service as part of the Globus 3 development [Foster et al. (2002)]. Grid services, which build upon web services, can be advertised to applications and users across the Internet and access through web browsers, enabling the processing of data or problems remotely. Unlike regular web services, grid services have state.

The Open Grid Services Infrastructure (OGSI) [Tuecke et al. (2003)] is a specification that implement OGSA capabilities with web services. It attempts to build a distributed object model on top of web services and has now been abandoned. It has since been superseded by the Web Service Resource Framework (WSRF) in conjunction with WS-Notification [Czajkowski et al. (2004a)]. Three implementations of the WSRF exist, two written in Microsoft's .NET Framework, using C# [ECMA (2005)]; a third written in Java, intended to be the basis for future releases of Globus (GT4[34]).

WSRF.net A Microsoft .NET implementation of the WSRF, under active development by University of Virginia Grid Computing Group [Wasson et al. (2003)]. It provides a container framework on which to do WSRF compliant grid computing on the .NET Platform[35].

MS.NETGrid Project Part of the UK eScience Initiative, the EPCC at Edinburgh University have also developed an implementation of the OGSI as an example of using .NET with Grid services [Byrne et al. (2003, 2004)]. Limited to a single twelve month period (3rd March 2003 – 19 March 2004), the project has released MS.NETGrid-OGSI Release 2.0 and made all deliverables available[36].

Globus Toolkit 4.x As part of the on going development by its partners, the Globus Alliance[37] has released their latest version of the Globus Toolkit as an independent implementation of the WSRF (previously OGSI) alongside a Java™ implementation of the still-popular 2.x GT.

[34] Globus Toolkit 4.0, http://www-unix.globus.org/toolkit/.
[35] OSGI.net, http://www.cs.virginia.edu/~gsw2c/ogsi.net.html.
[36] Available at http://www.epcc.ed.ac.uk/~ogsanet/.
[37] http://www.globus.org/alliance/.

2.2.4.3 Sun Microsystems' Grid Engine

The Grid Engine is an open source community project aimed at increasing the adoption of distributed computing solutions [Bulhoes et al. (2004)]. Based historically on Genias' Codine[38] grid system, the Grid Engine's purpose is to allow uniform access to heterogeneous resources. While it describes itself as a Grid system, it is not much more than an advanced batch processing system and has been successfully used as a resource manager in GRIA.

2.2.4.4 Legion

Legion is a distributed system, developed at the University of Virginia, with the view of being a massively disparate virtual computer. It is large scale system, designed to tie together millions of hosts with high speed links. Users logged into the system view this vast collection as a single computational entity, with access to all kinds of data and physical resources connected to each participating host. Users can work together as groups, with access to virtual "work spaces". Transparent access is achieved using Legion's scheduling, data management, fault tolerance, site autonomy, and several security options[39].

2.2.5 EC IST Framework Grids

Several grid middlewares exist that have been funded under the EC IST Framework Programme, Unit F2[40]. Future grid projects in Framework 7[41] will be based upon the recommendations of the Next Generation Grids Expert Group (NGG) [Group (2006)].

2.2.5.1 GRIA

The Grid Resources for Industrial Applications (GRIA)[42] was an EC IST FP5 project aimed at bringing grid middleware to business processes. Rather than concentrate on high-throughput computing, GRIA recognises the need for distributed

[38] http://cch.loria.fr/documentation/batch/GRD_CODINE/.
[39] http://legion.virginia.edu/overview.html.
[40] Unit D3, Software & Service Architectures and Infrastructures in FP7.
[41] http://cordis.europa.eu/fp7/home_en.html.
[42] http://www.gria.org/.

resource management, a distinct departure from the classic centrally managed Virtual Organisation model in other grid middlewares. GRIA works on the basis of bilateral Service Level Agreements (SLAs) between organisations and supports federated access to resources meaning each organisation independently has control over who is authorised access.

2.2.5.2 gLite

gLite[43] is the Open Source grid middleware of the EC IST FP6 Enabling Grids for E-sciencE project (EGEE)[44]. gLite follows the SOA approach that promotes interoperability and aims to comply with emerging standards such as OGSA from the Open Grid Forum (OGF)[45]. It is comprised of a set of core services that provide basic functionality required by its users: security, information and monitoring, job management and data services.

gLite has been extremely successful in EGEE-related projects (including EGEE-II) to a point where there are an estimated 180 sites as part of the EGEE infrastructure[46].

2.2.6 China Grid

China's eScience programme has produced two significant grids: China Network Grid (CNGrid)[47] and China Resources Over Wide-Area Network (CROWNGrid)[48]. European collaboration with China has seen the inclusion of CROWNGrid in the OMII-Europe project, CROWNGrid and CNGrid in the OMII-China project, as well as CNGrid in the EC IST FP6 Bilateral Research and Industrial Development Enhancing and Integrating GRID Enabled Technologies (BRIDGE) project [Kalb (2006)].

[43] http://www.glite.org/.
[44] http://eu-egee.org/.
[45] http://www.ogf.org/.
[46] http://en.wikipedia.org/wiki/Enabling_Grids_for_E-sciencE.
[47] http://www.cngrid.org/.
[48] http://www.crowngrid.org/.

2.2.7 UK e-Science Projects

Many other national projects exist in the UK under the National e-Science Programme[49], sponsored by the Department of Trade and Industry (DTI)[50]. These projects are supposed to be for the development of applications for a variety of scientific disciplines using Grid technology. Several projects exist in Southampton (GEODISE[51], ^{my}Grid[52], Comb*e*Chem[53], et al.). The Open Middleware Infrastructure Institute (OMII)[54] [Atkinson et al. (2005)] builds on the work done on GRIA (see Section 2.2.5.1) and various UK e-Science projects, including Comb*e*Chem.

2.2.8 Public Key Infrastructure

Authentication and general security are key problems to be addressed in Grid computing. RSA-based[55] [Rivest et al. (1978)] Public key cryptography is currently an accepted method for standardising the behaviour and mechanisms that allow Grid systems to authenticate users. While older systems like Kerberos [Neuman and Ts'o (1994)] are still used on Windows and distributed file systems like AFS, newer mechanisms that are now internationally standardised, like the Public Key Infrastructure (PKI) and the SAML-based [Cantor et al. (2005)] Shibboleth[56], are being used for current projects.

A Public Key Infrastructure (PKI) describes the mechanisms and algorithms for encryption, cryptography and digital signatures, based on public key cryptography and how they maintain the following in terms of users and their data:

- Confidentiality.
- Authenticity.
- Integrity.

[49] http://www.rcuk.ac.uk/escience/.
[50] http://www.dti.gov.uk/.
[51] http://www.geodise.org/.
[52] http://mygrid.man.ac.uk/.
[53] http://www.combechem.org/.
[54] http://www.omii.ac.uk/.
[55] Rivest, Shamir and Adleman. In 1997 it came to light that GCHQ's CESG Research Group [Ellis (1987)] invented what is now know as the Diffie-Hellman key exchange protocol and the RSA algorithm several years prior to their eventual re-discovery and publication in the USA. Ellis (1970) only described the principle of *non-secret* cryptography, whilst Cocks (1973) and Williamson (1974, 1976) described the practical aspects.
[56] http://internet2.edu/shibboleth/

- Non-repudiation (Inability to refute the creation of a message).

All the above features can be found within a public key infrastructure. Integrity and confidentiality are especially important for secure Internet communication and electronic commerce (e-Commerce).

Many PKI libraries are written in Java™, due to Java's portability and acceptance in the Grid community. There follows a list of popular implementations including Sun Microsystems' Java™ Cryptography Extension (JCE)[57].

2.2.8.1 Digital Signatures

Unlike digital encryption, digital signatures do not attempt to obfuscate the contents of a message to maintain confidentiality; digital signatures are, in fact, used to maintain the *integrity* of a message. In public key cryptography, the private key is used to create the signature that can then be verified with the public key. Verification signifies *proof of ownership* of the private key by the sender since only the private key associated with the attached public key could have generated the signature.

To make signing faster and cheaper no matter what size message, digital signatures do not actually sign over the actual message, merely a representation of the message, known as a *digital digest*. Digital digests are one-way operations that produce a unique representation of a message. If the message is changed, so does the digital digest. The success of a digital digest algorithm is based on its ability to resist *collisions*; instances where two different messages have the same digital digest.

2.3 Provenance Frameworks

Traditionally, provenance charts the origin and history of an object, particularly in the fine art world, and is key to associating value with an artwork. On its own, an artwork may have intrinsic value rising from its beauty or utility; but with provenance, intrinsic value can be increased by several orders of magnitude. Provenance for an artwork is asserted by the owner each time it is sold. *Asserting authenticity* is one of two uses of provenance, which is in the service of the buyer of an artwork

[57] http://java.sun.com/products/jce/.

or reader of a rare book. The other use of provenance can be used to the detriment of the buyer, placing obligations and constraints on them. Common examples of this can be seen in the current generation of Digital Rights Management (DRM) systems such as Apple's Fairplay[58] and RealNetworks' Helix DRM[59]. Provenance is not limited to fine art, books and consumer content; substantial research has been done on the use of data and knowledge provenance as well as provenance in SOA [Chen et al. (2005)].

2.3.1 Data Provenance

Buneman et al. (2001b, 2000) have done substantial work on data provenance with regard to relational databases. This research concentrates on *where* data has come from and *why* is it in a database. They use query inversion to compute *where* provenance, until recently an untouched topic. ^{my}Grid continues much of the work done in this area.

The ^{my}Grid project[60] has developed Grid middleware to meet the needs of bioinformatics. In this domain, it is essential to be able to capture and manipulate provenance information. The project takes provenance records from sources such as the Freefluo[61] workflow orchestration tool and uses an ontology to annotate these provenance records for future analysis [Zhao et al. (2003, 2004b,a)]. Szomszor and Moreau (2003) and PASOA (2005) extend this work, exploring the need for a complete framework for data provenance in Service Orientated Architectures (SOA) [Groth et al. (2004)] and Grid environments [Groth et al. (2005)].

2.3.2 Provenance in Service Oriented Architectures

The Provenance Aware Service Oriented Architecture (PASOA) Project continues some of the work done by the ^{my}Grid project on data provenance. Its aim is to investigate the nature of provenance and reason about the accuracy of data and service in the e-Science domain. It has so far developed a provenance recording service, called PReServ, an implementation of the Provenance Recording Protocol (PReP) developed by the PASOA project [Groth et al. (2004)]. PReServ is currently being used by the European IST EU Provenance project. PReServ appears

[58] http://www.apple.com/lu/support/itunes/authorization.html.
[59] http://www.realnetworks.com/products/drm/index.html.
[60] http://www.mygrid.org.uk/.
[61] http://freefluo.sourceforge.net/.

to store provenance defined by an XML schema in a database, and not use any semantic markup. Groth (2005) defines a formal definition of the P-Structure and PReP (Provenance Recording Protocol) used in PReServ (Provenance Recording for Services) modelled as an Abstract State Machine.

2.3.3 Knowledge Provenance

Fox and Huang have made interesting observations about the nature of *knowledge provenance* [Fox and Huang (2003)]. They define Knowledge Provenance (KP) as follows: "Knowledge Provenance is an approach to determining the origin and validity of knowledge/information on the web by means of modelling and maintaining information sources and interdependence, as well as trust relations" [Huang and Fox (2004)]. In addition, they identify four distinct levels of KP:

- Level 1 (**Static KP**) considers the provenance of static information (basic webpages) that can be trivially verified.
- Level 2 (**Dynamic KP**) extends the static KP idea, introducing cases which involve determining the validity of information over time.
- Level 3 (**Uncertainty-orientated KP**[62]) provides insight into information whose validity is inherently uncertain.
- Level 4 (**Judgement-based KP**) intends to focus on the social processes necessary to support KP.

Levels 1-3 are covered in three papers [Fox and Huang (2003); Huang and Fox (2003, 2004)]. Nothing as yet has been published relating to Judgement-based KP.

For Level 1 KP, Fox and Huang (2003) introduce the problem of KP relating to the publishing of sometimes unreliable information that can potentially affect people, such as changes in stock price [Painter (2001)]. They argue that since anyone can publish information on the WWW, any such information may be true, false, uncertain or outdated, noting that there are no suitable tools for discovering the provenance of knowledge for any one resource. Identifying a 'proposition' as the basic unit of KP, they go on to identify other concepts for relating propositions, including "asserted propositions", "derived propositions", "equivalent propositions",

[62] Previously referred to as **Uncertain KP** in Fox and Huang (2003).

and "composite propositions". Developing an ontology using a semi-structured method by Grüninger and Fox (1995), based on the idea of propositions, Fox and Huang provide an approach to describing KP, and axioms for future rule-based reasoning on the meta-data. Predicates in the metadata allow for the annotation of information and the propositions they contain, including marking information with a "truth value". If a particular proposition is said to be true, then subsequent proposition that rely on the first can also be reasoned to be true. A sample implementation was produced in RDF-Schema (RDFS), using XML Digital Signature for verification of the meta-data itself.

In Level 2 KP Huang and Fox (2003) describe the addition of dynamic description to the KP ontology, and investigates how the *truth value* of a proposition can change over time. They find that propositions and further propositions derived from previous ones may only be effective within a specific period, known as the "effective period". After creating additional axioms to take into account the extensions to the ontology, they show reasoning over dynamic KP, demonstrating the ability to tell if an information resource is still within its "effective period".

da Silva et al. (2003) propose an alternative knowledge provenance infrastructure that includes proof-like information on "how a question answering system arrived at its answer(s)." This approach integrates tools such as Inference Web's IWBase [McGuinness and da Silva (2003)] and TAP [Guha et al. (2003)] for information inferencing and source construction.

2.3.4 Provenance Mechanisms

Provenance mechanisms distinguish themselves from the provenance descriptions outlined in Section 2.3 because they are either built into the underlying logic framework or part of the implementation of a particular system. Examples of logic based provenance mechanisms include RDF reification, Named Graphs, contexts, Minimal Self-contained Graphs, and RDF molecules. Quads and contexts tend to be dependent on the triple store used, for example, 3store or RDFStore.

2.3.4.1 RDF Reification

RDF reification, defined in Hayes (2004), was intended as a framework for making provenance statements and other statements about RDF triples. Each triple is described with a special vocabulary as shown in Figure 2.1.

```
_:xxx rdf:type rdf:Statement .
_:xxx rdf:subject <ex:a> .
_:xxx rdf:predicate <ex:b> .
_:xxx rdf:object <ex:c> .
```

FIGURE 2.1: Example reified statement, taken from RDF Semantics Recommendation

Whilst it has no formal semantics, RDF reification is by far the most popular mechanism for attributing provenance for RDF statements. It is, for example, trivial to add arbitrary triples that might relate to the reified triple; Figure 2.2 demonstrates this with the simple attribution of authorship and creation date of the original triple.

```
_:xxx rdf:type rdf:Statement .
_:xxx rdf:subject <ex:a> .
_:xxx rdf:predicate <ex:b> .
_:xxx rdf:object <ex:c> .
_:xxy foaf:maker _:xxx .
_:xxy dcterms:created ‘‘18-9-2006’’.
```

FIGURE 2.2: Reified Statement with Additional Arbitrary Triples

2.3.4.2 Quads

Harris and Gibbins (2003) have written a fast triplestore implementation, 3store [Harris and Gibbins (2003)], that uses quads to track the provenance of triples; this has been used in several novel applications including http://hyphen.info/ and CS AKTive Space [Shadbolt et al. (2003)]. The fourth element keeps a record of the source RDF document where the triple originally came. 3Store also supports RDFS entailment, although there does not appear to be any general purpose inference engine to date.

2.3.4.3 Contexts

Reggiori et al. (2003) use contexts as a means to record provenance in their RDFStore. They see contexts as an additional and orthogonal dimension to the RDF triple. Each RDF statement is flagged as belonging to a specific context. Figure 2.3 shows an example of RDFStore contexts.

Each line contains two triples; the first is the context and the second is the triple that resides in that context. Note that since each context has been defined by different users (X and Y), both triples are in effect in different contexts.

Quality → Defined by → User X:Newspaper A → Quality → "liberal"
Quality → Defined by → User Y:Newspaper A → Quality → "conservative"

FIGURE 2.3: RDFStore Contexts

2.3.4.4 RDFX

RDFX, part of the Universal Information Service Browser (UISB) project[63], is a set of RDF plugins for the Eclipse platform[64]. It provides a framework to manipulate RDF which includes a SPARQL editor. Baker and Boakes (2004) describes the use of RDFX and UISB and the role of provenance. They define their own provenance ontology[65] based on the RDF Reification vocabulary.

2.3.4.5 Named Graphs

TRIPLE, by Sintek and Decker (2002) adopts a Named Graph approach; however, it incorporates data representation and Horn-clause logic in the same syntax (Figure 2.4). It is intended as a rule language supporting applications that require RDF reasoning and transformation under different semantics. Its use of Horn-clause logic means it can be enacted by Prolog systems.

```
@dfki:document {
    dfki:d 01 01 [
      dc:title → TRIPLE
      dc:creator → Michael Sintek;
      dc:creator → Stefan Decker;
      dc:subject → RDF;
      dc:subject → triples; ...].
  ∀S, D search(S, D) ←
      D[dc:subject → S].
}
```

FIGURE 2.4: Example TRIPLE Syntax with Dublin Core

Carroll et al. (2005) note that imposing a single way to implement RDFS and OWL semantics with Horn-rules should be seen as a weakness.

[63] http://dsg.port.ac.uk/projects/uisb/.
[64] http://www.eclipse.org/.
[65] Available at http://rdfx.org/schema/2004/06/09-prov.rdf.

2.3.4.6 RDF Molecules

RDF molecules are an alternative method for RDF graph decomposition [Ding et al. (2005)] and provenance attribution. RDF molecules are described as sub-graphs of their parent graph that can be used to track provenance without loss of information. The argument set out in Ding et al. (2005) claims that provenance tracking at the document level yields too few matches, whilst the triple level has issues with blank nodes. The apparent granularity of RDF molecules places them between triples and Named Graphs (Section 2.3.4.5).

2.4 Logic Frameworks

2.4.1 Description Logics

Description Logics (DL) are a family of knowledge representation (KR) languages that represent knowledge in a problem domain. DLs are comprised of four main components: the Class Expression Language (CEL) which defines the logic; the TBox which defines the ontology; the ABox which contains instances of an ontology; the RBox which defines relationships between roles. In a DL knowledge-base, the TBox and RBox represent *intensional knowledge*, i.e., general knowledge. Superficially, DL shares a lot of the concepts found in object orientated languages.

DLs are a decidable fragment of First Order Logic (FOL). DLs exhibit high expressivity together with decidability, which guarantees that a reasoning algorithm will always terminate, with the correct answer [Artale and Franconi (1999)]. DLs form the basis for describing domains of knowledge, often in the form of an ontology (see Section 2.5.3). The Semantic Web Language, OWL [Bechhofer et al. (2004)] is split into three sub-languages: one, probably the most popular, is a DL.

2.4.1.1 Class Expression Language

The Class Expression Language defines the logical concept constructors used in the DL ($\cap$, $\cup$). The CEL also gives rise to a wide range of names to DLs that include: $\mathcal{ALC}$, $\mathcal{SHIF}$, $\mathcal{SHIQ}$, $\mathcal{SHOQ}$ [Horrocks and Sattler (2001)], $\mathcal{SHIOQ}$, $\mathcal{SHION}$. These DLs can be characterised as follows:

- $\mathcal{ALC}$

  $\mathcal{ALC}$ represents Attribute Logic Complement [Baader and Nutt (2002)] that includes Conjunction, Universal Value Restriction, and Limited Existential Qualifications. Modern DL languages denote $\mathcal{ALC}$ by $\mathcal{S}$ when describing more expressive languages.

- $\mathcal{H}$ Role Hierarchy

  $\mathcal{H}$ introduces Role Hierarchies for general TBoxes (see Section 2.4.1.2).

- $\mathcal{I}$ Inverse Roles

  Inverse Roles are useful when representing opposing relations such as *replaces* and *isReplacedBy*.

- $\mathcal{F}$ Functional Roles

  A Functional Role is a role that can have only one (unique) value y for each concept instance x, i.e., there cannot be two distinct values $y1$ and $y2$ such that the pairs $(x, y1)$ and $(x, y2)$ are both instances of this role. A common example of a $\mathcal{SHIQ}$ DL is OWL Lite (see Section 2.5.3.4) where both object properties and datatype properties (the *range*) can be declared 'functional'. Figure 2.5 shows another example taken from the OWL Guide [Smith et al. (2004)] that shows the `hasVintageYear` property. By defining this property as being functional, a wine will have a unique vintage year.

```
<owl:Class rdf:ID="VintageYear" />

<owl:ObjectProperty rdf:ID="hasVintageYear">
  <rdf:type rdf:resource="&owl;FunctionalProperty" />
  <rdfs:domain rdf:resource="#Vintage" />
  <rdfs:range  rdf:resource="#VintageYear" />
</owl:ObjectProperty>
```

FIGURE 2.5: Example functional property, taken from the OWL Guide

DL languages can also declare roles as inverse-functional, where the value of the role *uniquely* determines the concept (the *domain*) of an instance. For example, if we define the role *isMotherOf* and declare it inverse-functional for the concept *Mother*, then the value y can only be the value of *isMotherOf* for a single instance of *Mother*; it is not possible for two instances of *Mother* to have the same value of *isMotherOf*. Inverse-functional properties can be seen as equivalent to keys in relational databases.

- $\mathcal{O}$ Individuals

  Up until now each DL only considers concept and role subsumption in the TBox. DLs that include $\mathcal{O}$ also permit ABox reasoning on individuals.

- $\mathcal{Q}$ Qualified Restrictions

  $\mathcal{Q}$ denotes qualified restrictions on concepts and roles. $\mathcal{SHIQ}$, $\mathcal{SHOQ}$ and $\mathcal{SHIOQ}$ are known examples. $\mathcal{SHIQ}$ is supported by the FaCT inference engine, however, as we noted above, individuals are not supported by the absence of $\mathcal{O}$. $\mathcal{SHIOQ}$ adds individual support and forms the basis of DAML-OIL, the predecessor of OWL.

- $\mathcal{N}$ Nominals

  $\mathcal{N}$ adds an unqualified number restrictions (nominals, oneOf) to concepts and roles. Unqualified restrictions are useful when exhaustively enumerating concept instances in lists, for example, names of countries.

  OWL DL (see Section 2.5.3.3) includes $\mathcal{N}$ to form $\mathcal{SHION}$(D) where D represents datatypes (Section 2.4.1.5).

2.4.1.2 The TBox

The TBox (Terminology Box) defines relations between concept names and expressions. Concept names represent *things* in a particular domain of knowledge, for example, **mammal**, **human**, or **computer**. Concepts can be part of a hierarchy forming complex relationships between different concepts.

2.4.1.3 The ABox

The ABox (Assertional Box) is a world description containing *individuals* according to the TBox. Individuals are essentially class instances as defined by the TBox. Consistency checks can be made over the ABox (ABox reasoning)

2.4.1.4 The RBox

The RBox (Relational Box) defines the relationships between roles and the various properties a role might have. In practise, the RBox is not used very often since relations between roles, known as role value maps [Baader (2003)] increase the expressiveness of a DL to a point where it is no longer decidable.

2.4.1.5 Concrete Datatypes

Concrete datatypes are used to represent literal values, for example, numbers and strings. A type system typically defines a set of 'primitive' datatypes, such as string or integer, and provides mechanisms to derive new datatypes from ones that already exist. In the XML schema type system the nonNegativeInteger datatype is derived from the integer datatype by constraining values of nonNegativeInteger to be greater than or equal to zero [Biron and Malhotra (2004)].

2.4.2 Semantic Inferences

Semantic inferences are logical consequences based on a set of rules. Rules are built up from proposition statements such as those found in Horn clause logic [Horn (1951)]. Horn clauses express a subset of statements of first-order logic, where clauses contain at most one positive literal, L:

$$L_1 \dots L_n \rightarrow L$$

Clauses with exactly one positive literal are known as *definite clauses* while clauses with zero positive literals are known as *goals*. *Goal*-based reasoning and Horn clauses form the basis of expert system languages such as Prolog [Colmerauer and Roussel (1992); Covington et al. (1996)].

In DL, GCI (General Concept Inclusion) axioms are similar to rules that allow custom semantic constructs to be included in an ontology. GCIs can be used to complement existing constructs found in class subsumption (subclassOf, intersectionOf) to classify concepts. Unlike Horn clause-like rules, axioms do not have any operational grounding [Baader and Nutt (2002)].

Other rule languages exist that are able to be used for semantic inferences. Table 2.1 lists some rule approaches and includes notes on the expressivity and decidability. OWL DL is included since it is capable of a limited subset of concept-based subsumption that is known to be decidable. Unfortunately, this is not enough for general purpose reasoning that has long existed in expert system languages.

Work during the 1990's saw attempts to create "deductive databases" that could perform Prolog-like inferences with datalog languages that were efficient over persistent stores [Butler (2005)]. The PARKA-DB project[66] was one success, where

[66] http://www.cs.umd.edu/projects/plus/Parka/parka-db.html.

Approach	***Expressivity***	***Decidability***
OWL DL	$\mathcal{SHION}$(D) Tree-like rules	Decidable
Axiomatic (SWRL)	DL with role-value maps Unrestricted rules.	Un-decidable
DL-safe Rules	$\mathcal{SHIQ}$(D) [Motik (2006)] Concepts and roles in head and body of rule. DL-safe rules.	Decidable
AL-log	$\mathcal{ALC}$ Concepts in body of rule.	Decidable
CARIN	$\mathcal{ALCNR}$ Concepts and roles in body of rule [Levy and Rousset (1996)]. Role-safe rules.	Decidable for non-recursive rules
Intersection	Some constructors of DL. Variants include *if* constructor, occurs on left or right hand side of the rule. Unrestricted rules.	Decidable

TABLE 2.1: Comparison of Rule Language Approaches

researchers integrated a knowledge-base with a relational database management system (RDBMS) [Evett (1994); Stoffel et al. (1996)].

2.4.2.1 Types of Inferences

Rule-based systems generally operate in one of two modes: backward and forward chaining. Execution strategies for each mode are distinctly different, relying on different sets of algorithms to improve performance.

Backward Chaining Backward reasoning or Logic Programming (LP) is a common inferencing strategy. Used in Prolog systems, backward reasoning is goal-orientated where knowledge-bases are searched to support predetermined conclusions. In fact, Prolog is constructed entirely out of Horn clauses:

$$A \leftarrow B_1 \dots B_k$$

where A is a fact, $B_1 \dots B_k$ are goals, the left-hand-side (LHS) of the clause is the head (consequent) of the rule, and the right-hand-side (RHS) of the rule is known as the body (antecedent).

Backward reasoning engines tend to be more efficient compared to their forward counterparts. Given a sufficiently expressive logic, however, it is possible for backward reasoners to fail to terminate.

Forward Chaining Rather than seeking data to support a goal, forward reasoning involves drawing conclusions by matching rules to an underlying dataset. Rule matches are incremental which means rules can be triggered continuously as new knowledge is added to the dataset. Incremental updates are particularly useful in expert systems where use is made of previous knowledge. RETE [Forgy (1982)] is a well known algorithm used in many forward rules engines, for example, Jena 2 and Jess [Friedman-Hill (2003)].

2.4.2.2 Popular Inference Engines

Tools such as the Closed World Machine (CWM)[67], Jena, Redland[68], Pellet[69], FaCT[70] [Horrocks (1998)], and Racer [Haarslev and Mller (2003)] are being used by people to process RDF using rules. Unfortunately, despite the submission of the Semantic Web Rule Language (SWRL), no standards activity has been started [Butler (2005)].

Jena 2 provides a family reasoners that support RDFS and OWL entailments, and a generic reasoner that allows developers to define their own rules. The generic reasoner has both forward (RETE algorithm) and backward (Logic Programming) engines. Other rule-based expert systems engines include the C Library Integrated Production System (CLIPS) [Giarratano and Riley (1993)] and Jess. Jess in particular has been used as an OWL reasoner in OWLJessKB[71].

2.4.3 Frame Logics

Frame Logic (F-Logic) is an object-orientated approach to knowledge representation that extends Prolog. [Kifer et al. (1995)] describes the formal aspects of F-Logic including its relationship to frame-based languages found in Artificial Intelligence (AI) [Minsky (1981); Hayes (1979)]. It is interesting to note that while

[67] http://www.w3.org/2000/10/swap/doc/cwm.html.
[68] http://librdf.org/.
[69] http://www.mindswap.org/2003/pellet/index.shtml.
[70] http://www.cs.man.ac.uk/~horrocks/FaCT/.
[71] http://edge.cs.drexel.edu/assemblies/software/owljesskb/.

F-Logics take their name from frame languages, they do not use the same terminology since they are primarily object-orientated.

F-Logic formalisms such as the Web Service Modelling Ontology (WSMO)[72] and its concrete syntax, the Web Service Modelling Language (WSML)[73] are becoming popular as description frameworks for Semantic Web Services. Considerable effort in WSMO is being made to align F-Logic alongside Datalog, Logic Programming and even DL for the purposes of automated web service composition, discovery and invocation.

2.4.4 Open and Closed World Semantics

Logic-based systems fall into two kinds of semantics: Open World and Closed World. Open World semantics assumes that absence of knowledge means that it is not currently known; this means a knowledge-base is perhaps incomplete. Any negation is due to unsatisfiability, i.e., something is false only if it can be proven to contradict other information in the knowledge-base [Rector et al. (2004)]. Formally, given a knowledge-base $\sum$, for every formula φ, if $\sum \nvDash \varphi$ and $\sum \nvDash \neg\varphi$, the answer is *unknown*.

If we were to assert that John *is-a* Human, and Human *is-a* Mammal the fact that the transitive closure John *is-a* Mammal does not yet exist means that it is unknown for the time being. While logics based on Open World semantics might look like a giant jigsaw puzzle, their incompleteness makes them expandable since there is normally[74] new knowledge to be asserted. The one major disadvantage of Open World systems is that the more they grow, the more computationally complex semantic inference becomes.

Closed World semantics follows the principle of *Negation as Failure* (NAF). If knowledge is not present in a knowledge-base then it cannot be 'true', as the knowledge-base is assumed to be complete. Formally, given a knowledge-base $\sum$, for every formula φ, if $\sum \nvDash \varphi$, then $\neg\varphi$.

A classic example of a Closed World system is the relational database. Queries for non-existent information will always return false. A Closed World system has complete control and knowledge of its state; as such nothing is unknown.

[72] http://www.wsmo.org/.
[73] http://www.wsml.org/.
[74] Some small logics, e.g., prepositional, are complete.

This does mean, however, that Closed World systems are less scalable than Open World systems. For example, it is rather difficult to expand a relational database schema (tables, columns). Adding new rows is trivial although it may invalidate previous results. The strict control of information in a relational database also means information is less likely to become inconsistent or contradictory.

Contradictory facts are possible in Open World systems. Given a knowledge-base $\sum$, for every formula φ, there may be instances where $\sum \vDash \varphi \cdot \sum \vDash \neg\varphi$ hold. Critics of the Semantic Web often cite contradictions as a flaw. This may be true, but only given our current knowledge and understanding of open systems [Hewitt (1985)].

The Semantic Web by necessity operates under Open World semantics. Not only does Open World semantics maintain the scalable nature of the WWW, it also reflects the 'incomplete' nature of the WWW.

2.4.5 Monotonicity

Logics can be either monotonic or non-monotonic. The differences between the two approaches are subtle; these differences can, however, have profound effects on the interpretation of a knowledge-base and inferences.

Monotonic Logics state that all assertions in a knowledge-base are considered 'true' irrespective of whether the statement has factual grounding. If we asserted that "pigs can fly" into our knowledge-base that was a monotonic logic, then the statement would be true despite the laws of physics stating otherwise. An important feature of monotonic logics is that the assertion of new knowledge never negates existing knowledge, although it is possible for contradictions to appear. This would imply that the Semantic Web would always grow and never shrink.

A non-monotonic logic permits the *loss* of knowledge. While assertions may infer new relations in a knowledge-base, they may also negate knowledge and thus reduce the size of the knowledge-base. Classic expert systems are predominantly non-monotonic; Horn-clause rule engines can remove 'old' assertions as as necessary.

2.5 World Wide Web Technologies

2.5.1 URIs

Uniform Resource Identifiers, names for web-based resources can be split into two categories: URLs and URNs.

- URL

  A Uniform Resource Locator is the only standard existing today for identifying information resources on the Internet. The URL is an abstract, human readable address to the location where a specific resource can be found, specifying the protocol with which to retrieve the resource. URLs invariably become unstable references to a resource. URLs can also be accessible through different protocols, part of the URL, which can cause headaches and confusion for the casual web user.

- URN

  The Uniform Resource Name is an attempt to solve the problems of URLs, especially in terms of longevity. A URN is a non-human readable string that use namespaces to identify how the URN should be handled. There are numerous registered URN namespaces, however, it is possible to create URN namespaces for the purpose of an application or system where URI preservation was important.

2.5.2 Description Frameworks

2.5.2.1 Resource Description Framework

RDF [Klyne and Carroll (2004)] provides a way to describe relations between WWW resources as a graph where the arcs are represented by XML Qualified Names (QNames) [Bray et al. (1999)], an alias for a URI and nodes are represented by QNames, local names, blank nodes and (typed) literals (see Figure 2.6). RDF records these relationships as (subject, predicate, object) triples (see Figure 2.7). The RDF recommendation defines how we may merge a set of graphs into one, while the formal semantics [Hayes (2004)] defines the meaning of triples and basic entailments.

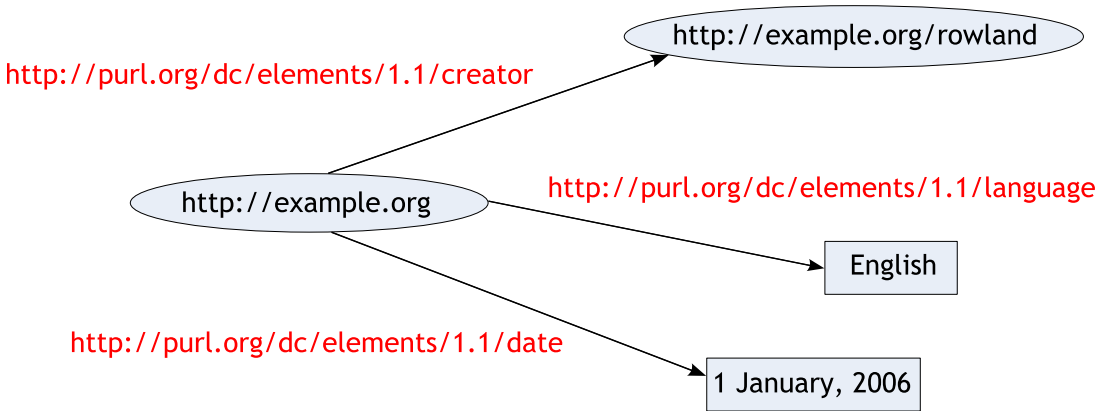

FIGURE 2.6: An RDF Graph

Resources in RDF are uniquely identified by URIs (see Section 2.5.1). The subject and predicate of a triple *must* be a URI, whereas the object can be either a typed literal (XML Datatypes) [Biron and Malhotra (2004)] or a URI. There are some who argue that literals as subjects should be allowed [Carroll et al. (2005)], however, due to constraints made in the RDF/XML concrete syntax, literals as subjects are not permitted.

RDF triples form binary relationships in a graph which makes RDF semi-structured. New binary relations can be added at will without restrictions. As we shall see in Section 2.5.3, ontologies give frameworks like RDF a structure based on formal logics.

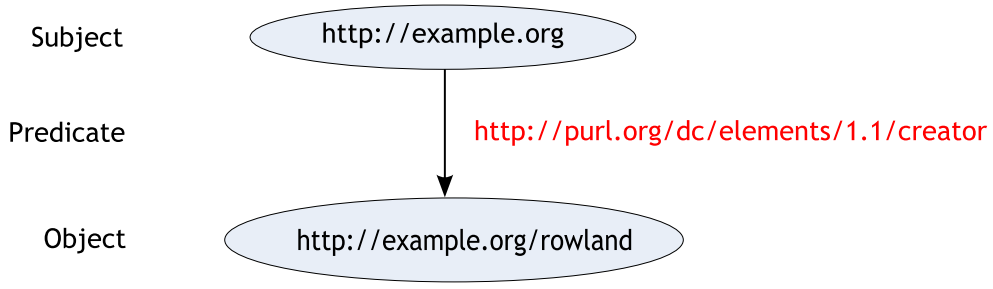

FIGURE 2.7: An RDF Triple

2.5.2.2 Topic Maps

Topic Maps is an ISO standard[75] for describing WWW resources, similar to RDF[76]. While *Topic Maps* appear to be in direct competition with RDF, it does support ontology building, using OWL as its ontology language of choice.

[75] http://www.topicmaps.org/xtm/1.0/#ref_iso13250.
[76] http://www.topicmaps.org/.

2.5.3 Ontologies

An ontology is "a specification of a conceptualisation" [Gruber (1993); Noy and McGuinness (2001)]. Ontologies are useful to us because they explicitly define relationships or *roles* between abstract concepts. In DL, ontologies are referred to as vocabularies or *Tboxes* (see Section 2.4.1.2). *Instances* (the Abox, Section 2.4.1.3) of a given vocabulary can be supplemented by inferencing: the creation of new relations by implicit relationships defined by a rule set.

2.5.3.1 RDFS

RDF Schema (RDFS) adds basic structure to RDF and the beginnings of an ontology language. It allows URIs to be labelled so we can give them human readable interpretations. RDFS adds the following to RDF:

- Class/SubClass relationships
- Instances
- Properties (relations)
- Multiple inheritance

It is important to note that RDFS is meant as an ontology language for use in logical reasoning; unlike XML Schema it is not meant for validation.

Class/SubClass declaration Class and subclass declarations allow basic concept taxonomies to be developed. Class declarations represent concepts that can be formed into controlled vocabularies.

Instances An instance is a particular realisation of a class, similar to an individual in DL. Class instances can be checked for consistency against an RDFS ontology and classified against the class hierarchy.

Properties (relations) Properties (roles in DL) form the basis for relationships between classes in an ontology. Properties *may* be restricted by declaring a domain and range. The domain specifies which classes have a particular property; the range specifies the kinds of values (classes, datatypes, etc.) a property can have. RDFS supports only simple properties and as such does not support datatypes (see Section 2.4.1.5).

Multiple Inheritance Multiple inheritance is a well known feature of object orientated programming languages. While some well known programming languages such as Java™ forbid multiple inheritance, knowledge representation languages like RDFS do not. RDFS permits multiple inheritance for both classes and properties.

2.5.3.2 OWL

OWL builds upon RDFS and expands the vocabulary of possible constructs. OWL provides sub-languages with reduced expressivity and computational complexity. Ontologies that import an RDFS ontology fall within OWL Full. Two smaller languages exist: OWL DL and OWL Lite.

OWL Full's rich expressiveness means that it is computationally expensive; it is not decidable. As a result, OWL Full tools are difficult to build.

2.5.3.3 OWL DL

OWL DL, based on Description Logics and equivalent to $\mathcal{SHION}$(D), adds numerous restrictions on OWL constructors including classes and properties. The reduction in expressivity means OWL DL is decidable.

OWL DL includes class restrictions so that, for example, a class cannot be an instance of another class; metaclasses are therefore forbidden. Property constructs such as, FunctionalProperty and InverseFunctionalProperty cannot be used with datatypes, they can only use used with the ObjectProperty construct.

2.5.3.4 OWL Lite

OWL Lite inherits all the restrictions of OWL DL and introduces its own. While OWL Lite is the least expressive sublanguage of OWL, it is the most tractable and several Semantic Web toolkits support it, for example, Jena, Pellet, and FaCT.

In addition to OWL DL restrictions, OWL Lite forbids owl:minCardinality and owl:maxCardinality while owl:cardinality may only be 0 or 1. owl:hasValue, owl:disjointWith, owl:oneOf, owl:complementOf and owl:unionOf are all forbidden.

2.6 Summary

This chapter summarised a range of topics relevant to the research described in this thesis. In later chapters we will explore in more detail some of the concepts and technologies mentioned in this chapter.

Key to this thesis are questions relating to the relative benefits of the RDBMS compared to recent advances in Semantic Web technology. In the next chapter we will investigate our two case studies and analyse the suitability of RDBMS and Semantic Web technology in version control for distributed collaborative software development.

Chapter 3

Analysis

In the previous chapter we presented a general overview of the background concepts and technologies relevant to this work. This chapter concentrates on how version control is used in two collaborative development case studies and goes on to analyse the various issues relating to the use of contemporary version control systems. We then go on to analyse the potential use of Semantic Web technology, the role of provenance, and our motivation for RDF digital signatures.

3.1 Distributed Collaborative Software Development Case Studies

Distributed collaborative software development is typically characterised by medium to large scale projects with development partners disparately located, often in separate domains of trust. Here we concentrate on two examples of distributed collaborative software development: Free, Libre, Open Source Software (FLOSS) [Di-Bona et al. (1999, 2005)] and European Community (EC) Information Society Technology (IST) projects [EC-IST (2006b,a)].

3.1.1 FLOSS

FLOSS development, once restricted to university campuses, has now become a major movement in the software development community. It is at the heart

of the GNU[1] philosophy, particularly the GNU/Linux kernel[2]. In recent years numerous commercial companies have started to contribute to FLOSS projects including Sun Microsystems, IBM, HP, and Apple Computers. Such contributions and commitment shows the viability of FLOSS in various business models.

The vast majority of FLOSS projects, however, do not enjoy financial backing and therefore rely on the contribution and goodwill of developers around the world. Projects tend to have a small core development team who control the development and integrate contributions. The number of contributors can vary, but often be in the thousands within large projects. Since core developers and contributors spread throughout the globe, FLOSS development is truly decentralised in its structure. Management meetings will invariably be conducted in online chatrooms, for example, Internet Relay Chat (IRC).

Whilst FLOSS development is laudable in its efforts to produce software that is freely available for anyone to use and modify as they wish[3], the quality of such software is questionable. FLOSS projects rarely attract financial support, receiving contributions from amateur and professional programmers on a *best effort* basis. As a result there is very little accountability in the production of code since developers are not bound by contractual obligations, nor are their contribution guaranteed to work. The vast majority of FLOSS licences include a statement that states that the software is provided *"as is" and without warranty.*

FLOSS projects, whilst decentralised in terms of social structure, typically use a centralised repository for storing code. This is one of the predominant reasons for the checkout-modify-commit model found in most version control systems. Developers will checkout a local copy of the repository, hack code until they are satisfied, then commit changes back into the repository, making merges as necessary. Source Forge[4] is a good example of a freely available, centralised, hosted FLOSS development repository. Other examples include GNU Savannah[5], Tigris[6], BerliOS[7].

[1] GNU is Not UNIX®.
[2] http://www.kernel.org/.
[3] GPL, LGPL, BSD, Apache, and Creative Commons are some examples of FLOSS licences that permit modification and redistribution of *derivative works*. All these licences are available at http://www.opensource.org/licenses/.
[4] http://sourceforge.net/.
[5] http://savannah.gnu.org/.
[6] http://www.tigris.org/.
[7] http://www.berlios.de/.

The centralised approach to FLOSS project management is convenient for most developers, even though it means that there is less emphasis on accountability and trust, since most host repositories are support on a "best efforts" basis. Project administrators have the power to do what they like to projects with no responsibility to the wider public.

3.1.2 EC IST Grid Collaboration

Recent years have seen an increase in the funding of Grid projects under the European Community (EC) Information Society Technology (IST) Framework programme. Projects that have been funded in the past include Fifth Framework projects including GRIA[8], EGEE[9], UNICORE[10], and more recently Sixth Framework projects such as SIMDAT[11], NextGRID[12], Akogrimo[13], CoreGrid[14], Edutain@Grid[15], BREIN[16], ArguGrid[17], BEinGrid[18], and BRIDGE[19]. Each project consortium is formed from various academic and commercial partners under contract to produce novel and *commercially exploitable* products, based on Grid technology. Each partner is a stakeholder in the project, however, commercial partners often have more risk since they must contribute to their budget costs.

The size of a consortium varies and can increase over the course of a project; an EC project with eight partners can expect to contribute tens rather than hundreds of developers. Each partner leads a Work Package (WP) which concentrates on a particular aspect of the project, which shows that not only are EC projects heavily centralised, but also that there is a clear understanding of responsibility and accountability between partners.

One major *feature* of EC funded projects is the management process that the project consortium must adhere to. The EC requires regular progress reports, deliverables and annual review meetings, all of which are outline in the Description

[8] http://www.gria.org/.
[9] http://public.eu-egee.org/.
[10] http://www.unicore.eu/.
[11] http://www.scai.fraunhofer.de/simdat.html.
[12] http://www.nextgrid.org/.
[13] http://www.mobilegrid.org/.
[14] http://www.coregrid.net/.
[15] http://www.edutaingrid.eu/.
[16] http://www.gridsforbusiness.eu/.
[17] http://www.argugrid.org/.
[18] http://www.beingrid.com/.
[19] http://www.bridge-grid.eu/.

of Work (DoW) negotiated before the project starts. Each partner is contractually obliged to contribute to deliverables and send them to the EC in a timely manner. Annual review meetings with the EC can include a panel of experts whose job it is to analyse the originality and novelty of a project, and can provide recommendations that should be taken into consideration.

Whilst development in EC IST Grid projects is distributed in nature, due to the potential commercial exploitation of developed software, each partner will normally develop in a private source code repository. Software integration phases require more collaboration between partners with one partner responsible for producing an integrated prototype.

Intellectual Property Rights (IPR) and confidentiality issues mean that most consortium members will (initially at least) work on separate repositories then permit controlled access (often licenced) to source code or binary distributions. The EC, whilst keen on reaping the benefits of funded projects in the form of FLOSS understand the need for industrial partners to close source certain project results for the purposes of exploitation. In most cases companies will deliver individual confidential exploitation plans to project officers and EC representatives.

Both of the above case studies have particular requirements in terms of the type of version control system that should be used for effective collaboration. As we have already noted (see Section 2.2.2), the vast majority of version control systems are based around RDBMS technology with some form of remote access protocol. Only GNU Arch, Git and depart from this, relying on the underlying file system as the storage mechanism. While these newer systems are of interest, they are used in a community far smaller than that supported by Source Forge and others.

As we stated in Chapter 1, one of the motivations in this work is to test the suitability of Semantic Web technology in the same problem domain. Before doing this, however, we need to analyse the relative benefits and issues of both approaches, particularly with regard to data federation, trust, scalability, interoperability, as well as server and metadata integrity.

3.1.3 EC IST Collaboration with FLOSS

The two case studies described above are not isolated from one another. The vast majority of EC IST projects include industrial partners to help improve the exploitation of project results. While the EC understands that industrial partners

want to individually exploit results, there is an increasing emphasis for FLOSS exploitation.

Large organisations are increasing finding that FLOSS products are cost effective and are bringing value to the enterprise. In the past companies had to develop their own operating systems for specialised products, but can now reuse the GNU/Linux kernel. IBM, Novell, Sun, Apple Inc., and many other industry leaders supply at least one FLOSS product.

Although FLOSS adoption is argued to be a good thing, companies must be careful how they incorporate FLOSS in their product development pipeline. FLOSS licences must be carefully analysed so that companies can leverage in their products. For example, most companies will not be willing to incorporate GPL-based code since *derivative products* must also be GPL. The *Lesser* GPL (LGPL) is far more attractive since proprietary code can link to it without having to be similarly licensed.

3.2 RDBMS Approach

Early version control systems such as Revision Control System (RCS)[20] worked on flat file systems which made them rather slow and cumbersome to use. File metadata and diffs were kept in the same logical structure making them more difficult to manage over time, often leading to the possibility of inconsistent metadata in the event a failed commit.

The use of an RDBMS in version control is a fairly old fashioned approach but predominant method, despite the exponential growth of open standards that make up the WWW. More recent systems including CVS, Subversion, and Git are examples of the RDBMS approach, all of whom are used extensively in FLOSS projects. For example Source Forge supports both CVS and Subversion repositories for users.

[20] http://www.gnu.org/software/rcs/.

3.2.1 Benefits

RDBMSs form the basis of many successful businesses (Amazon[21], Google[22], Ebay[23]) and enterprise technologies (Java™2 Enterprise Edition[24], Microsoft .NET Platform[25]) which gives an idea of the maturity of the technology. As their maturity has increased, so has their speed and availability. The vast majority of modern operating systems now ship with some form of RDBMS (Microsoft SQL Server[26], MySQL[27], PostgreSQL[28]).

3.2.1.1 Interoperability

Most modern RDBMSs can be access remotely for administration purposes, and more importantly remote querying of databases; the vast majority of RDBMSs support the Structured Query Language (SQL) or a major dialect. Recent standardisation of data access in web services has led to the OGSA-DAI (Data Access and Integration) [Antonioletti et al. (2003, 2005)][29] standard.

In more recent version control systems, version metadata is stored in an RDBMS rather than in flat files. By keeping the metadata distinct and separate from the documents under version control, modern communication protocols can then be used for generic access, for example, HTTP and WebDAV. The use of standardised protocols has obvious interoperability advantages over proprietary protocols used in early version control systems.

3.2.1.2 Performance and Scalability

The maturity of RDBMS technology means that the majority commercial and FLOSS products are high performance and very scalable. Performance is normally accomplished using multi-threading for query processing. More advanced concepts such as transactions and two-phase commits also improve performance and reliability.

[21] http://www.amazon.co.uk/.
[22] http://www.google.co.uk/.
[23] http://www.ebay.co.uk/.
[24] http://java.sun.com/javaee/.
[25] http://www.microsoft.com/net/.
[26] http://www.microsoft.com/sql/.
[27] http://www.mysql.com/.
[28] http://www.postgresql.org/.
[29] http://www.ogsadai.org.uk/.

Scalability can be achieved by server replication, however, version control systems have yet to take advantage this. Repository hosts like SourceForge typically replicate repositories to maximise availability which is appropriate for serving FLOSS projects. IST projects on the other hand are small enough that scalability is not a core requirement.

3.2.2 Issues

Although RDBMS technology has number benefits as outlined above, it is not the panacea for version control systems. Remote access to servers has its limitations in terms of reliability and security. Repository federation is also be problematic due to the data model and transport issues.

3.2.2.1 Federation

Federation is a challenge in an RDBMS environment. Challenges range from differences in data models to transport protocol interoperability between organisations wishing to collaborate. Java™2 Enterprise Edition (J2EE)[30] has been one attempt to ease interoperability issues, defining a set of standards that helps developers to abstract access to databases using Enterprise Java Beans and transport them across enterprises using Remote Method Invocation (RMI) [Grosso (2001)] in a reliable manner with transactions [Baksi (2001)]. While J2EE has been widely accepted by industry, many developers have criticised its stack as being "heavy-weight", over complicated [Tate and Gehtland (2004)] and too closed since it is developed under the Java Community Process (JCP)[31]. Some companies, including IBM, HP, Intel and Microsoft are moving toward a more decoupled, SOA approach using web service based around extensions to WS-Transfer [Cline et al. (2006)] including WS-ResourceTransfer (WS-RT) [Reistad et al. (2006)].

Repository federation is a highly desirable feature for IST project collaboration, especially during integrated prototype phases. Contributing partners can allow remote access to the integrating partner, who then federates each repository to construct the integrated prototype. At present, the vast majority of projects either copy all necessary code into a new central repository or settle with binary distributions from contributing partners.

[30] http://java.sun.com/javaee/.
[31] http://jcp.org/.

Newer web service-based protocols such as OGSA-DAI offer another approach to realise federated queries. It is unlikely, however, that web services will find their way into version control systems any time soon due to performance and scalability. SOAP adds additional overheads (XML processing) that do not exist in custom protocols used in SVN and Git.

3.2.2.2 Trust Management

Trust management is a significant issue when it comes to RDBMS-based version control repositories. In FLOSS projects, source code is typically held in one or more repositories, stored on a single server[32]. This server must be *explicitly trusted* by all developers to reliably store and protect the contents of each repository. Unfortunately, in most cases the server is *implicitly trusted* without the developer having the opportunity to analyse the accountability mechanisms available at the hoster. In the vast number of incidents when a server is compromised, the entire server can no longer be considered reliable and must be rebuilt [Kemp (2006)]. In most cases a complete rebuild is necessary which not only takes time but can cost a project financially. Trust in centralised hosting environments is not limited to the integrity of the machines used for storage and processing; owners of projects and system administrators must also be trusted not to abuse their positions.

In the case of EC IST projects, trust management is essential. Each partner will have a repository within their own trust domain, accessible to other partners within the terms of the Consortium Agreement. As such, all source code must have appropriate copyright attribution so that other partners understand the *provenance* of third-party code. Any version control system used should be able to provide facilities to enforce copyright attribution to maintain IPR.

Some version control systems including GNU Arch and Git attempt to increase commit trust with SHA-1 hashes, and *optionally* GNU Privacy Guard (GPG)[33] or Pretty Good Privacy (PGP)[34] signatures. The use of SHA-1 digests is a welcome improvement, although recent discovers call into question its ability to resist collisions [Wang et al. (2005b,a)]. The use of digital signatures is also a good development, unfortunately this is appears to be only optional. Unless developers have had previous experience in the use of PGP, GPG, or any other PKI they are

[32] More advanced server configurations like those hosted at Source Forge include a certain amount of replication to increase availability.

[33] http://www.gnupg.org/.

[34] http://www.pgpi.org/.

less likely to take advantage of digital signatures. To truly improve commit trust in these systems, digital signatures must become an integral part of the version control workflow.

As with most secure systems that employ advanced cryptography, in a truly distributed collaborative environment, it is not enough to simply sign the commits and trust the server to handle them correctly. A *third-party* server must always be explicitly trusted rather than implicitly trusted as the current model with Source Forge; if the server becomes compromised, signatures can easily be ignored, source code could be modified, at worst lost, to the detriment of the project. By making trust decisions explicit together with digital signatures, it is then possible for repository integrity to be vested in the repository metadata itself.

3.2.2.3 Interoperability

Despite the existence of standardised query languages, RDBMS approaches tend to suffer from data format interoperability issues. Subversion, for example is promoted by its authors as the successor to CVS [Nagel (2004)]. This has perhaps led to a problem whereby the developers view Subversion as a gold plated CVS rather than a new repository [Collins-Sussman (2004)]. It adds more mainstream networking capability (HTTP, SSL/TLS [Dierks and Allen (1999)], and WebDAV), introduces atomic commits, and the ability to be accessed via WebDAV. Subversion implements a strict subset of Delta-V; however, Subversion's authors stress that this does not make it Delta-V compliant, which opens Subversion to interoperability problems.

It is a common problem that CVS, SVN, Git, etc. all have different metadata formats, incompatible with one another. To get around this problem each have import programs that allow developers to import from one type of repository into another. This can be cumbersome and unreliable for large repositories; any mistakes or import failures can cause data loss in large parts of a repository. Fortunately, access to actual source code is better than in the past, since many repositories support HTTP extensions such as WebDAV, rather than simple custom protocols used in CVS.

3.3 Semantic Web Approach

There is promise in bringing Semantic Web technology to distributed version control. Both cases studies outlined in Section 3.1, whilst differing in method and motivation, both have similar needs since both employ distributed collaborative software development. The Semantic Web is being pushed as a means for data federation, knowledge sharing, and trust analysis among others. Whilst take up has been reasonable, is still difficult to determine how successful it has been.

We have already argued that version control systems based on an RDBMS, whilst mature and fast, do not provide the facilities required by our case studies. Part of our research question in this thesis (Section 1.3) is to discover the relative benefits of Semantic Web technology over the RDBMS to further version control.

3.3.1 Benefits

One of the first and foremost advantages of Semantic Web technology is that it is built upon existing, mature, and standardised protocols. The use of URIs (see Section 2.5.1) to label nodes in RDF graphs and HTTP to retrieve serialised RDF is a distinct bonus since development and more user-centric tools are trivial to implement. At least at the syntactic level, Semantic Web technology has a high degree of interoperability with existing web tool kits.

3.3.1.1 Federation

RDF Semantics [Hayes (2004)] explains how to merge RDF graphs from multiple sources. This is important if we want to federate repositories based on a common ontology. Significant work has already been done in this area, where Semantic Web technology has been used to map native SQL databases onto a common ontology, that can then be queried by RDQL or even SPARQL [Bizer and Seaborne (2004)]. Extensions to this work now include several service-based implementations such as R2O [Barrasa et al. (2004)] and D2R [Bizer and Cyganiak (2006,?)], based on the Joseki RDF Server [Seaborne (2003)].

Even simple RDF data federation provides natural, obvious results at the crudest level using semantic query languages. SPARQL is fast becoming the standard

Semantic Web query language[35] which has a rich feature set, including the ability to distinguish the originating graph of a triple.

3.3.1.2 Trust Management

As we noted earlier, RDBMS-based version control systems have poor trust management in server-based deployments. Far too little effort is made to maintain the integrity of metadata stored on the server, not to mention disregard for the human factor of secure systems. Once data is federated across distributed servers, especially in different domains of control, server integrity becomes crucial. For example, project partners in an EC IST project must have confidence when accessing a remote partners' software repository, that the repository's integrity has been maintained and appropriately licenced. Loss of remote repository integrity can leave dependent partners open to IPR contamination.

Trust management in a Semantic Web environment should involve the integration of digital signature technologies with RDF. Some recent approaches to trust management include the use of policies [Dimitrakos et al. (2001)], trust metrics [Golbeck and Hendler (2004a,b)], and several web service drafts, for example, WS-Trust [Anderson et al. (2005)] and WS-Federation [Bajaj et al. (2003)]. In addition to providing message integrity, there needs to be a mechanism for managing RDF graphs that have been digitally signed.

3.3.2 Issues

There are several issues related to the use of Semantic Web technologies; some of these issues are due to the relative immaturity of available toolkits for developers, others are to do with the underlying logic that forms the foundations of the Semantic Web. Issues of particular interest include scalability, provenance mechanisms, semantic interoperability and performance.

[35] Latest W3C Working draft, 4 October, 2006: http://www.w3.org/TR/2006/WD-rdf-sparql-query-20061004/.

3.3.2.1 Performance and Scalability

While Semantic Web is in principle capable of effective data federation, it is not clear how scalable and *efficient* this federation can be. In effect, storage mechanisms for storing RDF should scale in a similar manner to the WWW itself. Semantic Web toolkits at present, however, are not very mature, unlike their RDBMS counterparts with the result that performance is non-optimal. Most toolkits support both flat files as well as persistent storage in an RDBMS. Unfortunately, schemata for persistent storage is far from optimal with the result that queries are not as fast as native SQL queries.

Unfortunately, performance of Semantic Web technology is a real issue that has yet to be improved. Even relatively fast triplestores such as 3Store[36], Sesame[37], and Kowari[38] fail to compare with the performance of an RDBMS[39]. One of the main reasons for this is a lack of decent indexing strategies used to speed up queries.

3.3.2.2 Provenance Mechanisms

RDF Reification Issues Several problems become apparent when we attempt to assert provenance with RDF Reification. The key issue is that the presence of a reified triple in the knowledge-base is unrelated to the presence of the triple itself; thus including the reification does not of itself assert the triple. If we choose to assert each triple as well as its reification, then it is asserted unconditionally and this triple is not bound to the reification.

One major consequence of these problems is that it is difficult, if not impossible, to reason about reified triples. If the reified triple is not bound to an asserted (or not as the case may be) triple, then RDF reification is of no real use in the recording of provenance.

Another consequence is that reification is limited to the triple level; there is no support for making statements about other RDF graphs. Relationships between graphs are desirable if we are to introduce digital signatures into Tim Berners-Lee's Semantic Web stack (Figure 3.1) [Berners-Lee (2005)].

[36] http://www.aktors.org/technologies/3store/.
[37] http://www.openrdf.org/.
[38] http://kowari.sourceforge.net/.
[39] Chapter 5 includes an evaluation of Semantic Web and RDBMS performance.

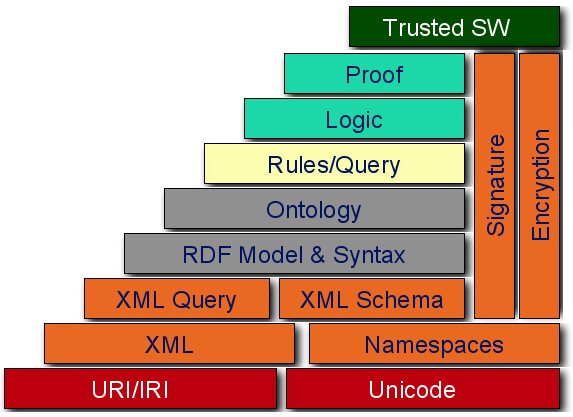

FIGURE 3.1: Semantic Web Stack by Tim Berners-Lee.

MSG and RDF Molecule Issues Although Minimal Self-contained Graphs (MSGs) and RDF molecules appear to be different at first glance, they are rather similar. They both define themselves as methods to decompose RDF graphs, and attempt to define a minimal set of triples that can make up a graph without loss of information. Ding et al. (2005) argue that RDF molecules can be used to *annotate* named sub-graphs, individual triples and other RDF molecules with provenance information. This suggests that only portions of an RDF graph (single triples) can be annotated, not the graph itself. This is very similar to RDF reification where only single triples can be reified to attached additional information (see Figure 2.2).

MSGs follow a similar provenance annotation approach, although they use RDF reification to 'attach' digital signatures to arbitrary triples [Tummarello et al. (2005)]. As we have already noted in Section 3.3.2.2, RDF Reification has several semantic flaws that prevents such digital signature information being used in semantic inferences. Another potential issue is that a digital signature can only be attached to a single reified triple, not a group of triples or graph. While it is relatively simple to detect reified triples, Tummarello et al. (2005) do not specify a method for selecting which triple to reify; arbitrary selection could quite easily lead to loss of the signature when graphs are merged or the triple is de-reified. This is, of course, a signature management problem.

It is difficult to discern the relative benefits of RDF molecules and MSGs. Both approaches claim an advantage over Named Graphs in that they each define the smallest RDF graph to which one might want to attribute provenance. Unfortunately, neither approach explain how to create *relationships* between RDF graphs,

an important feature of Named Graphs. RDF molecules can *name* sub-graphs, however, do not provide an explanation of what this means, nor how it is used.

While the term *sub-graph* is defined in RDF Semantics [Hayes (2004)], the term *named* sub-graph is not. Ding et al. (2005) do not specify the semantics or syntax of a named sub-graph so it is difficult to determine if there is a relationship with Named Graphs; this is unlikely given that Ding et al. (2005) believe Named Graphs of arbitrary size are problematic for provenance.

Semantic Interoperability Issues Wong et al. (2005) argue for a platform-independent framework to validate workflow execution based upon XML-based provenance pioneered by Groth et al. (2004, 2005, 2006); Groth (2005). Wong et al. (2005) defines custom rules in the Semantic Web Rule Language (SWRL) [Horrocks et al. (2004)], executable in the Jena 2 environment.

While defining custom rules in Semantic Web Rule Language (SWRL) is the start of good practice (SWRL is a W3C Member submission) and is supported by a range of toolkits (SweetRules[40]), SWRL is known to be an undecidable logic which can lead to incomplete results. SWRL simulates *role value maps* [Schmidt-Schauss (1989)] which are not available in languages such as OWL DL. Wong et al. (2005) claim the need for *role value maps* for their reasoning use cases, although they do not elaborate how their use of SWRL remains decidable. A Horn-clause based rule language that is DL-*safe* might be more appropriate [Motik et al. (2005)].

The Two Towers of the Semantic Web The Semantic Web's foundations rest firmly on Open World semantics and mono-tonic logics. The wider community is now active building systems based on OWL, particularly OWL DL. Work has started extending DL with rule components, particularly those based on Datalog, Logic Programming (LP) and more recently Description Logic Programming (DLP) [Grosof et al. (2003)].

However, recent proposals such as Katz and Parsia (2005), describing rule extensions to OWL, are likely to cause a substantial rift in the Semantic Web community. Horrocks et al. (2005) effectively demonstrate the problem based on a new version of the Semantic Web stack [Figure 3.2]. They note the following regarding Figure 3.2:

[40] Available at http://sweetrules.projects.semwebcentral.org/.

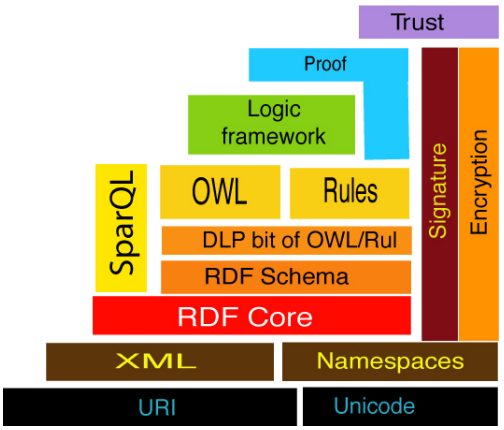

FIGURE 3.2: Latest Version of the Semantic Web Stack.

- Datalog style rule extensions to DLP can only be *syntactic*. Any semantic compatibility would require DLP, and thus OWL to follow Closed World Semantics and effectively become a non-monotonic logic.
- Semantic incompatibility with OWL means DLP cannot be layered between OWL and RDFS. As a consequence this also means DLP is incompatible with RDF.
- Incompatibility between DLP and RDF means a separate Semantic Web could arise, based completely on XML and non-monotonic logics, rather than RDF.

Horrocks et al. (2005) go on to state that it makes more sense to layer DLP and rules on top of XML or *another syntax* and ignore RDF and OWL. Even current efforts in web service description frameworks such as the Web Service Modelling Ontology (WSMO)[41] uses Datalog and LP style rules for various purposes. All these signs point to the conclusion that the Semantic Web community is in danger of splitting into two camps each with diametrically opposed semantics that are completely incompatible with one another. This lack of consensus could hamper future WWW development.

It is interesting to note that this new Semantic Web Stack and its implications is in apparent contradiction to Berners-Lee (2001) and Russell (2003) which regard Open World semantics and monotonic logics as essential to the Semantic Web. Patrick Hayes [Russell (2003)] notes that while he agrees with the inherent monotonicity of the Semantic Web, it is inevitable that other third-parties (software or otherwise) *will* perform non-monotonic inferences over otherwise monotonic logics.

[41] http://www.wsmo.org/.

While future development of the Semantic Web *may* include non-monotonic extensions, current research and toolsets support monotonic logics as best practice. It would therefore be prudent for any ontologies we design to be based on DLs until a satisfactory compromise is found between the Rule and DL researchers.

3.4 Digital Signatures

As we noted in Section 2.2.8.1, a digital signature is cryptographic function that proves the signer of a message possesses the private key associated with the transmitted public key. Successful verification of a digital signature validates the integrity of the message and subsequent messages[42]. If the digital signature fails, then the message has been maliciously or otherwise modified. Digital signatures in a PKI are also useful in identifying the sender, a common form of digital *authentication*.

In secure web services, digital signatures form a crucial part of the WS-Security Basic Profile [McIntosh et al. (2006)] which describes how to secure SOAP messages. WS-Security relies on XML Digital Signature [Bartel et al. (2002b)] and XML Encryption [Bartel et al. (2002a)] to provide integrity and confidentiality to SOAP messages during transit. The sender signs the message with their private key, and *optionally* attaches the public key, usually contained in an X.509v3 certificate [Solo (2002)]. The receiver uses the same public key to verify the integrity of the message; if the key is inside an X.509v3 certificate, the receiver can also verify the identity of the sender against a well known Certificate Authority and form a simple trust relationship.

Other approaches to web service security have taken the ReST approach, but without the need for SSL/TLS [Dierks and Allen (1999)]. HTTPSec[43] is the result of work done by Secarta[44]; rather than securing only the body of an HTTP message, HTTPSec also secures portions of the HTTP header. Asymmetric key exchange protocol for accessing public keys is similar to the public directory structure described by Diffie-Hellman [Diffie and Hellman (1976, 1988)].

[42] WS-Security Basic Profile mandates exactly one digital signature per SOAP message. This has several consequences when it comes to routing SOAP messages through intermediaries.
[43] http://httpsec.org/protocol/1.0/.
[44] http://secarta.com/.

3.4.1 Digital Signatures and the Semantic Web

Digital signatures provide a convenient, yet powerful way to verify the integrity of a message (see Section 2.2.8.1). Standards such as XML Digital Signature describe how to sign XML-based documents as well as arbitrary binary objects in an efficient manner. The Semantic Web, while based on WWW standards, provides a fundamentally different semantic (Description Logic) and syntactic model (RDF graphs) that makes digital signatures more of a challenge. Integrity verification Tim Berners-Lee has argued for some time that digital signatures form part of the solution to trust on the Semantic Web[45].

We can identify three challenges that need to be overcome before digital signatures on the Semantic Web become reality: RDF canonicalisation, semantic interoperability, signature serialisation.

3.4.1.1 Canonicalisation Issues

Unlike XML, RDF does have a canonical form. *Canonical* XML is characterised by the XML Information Set [Cowan and Tobin (2004)] which attempts to guarantee that logically identical XML documents produce identical serialised representations. While Gutmann (2004) argues that Canonical XML is fundamentally broken, XML Digital Signature has successfully used as the basis for various security specifications including WS-Security and SAML [Cantor et al. (2005)].

Before digital signatures in RDF can be realised, it is vital that some form of canonicalisation (C14N) is achieved. Cloran and Irwin (2005) argue that canonical RDF can be broken down into two categories: canonicalisation of the RDF model and canonicalisation of a *serialised* RDF model. We will see later (Section 3.4.1.3) why using a canonical serialisation of the RDF model is not an ideal approach. To our knowledge, at least two algorithms exist for creating canonical RDF models [Carroll (2003); Sayers and Karp (2003)], with only Carroll's algorithm having an implementation in the public domain.

Blank Nodes Blank nodes as defined by the RDF Recommendation [Klyne and Carroll (2004)] are used to label resources not described by a URI. Figure 3.3 shows a fully labelled RDF graph that contains no blank nodes. If we wanted to

[45] http://www.w3.org/DesignIssues/Toolbox.html.

```
<urn:uuid:CA2CAF30-21A8-11DB-8270-9859210973A2> {
   <https://localhost:8443/JSPWiki/Wiki.jsp?page=
          org.embl.ebi.escience.scuflui.workbench.Workbench>
      a       dp:Wikipage ;
      dp:content "description content" ;
      dp:firstVersion <https://localhost:8443/webdav/taverna/
        taverna/org/embl/ebi/escience/scuflui/workbench/Workbench/
        1/1/Workbench.java> ;
      dcterms:created "Tue Aug 01 22:57:55 BST 2006"^^
        <http://www.w3.org/2001/XMLSchema#dateTime> ;
}
```

FIGURE 3.3: A Fully Labelled RDF Graph

digitally sign this graph, we would canonicalise it according to Carroll's algorithm, which would trivially reorder all triples preceded by the graph name.

Figure 3.4 illustrates a more complex example where not all triples in the graph are fully labelled, encapsulated in square brackets. This example happens to represent an RDF collection. The difficulty in this case is when a triple's subject and object are both blank nodes; if several such triples exist then they can become indistinguishable from one another and therefore need to be altered if the graph is to be suitably reordered for signing.

Carroll's solution to this problem is to actually modify the graph with *meaningless changes*, defining a special property **c14n:true** which is always true; this means triples with this predicate can be added and subtracted from the graph without changing its meaning according to RDF Semantics [Hayes (2004)]. This means the RDF graphs digitally signed is different from the original. While Figure 3.4 can be reliably canonicalised (see Appendix C.1.2), the more blank nodes in the graph, the more likely it is for Carroll's algorithm to fail.

If we want to find an extreme example where Carroll's algorithm really does fail, we should consider a complex graph such as the Petersen Graph [Holton and Sheehan (1993)]. Figure 3.5 shows one graphical representation of the Petersen graph with its ten nodes and fifteen edges (An example TriG serialisation can be found in Appendix C.1.2).

Since the Petersen graph, like an RDF graph with only blank nodes, can have many different representations based on its labelling, it can be extremely difficult to determine if two graphs are identical (isomorphic).

```
<urn:uuid:E192F360-226F-11DB-94B3-E05EDA46CF20> {
   wn20schema:NounWordSense
     rdfs:domain
       [ a       owl:Class ;
         owl:unionOf
           [ rdf:first wn20schema:AdjectiveWordSense ;
             rdf:rest
               [ rdf:first wn20schema:VerbWordSense ;
                 rdf:rest () ;
               ] ;
           ]
       ] ;
}
```

FIGURE 3.4: Partially Labelled RDF Graph

McKay (1981) describes a more robust algorithm that solves the graph isomorphism problem [Köbler et al. (1993)] and can therefore cope with blank nodes and reliably relabel the Petersen graph (Figure 3.5)[46]. McKay's algorithm has made an implementation available in the ***nauty*** distribution[47]. While this algorithm satisfactorily creates a canonical representation of an arbitrary graph, unlike the Carroll and Sayer algorithms, it has non-polynomial complexity [Miyazaki (1997)], which makes it less favourable in a Semantic Web environment.

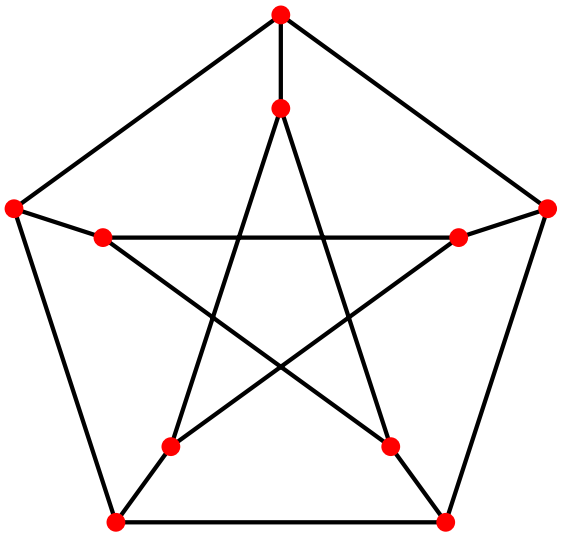

FIGURE 3.5: A Petersen Graph

[46] See Appendix C.1.2 for further details.

[47] *no aut*omorphisms, *yes*?, available at http://cs.anu.edu.au/ bdm/nauty/.

3.4.1.2 Semantic Issues

While DBin's RDF digital signature solution provides a starting point for future implementations, its reliance on RDF Reification as the signature attachment mechanism is problematic. One major problem is that treating a digital signature as a reified statement only applies to that statement, not the graph itself. As we also noted in Section 3.3.2.2, there are also semantic problems.

Because reified triples are not part of the knowledge-base, they are not part of the underlying logic. If we consider an OWL DL knowledge-base with a number of reified digital signatures, basic DL subsumption is not possible over any reified statement. It is also true that any custom GCI axioms or Horn-clause rules would not be able to operate over reified statements. Even Semantic Web toolkits such as Jena 2[48] have to provide a specialised API to access reifications.

3.4.1.3 Serialisation Issues

Dunbill[49] and Cloran and Irwin (2005) both suggest that canonical serialised RDF can be used as the basis for RDF digital signatures. Dunbill's FOAF signatures use PGP, while Cloran and Irwin (2005) take a more interoperable approach with XML Digital Signature.

Signing serialised RDF has the obvious benefit in that it avoids the various canonical RDF issues mentioned earlier. RDF documents and their *detached* signatures (PGP and XML Digital Signature) can be stored on a personal website and verified at a later date. On the other hand, storing an RDF document and its signature in a triple store would yield a different serialisation at verification time, and would thus invalidate the signature.

3.5 Querying Semantic Version Control

If we are to take full advantage of the DL underpinnings of the Semantic Web approach, then there is a need to consider the types of questions developers in our two case studies would pose to the version control repository. Such questions must be sufficiently complex enough to be not to implemented in a trivial manner in an

[48] http://jena.sourceforge.net/.
[49] http://usefulinc.com/foaf/signingFoafFiles.

RDBMS. Complexity should come in the form of repository federation, curation of different data sources and the use of procedures (builtins) found in most rule languages.

A reasonable set of questions should be able to discover additional evidence in the event a version control repository loses its integrity. Computer systems, especially version control systems, typically do not attempt to search for *additional* information that can contribute to forming a conclusion about the validity of metadata. Although Git and GNU Arch digitally sign commit metadata, if a signature failed, the system would fail; it would not pursue another avenue to see if external information could override the broken signature.

While searching for additional information to determine validity in a version control system is desirable, it is important that any questions programmed into the system do not make *exhaustive* searches or use algorithms that could run out of control. With this is in mind we have put together some questions that might be asked in the domain of our case studies. These questions attempt to use federation where ever possible, so not to limited the system to pre-existing knowledge.

Rather than attempt to automatically infer trust, we want the answers to these questions to *guide* developers and administrators in deciding their next actions. Automatic inference of trust, for example, creating trust policies [Dimitrakos et al. (2001); Bizer (2004b)], scores, or metrics [Golbeck and Hendler (2004a,c)] is a complex subject that goes beyond the scope of this thesis and will not be covered in this chapter[50]. A reasonable approach would be for answers to be in the form of a report that explains what has been found, along with a set of recommendations.

Below is a set of scenarios that include example queries for our semantic version control system. We have defined two types of scenario: metadata integrity recovery, and repository federation. Each scenario has been tailored to each of our case studies.

3.5.1 FLOSS Questions

1. A digital signature fails in the repository, search for other information that can help determine trust:

   In the event a digital signature fails in the repository, the project administrator must take steps to determine the reason for the failure. Failure could be

[50] These topics are, however, suitable for future work as we will describe in Section 6.3.

either due to inadvertent corruption in the repository or a sign of malicious modification. Apart from the repository itself, additional information can be found in the following places:

- Project DOAP description
- Author's project FOAF description

The administrator should firstly check that the author of the failed signature is actually an authorised committer, since this will quickly determine that an unauthorised intrusion has taken place; this information is contained in a DOAP file, published on the project's main webpage. If the author is a known committer, the administrator should instruct the repository to search for other commits by this author to analyse if they are not committing properly. The repository will

2. Source code dependency federation

   Developers will often experience missing dependencies that require importing into their repository. If we consider a Java™ example, the missing dependency will most likely be an unsatisfied **import** declaration. In most cases the developer will need to find third-party libraries to satisfy the dependency which diverts them from their problem solving.

   Rather than explicitly importing third-party libraries, the developer should ask the repository to search known repositories for the missing dependency. Metadata for the dependency is then imported into the developer's repository, completing the dependency. The actual source code will remain at the remote repository, creating a federated compilation environment.

3.5.2 IST Project Questions

1. A digital signature fails in the repository, search for other information that can help determine trust:

   As in Section 3.5.1, the repository administrator must take steps to discover the cause of the digital signature failure. Since an IST project has a smaller collaboration community this analysis should be slightly easier and have more information at its disposal.

   The administrator should firstly check whether the author's public key (FOAF) has been signed by one of the known CAs in the project consortium.

The author's FOAF description and partner CAs are just two example of federable information published on the project webpage. Additional checks of the published project DOAP description will reveal whether the author is supposed to be making commits to the source code based on their responsibilities in the associated workpackage.

The repository should then produce a report based on the above information. As well as providing information on the number of changes since the last verifed commit, the repository can offer the administrator the option to *override* the broken signature to fix the problem. Future verifications ignore the original signature in favour of the new one.

2. Source code integration federation

   Project partners during an integration phase need access to source code from other partners to produce an integrated prototype. A more convenient method is for metadata from each contributing partner to be federated to the integrating partner. To keep track of IPR attribution, the integrating partner will *tag* each set of metadata with a digital signature so to assert where it came from. Then, in a similar fashion to Section 3.5.1, the integrating partner will access source code directly from the remote contributing partner's repositories.

Unlike the FLOSS scenarios which operate in a more open environment, both IST scenarios require more trust of federated information to be considered reliable. DOAP, FOAF, and CA information must be published by a trusted party (the project coordinator) otherwise answers in the event of a signature failure could be misleading. If wrong doing was discovered in an IST project and proved with these questions, contract obligations could be employed to resolve issues. In the case of FLOSS projects, all a repository administrator can is to ban the CA of the committer and manually check the affected source code.

The above scenarios and queries will form the basis for our analysis of the benefits of Semantic Web technology in Chapter 5. We will expand on these questions, demonstrating how they can be implemented in practise and provide experimental performance results.

3.6 Summary

Distributed collaborative software development, by its very nature, relies on interaction with third-party remote servers. Analysis of two approaches to version control reveal that the current RDBMS-based version control systems do not provide the necessary support necessary for successful inter-domain development as required by our two case studies: FLOSS and EC IST Framework projects. In each case study there is a need for collaborators not to *implicitly* trust the integrity of the remote host, rather rely on the integrity of the repository's metadata, secured using digital signatures and a PKI.

Semantic Web technologies offer another approach that should be considered. Unlike an RDBMS, RDF graphs appear ideal for data federation, which is desirable in distributed collaboration. The selection of Semantic Web technology in version control can be seen as a significant test as to whether it is a valuable and practical technology. We have noted issues that still need to be resolved before take up improves; further analysis can be found in Chapter 5. Issues that should be addressed in our design include canonical RDF and provenance.

Nevertheless, the capabilities of SVN, GNU Arch, etc. represent a baseline version control capability which has proven itself over the years. Both Subversion and GNU Arch have introduced new architectural refinements, for example, file hierarchy restructuring, scalable and distributed repositories, and atomic commits to version control. It might therefore be productive to introduce some of the lessons of RDBMS version control into a Semantic Web DL approach.

In the next chapter we describe a design for an ontology for version control that uses Named Graphs as a mechanism for provenance. The purpose of this ontology is to act as the schema for a knowledge-base that performs the same functionality as a Subversion database. Our ontology leverages existing Description Logics and integrates with a new method of RDF digital signatures that promotes *explicit* trusted collaboration based on an established PKI. We go on to describe how our ontology, Named Graphs and digital signatures form the basis of an online collaborative tool that can support the necessary requirements of our case studies.

Chapter 4

Design and Implementation

This chapter provides an overview of our version control ontology, the use of Named Graphs for provenance, and security considerations in the form of digital signatures. We provide our rationale for using DL as the underlying logic; we also demonstrate ontology extension and argue for re-using other ontologies to promote interoperability on the Semantic Web. Our design shows how Semantic Web technology can be used as the foundations of a next-generation version control system that supports distributed collaborative software development.

There are two parts to our design: Document Provenance which provides *descriptive* provenance similar to related work in Section 2.3.1 based on DL; IPR attribution based upon Named Graphs. Attribution is enforced by RDF digital signatures that help maintain integrity, forming the foundations of developer trust and accountability, in an otherwise open environment.

4.1 Ontology Design Overview

Software version control repositories like SVN manage the changes made to documents over time. SVN uses a bespoke metadata format to record the author, description and version of a document which cannot readily be shared externally. Unlike CVS which uses a form of delta versioning [Hudson (2002)] on documents only, SVN is in addition capable of versioning directory structures and metadata. A well-known consequent restriction of CVS is its assumption of long-lived file names and, particularly, directory structures.

Another immediate problem with older tools such as CVS is that they keep the history metadata and delta versioning information together in the same logical structure. The Delta-V Working Group addressed this problem by separating the history and version metadata. Subversion [Collins-Sussman et al. (2004)] also improves on this problem, introducing a relational database to store metadata. To further develop this and leverage the rich tools of the Semantic Web we introduce Document Provenance, a Description Logic (DL) [Baader et al. (2003)] framework based on open standards which can be used for semantic version control and validation.

4.1.1 Requirements

If we look back at our research motivation in Section 1.2 we listed a set of features that are missing from current version control systems, but necessary for effective distributed software development. We went on to analyse the suitability of RDBMS and Semantic Web technology against these features, based on the needs of our case studies. These features included:

- Trusted provenance of server
- Knowledge federation
- Semantic interoperability
- New facilities (reasoning)

In this chapter we will propose a design that satisfies the above requirements. By thinking about security as an integral part of our design, we can worry less about the provenance of the server since integrity is vested in the underlying metadata of the version control repository. Reusing existing ontologies provides the first steps of semantic interoperability and knowledge federation. These components then lay the foundations of new facilities that include semantic reasoning.

4.1.2 Document Provenance

Document provenance (DP) is an abstract and somewhat ill-defined concept that associates authenticity with a document, based on work done by Buneman et al.

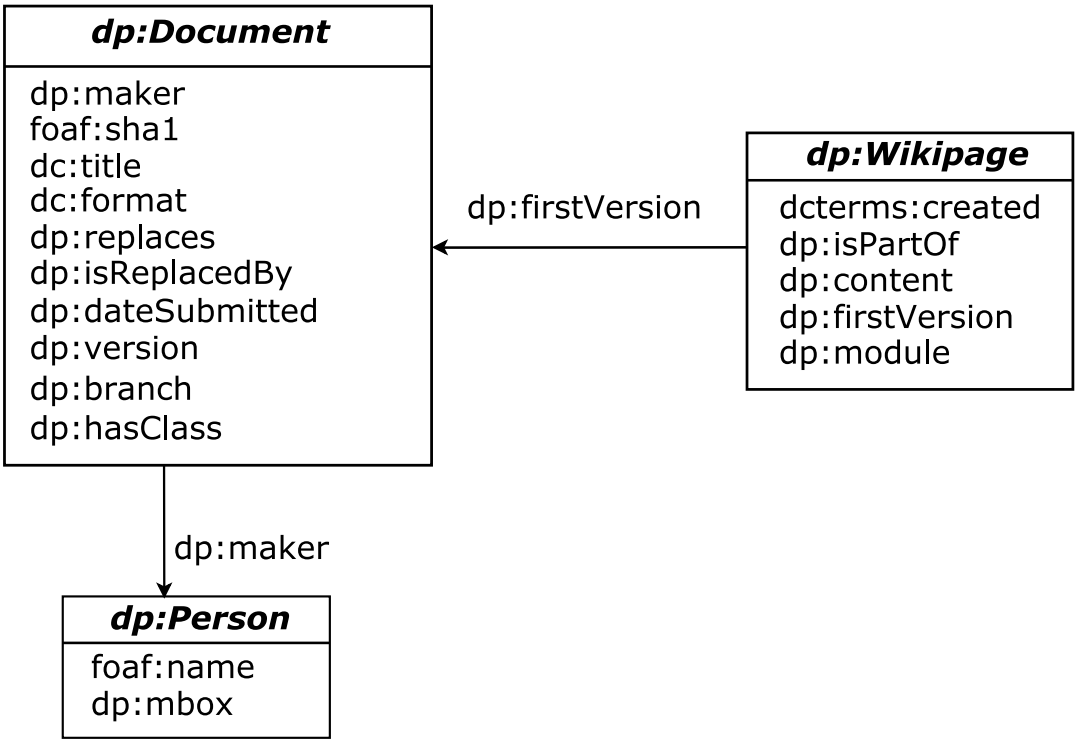

FIGURE 4.1: Document Provenance Ontology - OWL sub-language OWL DL, expressivity $\mathcal{SHIOF}$ (D)

(2001a), Goble, and Szomszor and Moreau (2003). The term implicitly assumes that provenance should be bound to information at the level of documents (URIs) rather than, for example, that of websites (as is achieved by HTTPS authentication) or of individual rows in a database. If we consider Java™ source code, the natural document unit is a class; these are usually kept in separate files. The Java™ compiler enforces this naming convention. Hence the class unit we map onto a document is intended to be the smallest natural source object that should be updated as an entity.

We have defined DP as a DL framework using RDF and OWL to develop an ontology that describes documents as they evolve. While we can leverage existing ontologies, we believe that we need to introduce small extensions for semantic version control. Figure 4.1 shows the three new classes we have created which themselves inherit from FOAF and DOAP as well as importing properties from Dublin Core. An RDF/XML-ABBREV version can be found in Appendix A.

Our use of existing ontologies is important because simply defining a new ontology does not help in shared understanding across domains. This concern was voiced

by Guus Schreiber at Berliner XML Tage 2004[1], who stated "Good ontologies are used in applications. They represent some form of consensus in a community [. . .] creating my own ontology is a misappropriation of the term. Ontology is about shared understanding" [Cyganiak (2004)]. The DP ontology describes only information that can be readily extracted from a document, e.g. file name, date last modified, etc.; this is simply metadata, it does not hold information describing who made it.

Here we briefly describe each OWL class in our ontology along with any OWL constructs used. We include paraphrasing of our concepts and roles similar to that used in Rector et al. (2004). Restrictions on classes are based on recommendations in Schreiber (2005) and Gruber (1995). The namespace for our ontology is denoted by the prefix dp:.

4.1.2.1 Document Class

dp:Document represents an individual version of a version controlled resource (see Table 4.1). It records common metadata that can be readily extracted from the document, as well as information that cannot: authorship, preceding and succeeding versions. Preceding and succeeding versions can be zero, one or many.

4.1.2.2 Wikipage Class

The dp:Wikipage class acts as an anchor point for a version controlled resource (see Table 4.2). This is comparable with a Version Controlled Resource found in WebDAV and Delta-V. The Wikipage has access to only the first version of the resource, includes a description and when it was first created. dp:Wikipage extends foaf:Document. Whilst the dp:Wikipage only points to the first version, there may be several versions in a chain after the initial version. Only one dp:Wikipage can exist.

By anchoring a version controlled resource with a dp:Wikipage, we effectively force the use of informal collaboration into the development cycle through the Wiki interface.

[1] http://www.xml-clearinghouse.de/ws/BXML2004/.

OWL:
class(dp:Document partial foaf:Document ***SubClassOf***(dp:Document, foaf:Document) ***restriction***(dc:title someValuesFrom(xsd:string)) ***restriction***(dc:format someValuesFrom(xsd:string)) ***restriction***(dp:replaces allValuesFrom(dp:Document)) ***restriction***(dp:isReplacedBy allValuesFrom(dp:Document)) ***restriction***(dp:maker someValuesFrom(dp:Person)) ***restriction***(foaf:sha1 someValuesFrom(xsd:string)) ***restriction***(dp:revision someValuesFrom(xsd:nonNegativeInteger)) ***restriction***(dp:branch someValuesFrom(xsd:nonNegativeInteger)) ***restriction***(dp:dateSubmitted someValuesFrom(xsd:dateTime)) ***restriction***(dp:hasClass someValuesFrom(java:Class)))
Paraphrase:
A Document is *any* document that has *amongst other things* a title, format, maker, sha1 sum, revision, branch, class, and a date of submission. A Document *may* replace zero, one or many Documents and be replaced by zero, one or many Documents.

TABLE 4.1: dp:Document Constructs

4.1.2.3 Person Class

The dp:Person class extends foaf:Person (see Table 4.3) and adds two properties describing which projects that person works on and which common classes they have created or modified. One or more persons may be attributed authors.

4.1.3 Other Ontologies

Several other ontologies have been used in conjunction with our classes that describe projects, people and Java™ source code. Developers who use our version control system will create FOAF descriptions to identify themselves and DOAP projects DOAP (Figure 4.4) descriptions for their projects. Not all these ontologies are written in OWL; as we will discuss shortly, FOAF is a combination of RDFS and OWL, while the Simple Java Ontology and the DCMI are both written in RDFS. If we are to import these ontologies and keep our ontology OWL DL, each ontology must be in their OWL DL form, otherwise our ontology becomes OWL Full. We will explore this issue further in Section 5.3.2.1.

OWL:
class(dp:Wikipage partial foaf:Document ***SubClassOf***(dp:Wikipage, foaf:Document) ***restriction***(dcterms:created someValuesFrom(xsd:dateTime)) ***restriction***(dp:content someValuesFrom(xsd:string)) ***restriction***(dp:module someValuesFrom(xsd:string)) ***restriction***(dp:isPartOf someValuesFrom(doap:Project)) ***restriction***(dp:firstVersion someValuesFrom(dp:Document)) ***restriction***(dp:firstVersion cardinality(1)))
Paraphrase:
A Wikipage is *any* wikipage that *amongst other things* has a creation date, some textual content, is part of a module and a known DOAP project. Each Wikipage has *exactly* one Document that represents the first version of a Version Controlled Resource.

TABLE 4.2: dp:Wikipage Constructs

Figure 4.2 shows our DP ontology together with relations to FOAF, DOAP, and the Simple Java Ontology. This visual representation will be especially useful when we consider how to partition our ontology for digital signatures in Section 4.5.2.

4.1.3.1 Friend of a Friend

The Friend of a Friend (FOAF) ontology is possibly one of the best known projects associated with the Semantic Web. It defines a simple vocabulary that allows indirect friendships to be discovered automatically. On-line services such as `http://plink.org` take FOAF descriptions to create a large social knowledge-base.

OWL:
class(dp:Person partial foaf:Person ***SubClassOf***(dp:Person, foaf:Person) ***restriction***(foaf:name someValuesFrom(xsd:string)) ***restriction***(dp:mbox someValuesFrom(xsd:anyURI)))
Paraphrase:
A Person is *any* person that *amongst other things* has a name, an email address has co-authored a Document, and works on a project.

TABLE 4.3: dp:Person Constructs

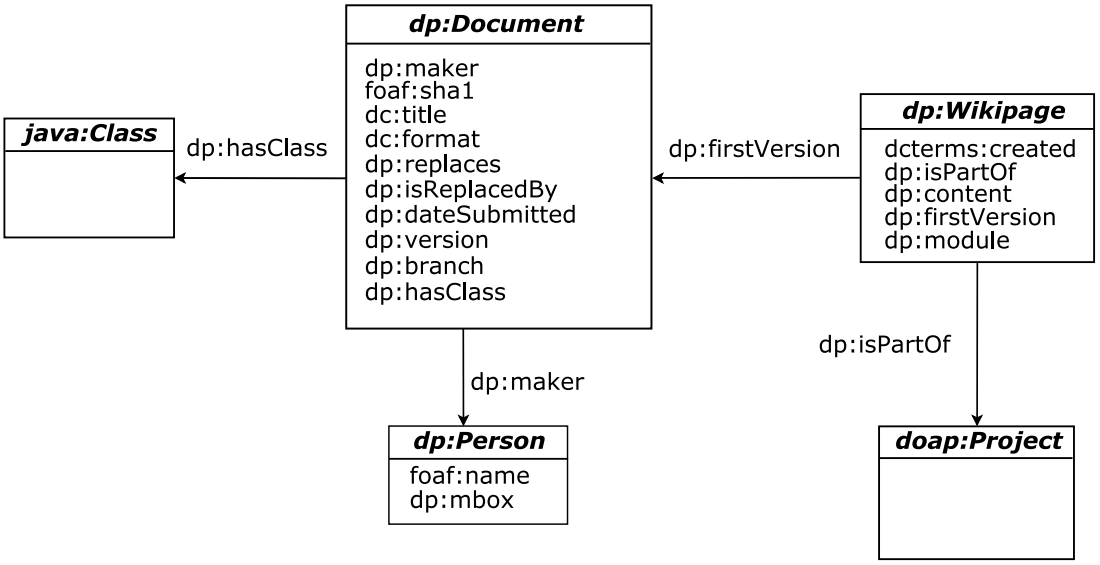

FIGURE 4.2: Expanded Document Provenance Ontology

Unfortunately, FOAF is more of a vocabulary than a formal logic for managing relationships between friends. It uses a combination of RDFS and OWL in its construction which means it must be categorised as OWL Full. OWL Full is known to be undecidable and therefore not recommended for basic subsumption and general purpose inference. In the future the FOAF community would be better served by an OWL DL version of the ontology. The Mindswap community[2] has produced an OWL DL version which would be a first step toward reducing the FOAF's current expressiveness.

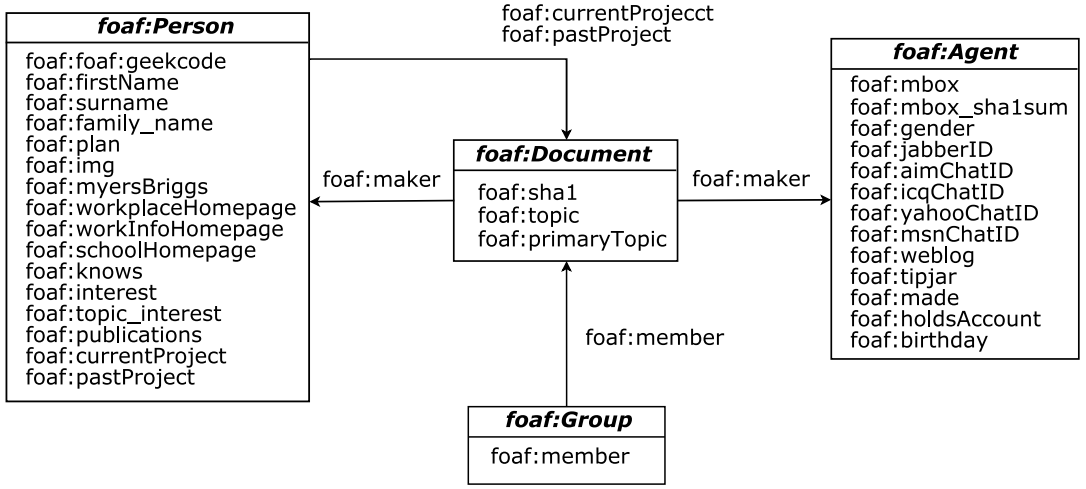

FIGURE 4.3: Friend of a Friend Ontology

[2]http://www.mindswap.org/.

4.1.3.2 Description of a Project

The Description of a Project (DOAP) ontology follows on from the success of FOAF by proposing a controlled vocabulary to describe a software-based project. Figure 4.4 shows a cut down version of the ontology with its various classes and properties.

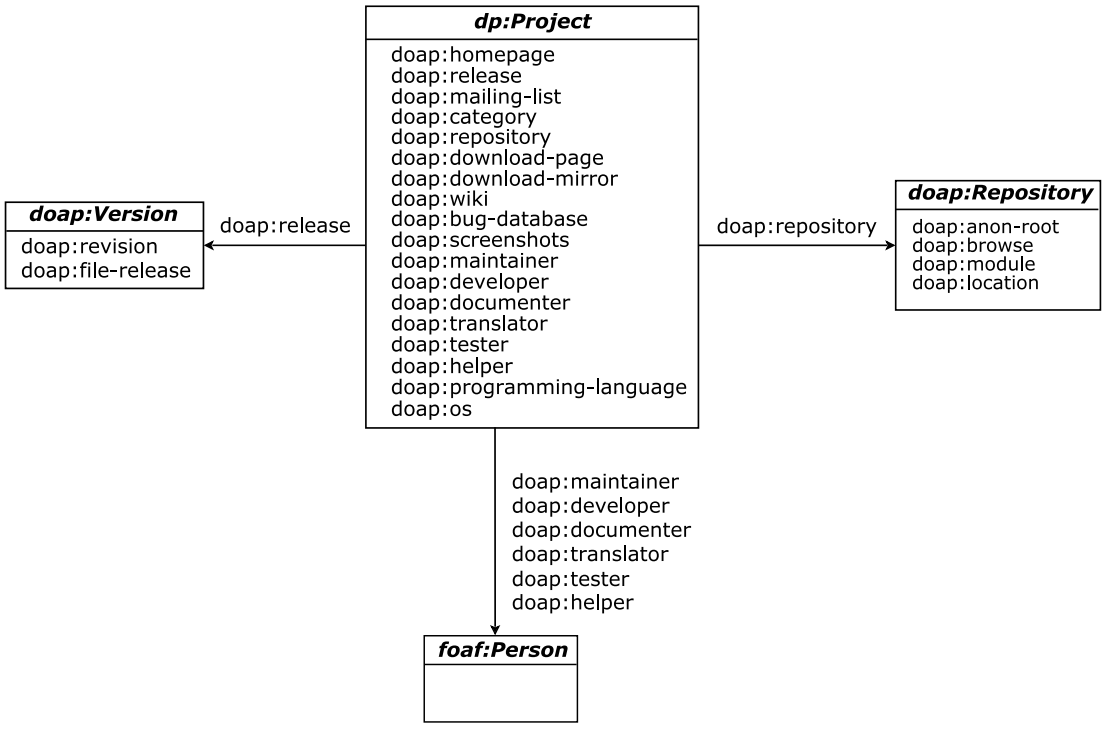

FIGURE 4.4: Description of a Project

4.1.3.3 Dublin Core Metadata Initiative

The Dublin Core Metadata Initiative (DCMI)[3] is a controlled vocabulary, primarily for resources found in libraries. The DCMI is attractive to use since it is a well known standard with a tightly controlled development cycle. The DCMI is formed of three vocabularies:

- DCMI Element Set (ISO Standard 15836[4])
- DCMI Metadata Terms
- DCMI Type Vocabulary

[3] http://www.dublincore.org/.
[4] Available at http://www.niso.org/internation/SC4/n515.pdf.

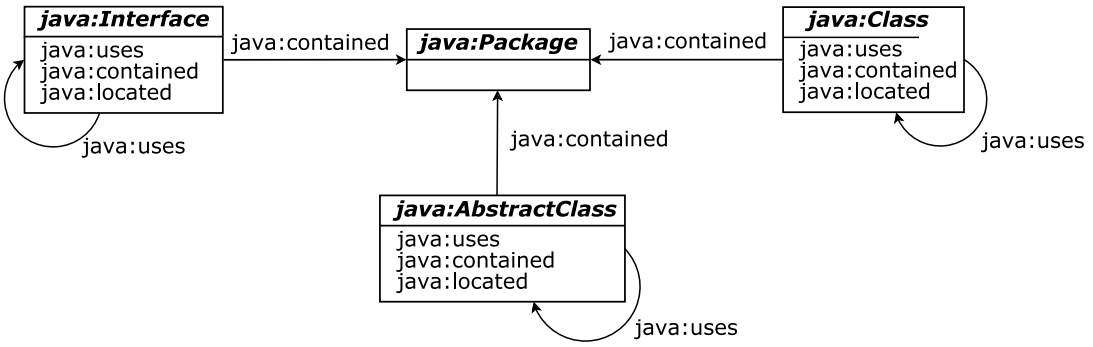

FIGURE 4.5: Simple Java Ontology

All three vocabularies are available in XML Schema and RDFS; Protégé distribute an OWL DL version[5].

Like FOAF and DOAP, the DCMI is more of a vocabulary than an ontology. This means its use of RDFS as a language is more of a convenience than a necessity. Properties defined by the DCMI are not restricted to particular classes, which means developers are free to use the vocabulary in any way they wish. It is quite common to use DCMI metadata embedded in HTML attributing authorship in a standard format. Although unrestricted qualification is useful in general, it is desirable for us to use the DCMI in such a way that we constrain its properties as we have shown in Table 4.1 and Table 4.2.

4.1.3.4 Simple Java Ontology

The simple Java™ ontology is a small component of the Simile project. It is a cut down version of certain Java™ concepts together with some non-standard properties (see Figure 4.5). The significance of the `java:uses` property is to show that a Java™ class may also *use* (import) other Java™ classes.

The simple Java™ ontology is based on RDFS, although we have written an OWL DL version that is suitable for our purposes. By extracting key portions of Java™ classes and attaching them to a dp:Document, we can infer relationships and dependencies in a project.

[5]Available at http://protege.stanford.edu/plugins/owl/dc/.

4.1.4 Construction

Ontologies can be constructed in various ways, from writing RDF/XML [Beckett (2004)] by hand to fully featured GUI interfaces. Due to the relative complexity of RDF/XML[6], hand written RDF is not encouraged. Most Semantic Web developers work with GUI interfaces such as Protégé[7] and SWOOP[8].

Protégé[9] is a well known knowledge management editor, developed by Stanford University. Whilst based on Frames and KIF (Knowledge Interchange Format)[10] it has been retrofitted to support RDF and OWL and has an extensible plugin mechanism. Protégé is capable of various forms of semantic reasoning including its native Protégé Axiom Language (PAL) for Frame-based ontologies, Racer for DL languages including OWL DL, and Jess for more complex reasoning.

SWOOP is an ontology editor and browser that is designed specifically for DL-based languages. It uses a web browser-like interface, that makes ontology authoring relatively quick with minimal fuss. SWOOP has access to its own Pellet [Parsia and Sirin (2004)] reasoner, an RDFS reasoner, and even SWI-Prolog for the Semantic Web Rule Language (SWRL)[11]. It also features a useful set of class expression constructs and an General Class Inclusion (GCI) editor.

SWOOP has several advantages over Protégé, primarily due to SWOOP's concentration on DL. SWOOP can quickly inform developers the expressivity of their ontology with the need for external reasoning tools. It is also very useful at debugging OWL ontologies [Parsia et al. (2005)].

Early versions of our ontology were designed using Protégé due to reputation and popularity. Recent versions, however, have been modified using SWOOP. This transition was based on the debugging features and highly visible OWL species validation in SWOOP. Species validation shows a knowledge engineer the DL sub-language at a glance. Determining the OWL sub-language in Protégé is a non-trivial task.

[6] RDF has two recommended concrete syntaxes: RDF/XML and RDF/XML-ABBREV.
[7] http://protege.stanford.edu/
[8] http://www.mindswap.org/2004/SWOOP/.
[9] http://protege.stanford.edu/.
[10] http://logic.stanford.edu/kif/dpans.html.
[11] http://www.mindswap.org/ edna/swoopRules/.

```
:G1 {
   :erw foaf:mbox <mailto:erw@it-innovation.soton.ac.uk> .
   :G2 foaf:maker "Rowland Watkins".
}
:G2 {
   :erw foaf:mbox <mailto:rowland.watkins@gmail.com> .
   :G2 dcterms:created "28-7-2006".
}
```

FIGURE 4.6: Self-Referencing and Cross-Referencing Named Graphs

4.2 Provenance Mechanism

While our Document Provenance ontology describes some interesting information about documents and basic associations between different versions of a document, we still require a mechanism to make assertions about our metadata and hence record provenance.

4.2.1 Named Graphs

Named Graphs provide a natural way to record provenance. Each graph is potentially labelled by a URI, which can then be referenced by other Named Graphs. Figure 4.6 depicts two Named Graphs using the TriG [Bizer (2005)] syntax, where the first graph states that the second graph was made by Rowland Watkins, while the second graph self-references, stating its creation date.

Each graph is named with a URI; blank nodes are not permitted under scoping rules defined in the RDF Recommendations. These scoping rules state that blank nodes are unique to the RDF graph where they are found and cannot be referenced outside of that graph. Since Named Graphs provide opportunities for creating relationships between graphs, any reference must be fully qualified URI. In Figure 4.6, the TriG syntax uses short-hand when labelling graphs and resources (:G1, :G2, :Bob). Blank nodes in most RDF concrete syntaxes take the form, _:blank_node_id. In many RDF libraries, blank nodes are automatically generated each time the serialised syntax is read from persistent storage.

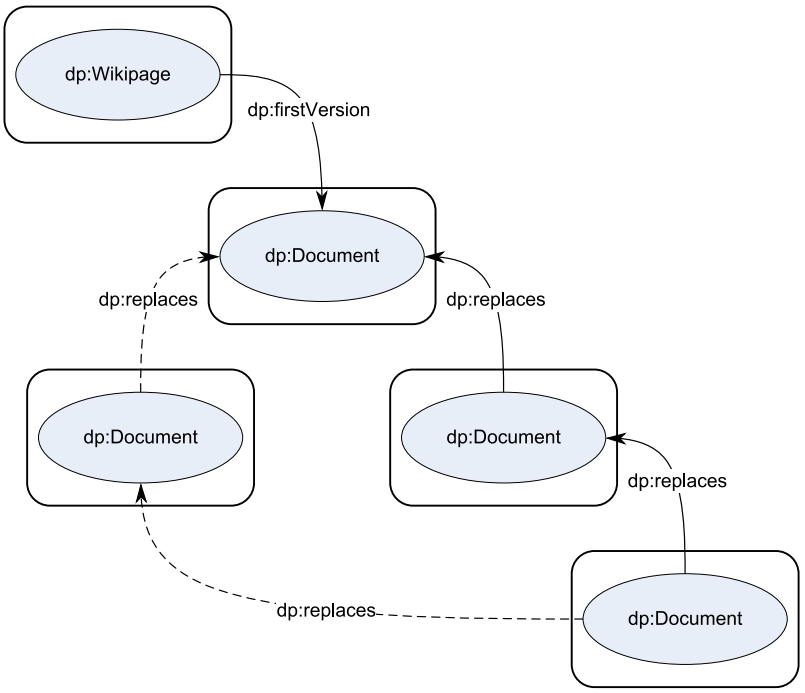

FIGURE 4.7: Version Control using DP

4.3 Modelling Version Control with DP

Up to this point we have defined an ontology that models version control and selected a provenance mechanism that allows us to created relationships between RDF graphs. To the casual observer this may seem no better than what has already defined by Goble; Zhao et al. (2003, 2004b), at best can be considered a glorified form of logging. To show how both points are not the case, we will firstly show how versions relate to one another visually, which will give an idea of DP's structure. We will then go on to explain how digital signatures can be attached to DP instances. Since the digital signature is created by the user, they become part of the process for generating DP; logging systems such as Log4J are typically side-effects of events in the system. The intention is for our version control structure to be self-contained, non-repudiable, and immutable.

Figure 4.7 is a graphical representation of a version history using DP. A **dp:Wikipage**, as we noted earlier, represents the anchor-point for a version controlled resource, which points to the first version. We then have several versions that feed off the first, including a *fork* and a *merge*.

In terms of RDF concepts represented in Figure 4.7, each ellipse represents a DP class and each box represents the encompassing Named Graph. Relationships feed back up the history to show how which version is being superseded.

Since each commit is contained within a Named Graph, we can start thinking about how to maintain the integrity and enforce IPR attribution for each version controlled resource. As we stated in Figure 4.1, each **dp:Document** contains the SHA-1 digest of the document it describes. If we could digitally sign the Named Graph, then attach the associated signature, we would not only be able to verify the document under version control, but also the metadata that describes it.

4.4 Security

Named Graph-based provenance is not limited to simple assertions about who made the assertion and when. Such assertions, whilst true according to the open world assumption (see Section 2.4.4), do not uniquely bind an owner to an assertion, or set of assertions. Cryptographic methods such as digital signatures offer one way to uniquely bind a security principal[12] to a digital document and, coupled with digital certificates, add non-repudiation to signatures. Since it is difficult to guarantee that every RDF graph found on the Semantic Web is error free, Named Graphs and digital signatures form a good heuristic for evaluating trustworthiness. Simply asserting an RDF graph does not mean that the information it contains is reliable. Trusted metadata methods based on digital signatures are also a first step toward a basic level of trust on the Semantic Web [Bizer (2004b)]. Note that while we do not preclude the use of access control or confidentiality mechanisms on our trusted metadata, such mechanisms are beyond the scope of this thesis.

Figure 4.8 gives an example of how Named Graphs can be signed. A separate Named Graph (in red) is created with the signature information, that then *asserts* the referred Named Graph. We will see a concrete example of this in our work on the NG4J project in Section 4.6.4.

4.4.1 Signing RDF Graphs

As we argued in Section 3.2.2, to help create a more robust trust infrastructure during distributed collaborative software development, digital signatures must become a core part of the version control workflow. Digital signatures ensure the

[12] See http://www.pluralsight.com/wiki/default.aspx/Keith.GuideBook/What%20Is%20A%20Security%20Principal.html for further details.

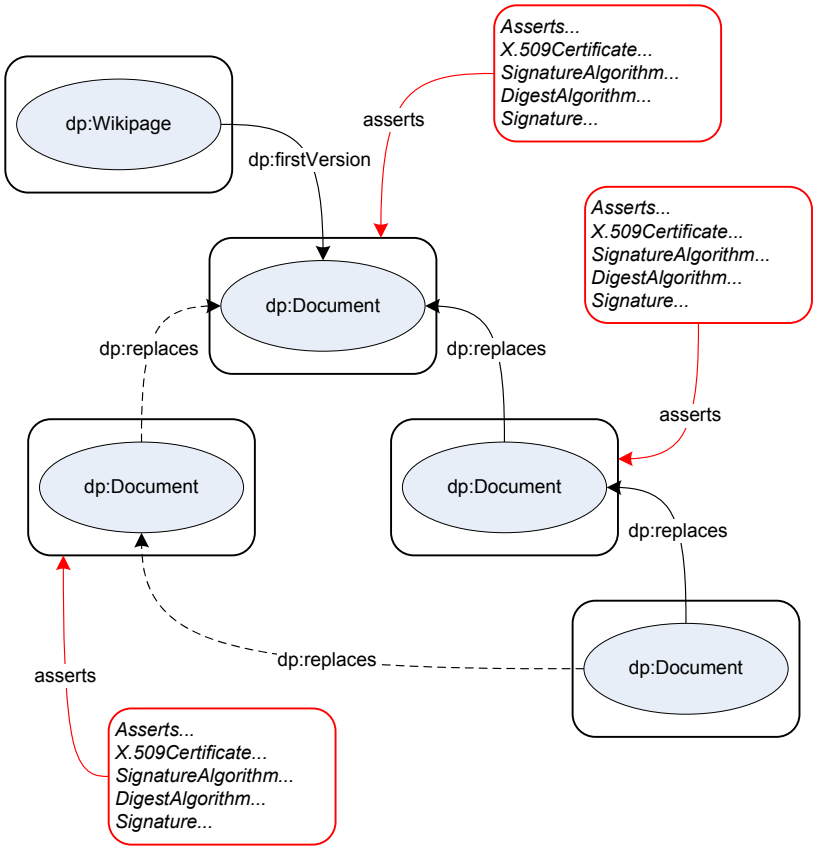

FIGURE 4.8: DP with Digital Signatures

integrity of messages and, when generated with an appropriate PKI, support non-repudiation[13] [McCullagh and Caelli (2000); Zhou (2003)]. The goal in our work is to create an RDF digital signature framework that follows several of the principles in Tummarello et al. (2005), although builds upon Named Graphs rather than RDF Reification.

It is important to note that non-repudiation in our framework is supported through the use of asymmetric public key cryptography. This means the onus of responsibility for protecting private keys lies in the hands of the developer or administrator of the online repository. This is why PKI-based systems such as X.509 go to the trouble of using the Certificate Authority as the trusted third-party, tracking compromised certificates using a Certificate Revocation List (CRL).

Symmetric key based systems such as Kerberos and the SAML protocol, do not support non-repudiation and therefore should not be used as part of our RDF digital signature mechanism.

[13]Non-repudiation is supported in X.509v3 with the *KeyUsage* (OID 2.5.29.15, http://oid.elibel.tm.fr/2.5.29.15) critical extension. The non-repudiation bit is limited to digital signatures and precludes certificate and CRL signing.

4.4.1.1 Carroll's algorithm vs. *nauty*

Due to the graph-like nature of RDF and the issue of blank nodes (see Section 3.4.1.1) generating reliable and robust digital signatures is non-trivial. As we have noted in Section 3.4.1.1), Carroll's algorithm and ***nauty*** take very different approaches to graph canonicalisation. Carroll's algorithm can be seen as a quick and cheap canonicalisation method that *does not attempt* to solve the isomorphism problem; Carroll (2003) goes to some lengths to state that the proposed algorithm is not intended for arbitrary RDF graphs. The ***nauty*** approach is far more elegant, satisfactorily solves the isomorphism problem; however, it is overly complex to program [Carroll (2003)] and non-polynomial. It would be unwise to rely on a non-polynomial algorithm in an RDF digital signature solution for the Semantic Web.

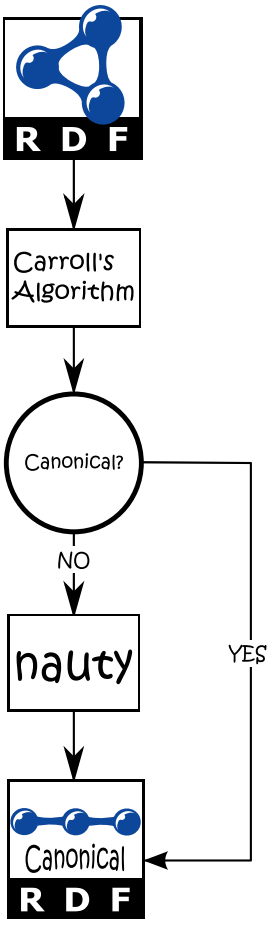

FIGURE 4.9: Comprehensive Canonical RDF Workflow

A another approach we have yet to consider is to combine Carroll's algorithm with ***nauty***. We could leverage the speed of Carroll's algorithm for simple cases and ***nauty*** in complex cases, creating a workflow solution that is comprehensive.

Figure 4.9 shows our proposed canonical RDF workflow. At first glance, this workflow seems reasonable; we modify Carroll's algorithm to detect when it has failed, then pass it on to ***nauty*** to complete the canonical reordering. Even if Carroll's algorithm produced false negatives, they would be taken care of by ***nauty***.

One potential problem with Figure 4.9 is the expectation that a canonical representation produced by each algorithm is identical. Both algorithms take very different approaches, meaning that it is possible in one instance for Carroll's algorithm to succeed and successfully sign an RDF graph. When the graph's signature comes to be verified, Carroll's algorithm fails, passing the graph to ***nauty***. ***nauty*** then produces a canonical reordering, however, completely different to Carroll's algorithm, thus breaking the signature. Appendix C.1.1 gives an example of where this case is true; we canonicalised the WordNet NounWordSense OWL class and compared the results of each algorithm which are shown to be very different. It is therefore important that the algorithm used to create the signature is the same as the algorithm used to verify the same signature.

If we were to take a pragmatic approach and choose between the two algorithms, Carroll's algorithm offers the better choice. It is fast and able to cope with the majority of graphs we are interested in for our research. To be used in our RDF signature solution, however, we must devise a *conservative* approach that will guarantee a canonicalisation that can be reliably replicated in the future.

4.4.1.2 Conservative Canonicalisation

Conservative canonicalisation is an approach that accepts the existence of false negatives when using Carroll's algorithm, i.e., if the algorithm claims it cannot canonicalise an RDF graph even though it should be able to, then we accept its conclusion. This approach may reject more graphs, however, should reduce the number of digital signatures generated that subsequently fail when verified. We have therefore placed restrictions on the introduction of blank nodes to improve reliability of our digital signature mechanism.

Our approach can be summarised as follows:

1. DP instances should consist of fully labelled RDF graphs.
2. RDF Digital Signatures should consist of fully labelled RDF graphs.

3. External federated RDF that is to be signed should be analysed for blank nodes.

Carroll's algorithm appears to suit to our needs in all cases given we take the conservative canonicalisation approach. We do not intend to use Sayers' algorithm, which would otherwise introduce additional complexity when managing digital signatures and future verification. Continual publishing and updating of signatures seems to be rather laborious for our purposes.

4.5 Open Issues

There are several open issues in our design that have yet to be fully resolved, mostly due to the experimental nature of our approach and the immaturity of the technology being used. While solutions to these issues are not critical to providing answers to our research question, they will have some impact on the final analysis in Chapter 5.

4.5.1 NG Management

Named Graphs are a relatively immature provenance recording mechanism. Our need for attaching digital signatures to RDF graphs means that there is a serious need for appropriate management of graphs and signatures as well as some understanding of what is to be digitally signed.

4.5.1.1 Signature Management

Digital signatures must be stored in some reliable manner so they can be verified at a later date. The XML Signature Syntax and Processing [Bartel et al. (2002b)] standard defines an XML Schema for signature storage, although similar to having no canonical form, no structure exists for storing digital signatures. Tummarello et al. (2005) and Dumbill (2002) provide ideas on how this might be done; we have, however, noted in Section 3.3.2.2 and Section 3.4 these approaches are not suitable for our needs. Section 4.6.4 gives an overview of our realisation of RDF digital signatures.

4.5.2 Ontology Decomposition

Although an RDF graph is a collection of triples which can be merged and aggregated with other graphs according to the RDF Recommendation, there is little guidance or consensus on how to decompose RDF graphs into a smaller set of subgraphs. Such decomposition is useful when considering large ontologies or portions of an ontology that requires digitally signing. Ding et al. (2005) describe *RDF molecules* as a possible unit of decomposition for an ontology and go on to describe an algorithm for automatic decomposition. $\mathcal{E}$-Connections [Grau et al. (2005b,a)], an alternative approach, is also capable of automatic partitioning, although its primary purpose is as a syntactic and semantic extension to OWL (DL) that allows different ontologies of equivalent expressivity to be combined with *link properties*[14].

While automatic decomposition might be useful for signing large or complex ontologies, our work concentrates on a small ontology with a large dataset and therefore does not require substantial decomposition analysis. If we consider Figure 4.1 we notice that the cardinality of each class is different. Each version controlled resource can only be represented by one wikipage, which links to one or more document versions, created by a known person. We could naïvely sign over an entire ontology instance. When we add, however, a new version to the ontology instance and then attempt to verify our original signature, we will discover that the signature is broken. A better method is to decompose our ontology into separate Named Graphs independent of one another. It would then be a simple case to create secure hashes for each Named Graph, and sign each separately, storing the signature in a separate Named Graph.

Figure 4.10 proposes one way to decompose the Document Provenance ontology. dp:Wikipage is contained within its own Named Graph, which provides a secure, trustworthy anchor for version controlled resources; only one dp:Wikipage exists for any one resource. Each dp:Document with associated metadata (Java™ class information) is decomposed in separate Named Graphs, with successive dp:Document graphs referring to their predecessors (dp:replaces).

The advantage of independent signatures is that successive versions will always be verifiable and would not affect other signatures. It does mean, however, that our knowledge-base will contain a large number of Named Graphs with signatures that will require verification.

[14] Similar to datatype properties.

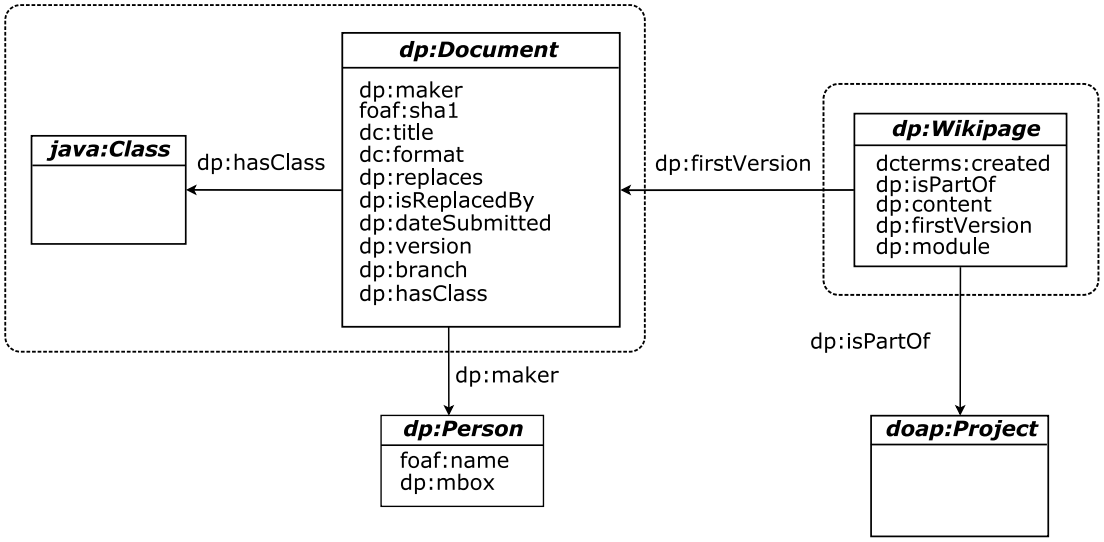

FIGURE 4.10: Document Provenance Ontology Decomposition

4.6 Implementation - An Online Collaborative Tool

Our online collaborative tool must provide version control services in a transparent manner, yet still allow developers to do their work. We have taken an existing Wiki, JSPWiki[15] as the our base system. As its name suggests, JSPWiki uses Java™ Server Pages and Java™ Servlets. Servlets provide a convenient mechanism for web applications. We have retained much of the general functionality of JSPWiki, although we have changed various underlying components to integrate the semantic and cryptographic features [Watkins and Nicole (2005a)].

4.6.1 Motivation

At first sight, the value of the synthesis of the WikiWikiWeb, Semantic Web and advanced cryptography for version control is not too obvious. They are distinct from one another even though they are used in the same application space: PKI-based cryptography is used to secure HTTP sessions with the Secure Sockets Layer (SSL); the WikiWikiWeb is used for dynamic collaboration on the WWW; the Semantic Web aims to make the WWW into a *Web of Knowledge*. Our motivation for using the WikiWikiWeb is based upon a WWW-based collaborative environment that is scalable and relatively easy to use. In Chapter 3 we discussed

[15] http://www.jspwiki.org/.

the needs of our case studies, which included the attribution and enforcement of IPR. In Section 4.1 we described how this could work with our DP If we have a PKI to validate users, then the same mechanism can be used for authorship.

The third part of the puzzle, the Semantic Web, is important not only for technologies such as RDF and OWL, but the logic it is based upon, namely Description Logic. The vast majority of version control systems use a relational database which is formal and static; such systems are difficult to change. DL languages, for example, OWL DL are built upon RDF which is much better suited to knowledge federation than an RDBMS. This means the underlying structure of a version control system that uses Semantic Web technology is relatively expressive and highly extensible. Coupled with this extensibility is another key advantage; the DL's inference capability, which provides effective tools for the distributed management of the software engineering process.

4.6.2 Architecture

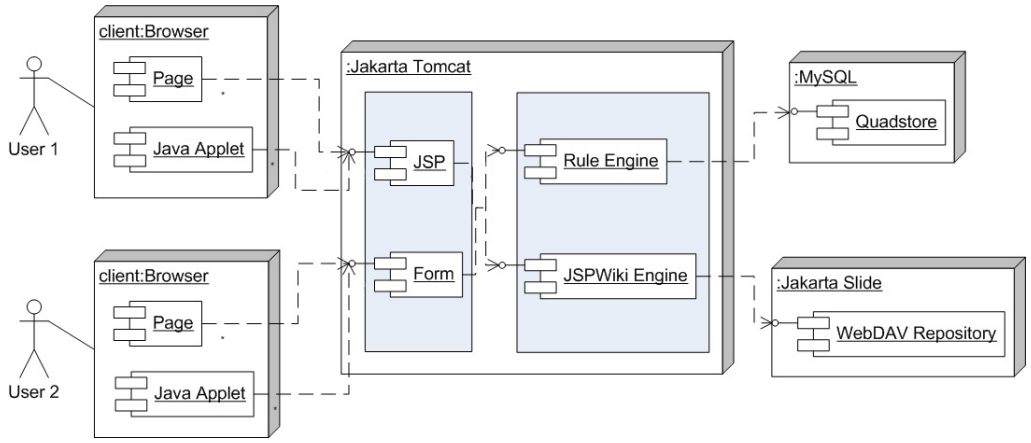

FIGURE 4.11: Online Collaborative Tool Architecture

Figure 4.11 shows the top level architecture of our online collaborative tool. Our architecture is split into three main portions: the client browser, the Jakarta Tomcat application server, and the RDBMS and WebDAV server.

Both on the client and server side, we use the Jena 2 Semantic Web framework [Carroll et al. (2004)] and its Named Graph extension library, NG4J[16]. We use NG4J extensively to manipulate RDF, Named Graphs and RDF digital signatures. Cryptographic support comes from the Bouncy Castle[17] JCE provider

[16]http://www.wiwiss.fu-berlin.de/suhl/bizer/ng4j/.
[17]http://www.bouncycastle.org/.

and the digital signing of Named Graphs allows us to track IPR attribution and enforce non-repudiation, suitable for our two use cases.

4.6.2.1 Client Side

The client side uses any standard web browser capable of executing a Java™ applet. Developers select their source code to upload; the applet's job is to generate metadata based on those files and cryptographically sign it. Note that in this architecture, the integrity of the repository is vested in the trusted metadata described in Section 4.4; the repository contents may freely be duplicated to protect against loss of the primary site and core trust is vested only in individual authors, not in the repository itself.

The browser interface allows developers not only to check-in and update new source code but also to actively collaborate with other developers using the Wiki as an online development journal: design issues can potentially be tracked and additional information such as UML diagrams or collaborative "whiteboard" sketches for each class can be attached in the Wikipage or linked from the WebDAV repository as appropriate. Figure 4.12 shows the main Wikipage for our online collaboration tool. Developers can navigate through package and class hierarchies since each is represented by its own Wikipage.

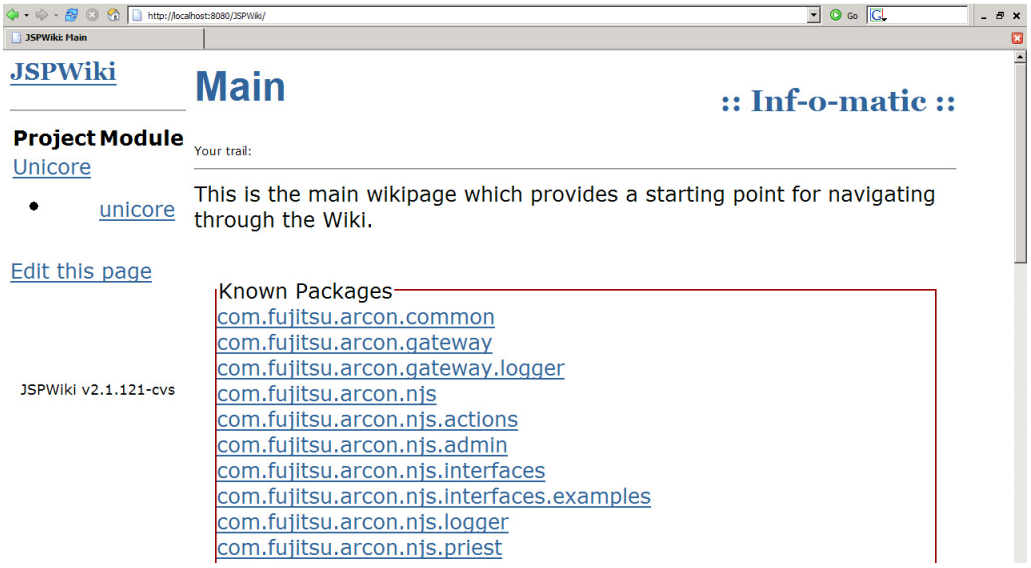

FIGURE 4.12: Online Collaboration Tool Interface

4.6.2.2 Server Side

The server side is a generic servlet-based web application that hosts an enhanced instance of JSPWiki. JSPWiki handles all portions of the interface based on a template system similar to, but simpler than, Struts[18]. Wiki content is stored in an MySQL RDBMS as a Named Graph quad store[19], in contrast to JSPWiki's flat file persistent storage. Source code is stored separately from its metadata, in a WebDAV repository which can either be co-located with the Wiki or run on a different remote host.

To support external access to the Named Graph quad store, we have developed a web service interface that enables distributed knowledge federation. This functionality will become critical when we consider new information that can be brought into the system by federating external sources (see Chapter 4.7).

Commit Process When the server processes requests from the client it verifies the attached signature and *endorses* the commit with its own signature if successful. The additional signature adds a timestamp that confirms receipt of the commit and that the client's Certificate Authority is known and trusted by the server. All signed metadata, once verified, is persisted in the quad store for later retrieval.

Wikipages Wikipages are stored as plaintext files, which give developers the opportunity to discuss design issues, post news, link diagrams, and make announcements. JSPWiki has several plugins that make it easy to add UML [Jacobson et al. (1998)] diagrams to wikipages. It is essential to complement the non-repudiable foundations of the repository (signed metadata) with this soft interface.

As an additional benefit besides the adherence to the Model View Controller (MVC) [Buschmann et al. (1996)] pattern, keeping the quad store and document storage mechanism separate from the Wiki means we can easily provide alternative access to the source codebase using Web or Grid Services [Atkinson et al. (2005)]. These can be used to support automatic build and installation of named releases onto Grid hosts.

Our online tool utilises Jena 2's forward RETE rule engine for inference support. We have written various rules that match triple patterns to create new relations

[18] http://struts.apache.org/.
[19] The Named Graph's URI is the first element of the quad.

which we can then query with an RDF query language like RDQL [Miller et al. (2002)]. While our DL implementation is based around Named Graphs, it is compatible with Jena 2, so we can take full advantage without any problems. Indeed, we have also used Jena 2's OWL reasoner to check periodically the consistency of the quad store based on our ontology.

4.6.3 Named Graphs for Jena (NG4J) API

Developed in collaboration with Chris Bizer[20] and Richard Cyganiak[21] of the Freie Universität Berlin[22], the NG4J API[23] is an experimental implementation of Named Graphs [Bizer et al. (2005a)]. NG4J defines a set of interfaces for manipulating Named Graphs based on Jena 2. Named Graphs can be created, merged, and serialised using an XML concrete syntax, TriX[24], and a Turtle-like concrete syntax, TriG[25].

The quad is the basic unit in NG4J which extends the Jena 2 Triple by appending the label of the Named Graph it belongs to. At the core of NG4J is the NamedGraphSet class, a logical set of NamedGraphs which extends the Jena Graph interface and adds the graph name. In Jena 2 it is common to manipulate graphs as *models*. Models provide a convenient API for developers to create resources and add properties to them. Since NG4J works primarily with graphs, it operates at a subtly lower level.

A NamedGraphSet can be *viewed* as a Jena 2 *model* by creating a union of all the Named Graphs in that set. The Model itself becomes a Named Graph and any additions to the model become part of that graph.

4.6.4 Semantic Web Publishing Framework

The Semantic Web Publishing Framework (SWP) extends NG4J and implements an RDF digital signature toolkit similar to what Apache Security does for the XML Digital Signature recommendation [Bartel et al. (2002b)]. Our contribution to this framework includes keystore manipulation and core functionality necessary

[20] mailto:chris@bizer.de
[21] mailto:richard@cyganiak.de
[22] http://www.fu-berlin.de/.
[23] Available at http://ng4j.sourceforge.net/.
[24] http://swdev.nokia.com/trix/TriX.html.
[25] http://sites.wiwiss.fu-berlin.de/suhl/bizer/TriG/.

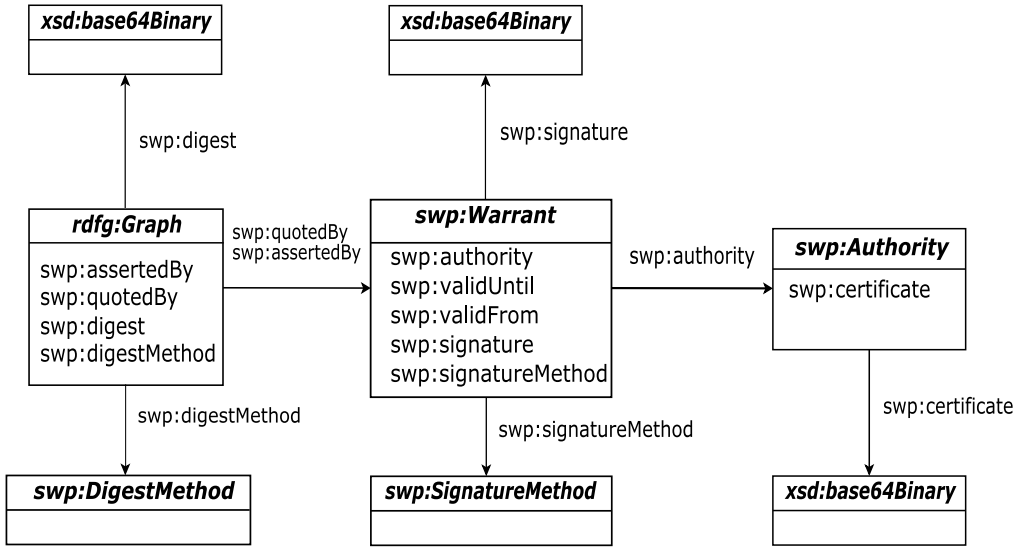

FIGURE 4.13: Semantic Web Publishing Ontology

to create, query and verify RDF signatures as described in Carroll et al. (2005); we have also contributed to the development of the SWP API and ontology. The SWP ontology[26] reuses a lot of terms found in the XML Signature Recommendation, although since XML is rather verbose and tree-like in structure, it is not necessary to directly map all elements found.

Figure 4.13 shows the SWP ontology as described in Carroll et al. (2005). As can be seen, it takes as much as possible from XML Signature; however, not all elements are necessary since modelling all elements in RDF would make querying inefficient and slow.

To generate an RDF digital signature, we first create a canonical Named Graph [Carroll (2003)] then hash it with an appropriate secure digest (SHA-1, SHA-224, SHA-384, or SHA-512. Please see Wang et al. (2005c,b,a) for reasons why SHA-1 may not be safe to use in the future.). This digest is placed in a special Named Graph called a Warrant Graph (see Figure 4.14) [Carroll et al. (2005)] and signed. Please note that the digest and signatures values have been abbreviated. Our work in the SWP has concentrated on the implementation of Warrant graph creation and verification. A full representation of a DP instance with a Warrant Graph can be found in Appendix A.

[26] Available at http://www.w3.org/2004/03/trix/swp-2/.

A Warrant Graph can contain any number of graph digests. Each digested graph is explicitly asserted by a known principal who possesses a digital certificate (X.509v3) or PGP key. The Warrant Graph asserts itself and signs itself with the principals credentials, certifying that not only did the principal make the assertion, but that the assertion has not been altered.

4.7 Federation Scenarios

In Section 3.5 we listed some example questions that should be answerable by our design. We will now expand on these questions and develop some concrete scenarios of where Semantic Web technology can be used to improve version control beyond the state-of-the-art, i.e., scenarios that cannot be achieved using an RDBMS. For the purposes of these scenarios, we have invented fictional names for a FLOSS and EC IST project; MyProject[27] and AcmeGrid[28] respectively.

The main requirement here is that each scenario must take advantage of Semantic Web knowledge federation. Rather than attempting to infer new knowledge from existing information inside our online collaborative tool, we want to reach out and extract useful information that can supplement existing knowledge to improve distributed collaborative software development.

```
:warrant {
   :warrant a swp:Warrant.
   :warrant swp:authority
   <mailto:erw@it-innovation.soton.ac.uk>.
   :warrant swp:signatureMethod
   swp:JjcRdfC14N-rsa-sha1.
   :warrant swp:signature
   "E2a...ylV"^^xsd:base64Binary.
   :warrant swp:assertedBy :warrant.
   :G1 swp:assertedBy :warrant.
   :G1 swp:digestMethod swp:JjcRdfC14N-sha1.
   :G1 swp:digest "YjR...hNz"^^xsd:base64Binary.
   :G2 swp:assertedBy :warrant.
   :G2 swp:digestMethod swp:JjcRdfC14N-sha1.
   :G2 swp:digest "NmM...2NW"^^xsd:base64Binary.
}
```

FIGURE 4.14: Warrant Graph Including Digital Signature

[27] http://www.ecs.soton.ac.uk/ erw/MyProject/
[28] http://www.ecs.soton.ac.uk/ erw/AcmeGrid/.

We will concentrate on two federation scenarios that can be of interest to software developers: metadata integrity recovery and distributed knowledge federation. Metadata integrity recovery investigates the use of knowledge federation to establish trust in the event part of a repository becomes compromised. Distributed knowledge federation uses multiple data sources in different trust domains to supplement missing information in a local repository. Each scenario will be applied to our case studies so that we get a range of new possibilities for data sharing in distributed software development.

The SPARQL web service used in these scenarios was developed using the GRIA 5 Development Kit[29]. This developer kit provides a reusable helper API for secure grid services. A simple client was also developed for calling the SPARQL grid service from other components of our online collaborative tool. The Jena 2 inference engine was used to drive the signature recovery scenarios; the rules can be found in Appendix D. Performance results for all federation scenarios can be found in Section 5.1.2.1.

4.7.1 FLOSS Federation

FLOSS federation presents a simple scenario where there exist two repository hosts (see Figure 4.15) who have a trust relationship. Developers at Host A have discovered missing dependencies in their code and query Host B for the missing information. Host B provides a secure SPARQL web service interface[30] so that developers can be confident of the connection between hosts.

Data passed between Host A and B includes only signed repository metadata[31]. This means that actual source code remains at the original host, with the result that distributed builds are possible in a similar fashion to Maven[32] repositories[33].

4.7.2 EC IST Federation

EC IST Federation presents a slightly more complicated scenario based on integrated prototype integration (see Figure 4.16). The Integrating Partner is responsible for integrating code developed by Contributing Partners A and B, either in

[29] http://www.gria.org
[30] Transport Layer Security and WS-Security.
[31] DP instances as found in Appendix A.
[32] http://maven.apache.org/.
[33] Please note that distributed builds have not been implemented during this research.

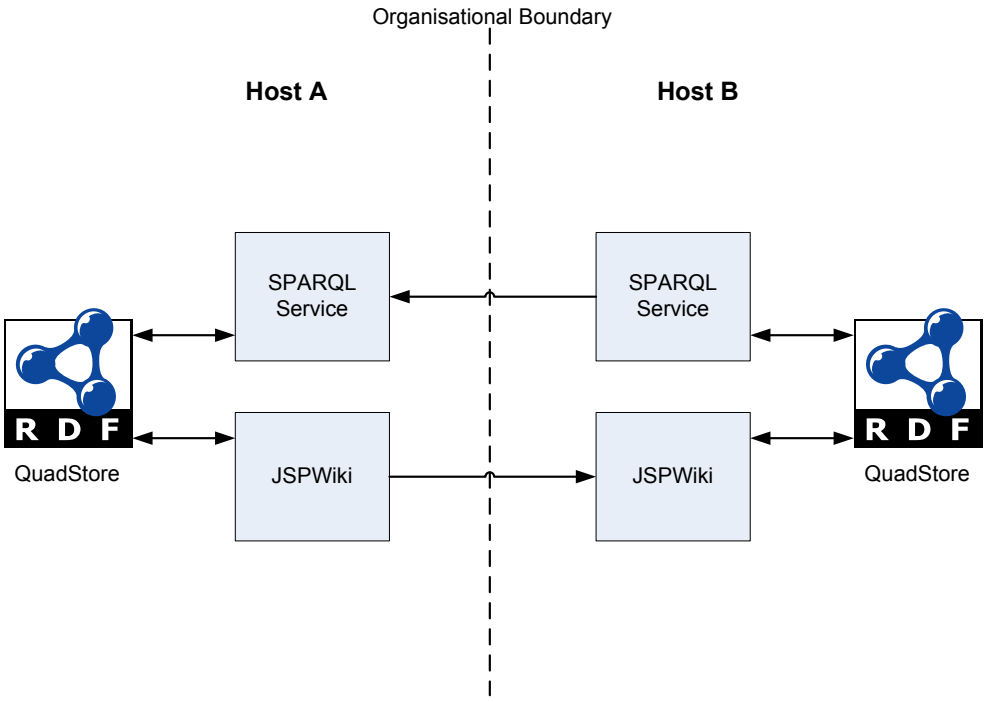

FIGURE 4.15: FLOSS Federation Scenario

source form or binary. Access to Contributing Partners A or B's repository is similar to Section 4.7.1, however, the Contributing Partner must take steps to record where they received metadata from to maintain IPR attribution.

The Integrating Partner will retrieve signed DP instances from each Contributing Partner, signing the result so to assert where they received the instances from (Figure 4.17). This additional signing process means the Integrating Partner acknowledges receipt of the metadata. Source code and binary code is accessed over HTTP from the Contributing Partners' WebDAV repository.

4.7.3 FLOSS Signature Recovery

The above scenarios describe the sharing of repository metadata across different trust domains. In the majority of cases, metadata integrity will not degrade due to transport between repositories. If a public FLOSS repository, however, becomes compromised as metadata is being federated, then it is possible for digital signatures to fail. The following FLOSS scenario looks at a strategy for simple integrity recovery with minimal external public information available to the affected repository.

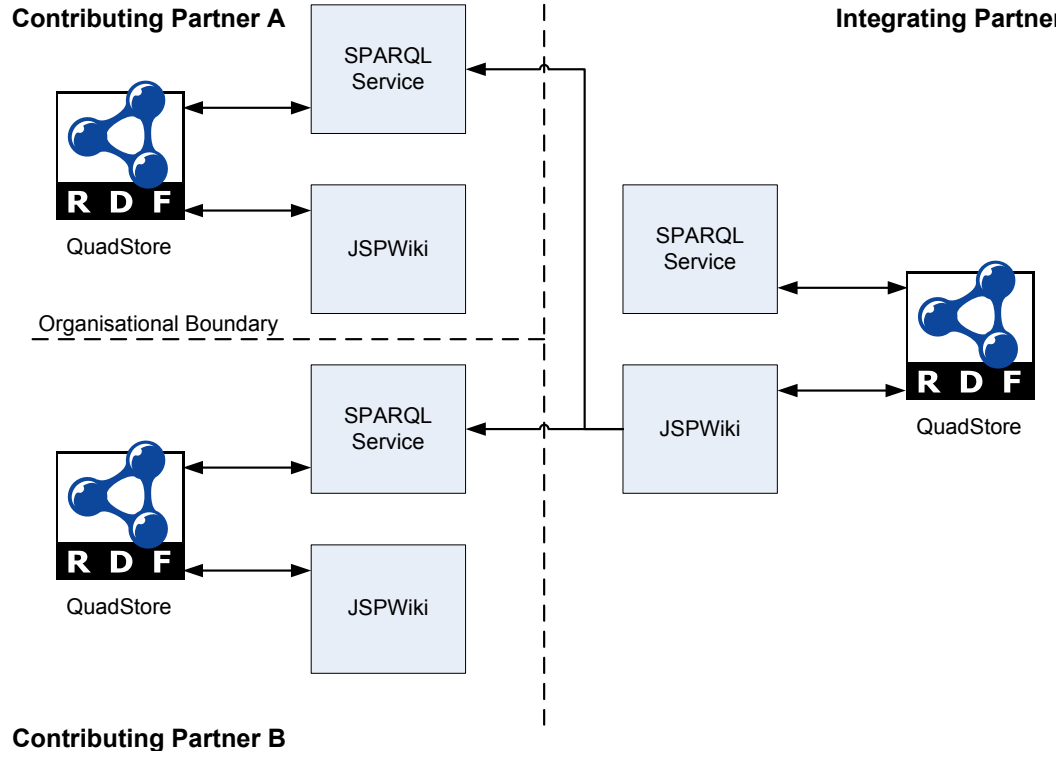

FIGURE 4.16: EC IST Federation Scenario

Figure 4.18 shows the scenario. When the host repository discovers a broken signature, it needs to establish whether the remainder of a version history can be salvaged. The repository should perform the following steps:

1. Determine the author of commit with the broken signature;
2. Check if author has made other commits for same document;
3. Check if author is listed a committer to project (DOAP);
4. Search for other commits with same author and check status;
5. Generate report of author for repository admin.

This scenario is a prime example of where the key benefits of the Semantic Web can be leveraged: semantic reasoning and knowledge federation. The repository needs to go beyond its current knowledge-base and discover additional information like FOAF and DOAP descriptions that it can use to support decisions to be made by a developer or repository administrator[34]. The Jena 2 inference rules used in this scenario can be found in Appendix D.1.1 with an example report in Figure D.4.

[34] An extension this scenario could see the searching of mirror repositories for backup metadata. If such metadata is valid, then it can be imported like in Section 4.7.1. If invalid, then it reveals a larger problem that must be addressed manually. This extension has not been implemented.

One difficulty with this scenario is that it is unlikely that published FOAF and DOAP information is going to be particularly reliable. Users will have links to their FOAF descriptions in a DOAP document located on the project webpage.

4.7.4 EC IST Signature Recovery

Our last scenario looks at signature failure during the integration of code from several Contributing partners. Unlike Section 4.7.3 which had little information (simple FOAF and DOAP description) to determine the remaining trust after signature failure, this scenario a range of additional information at its disposable.

Partners in an EC IST consortium will each have repositories kept within their own trust domains. Information, including source code will be made available to other partners only under strict guidelines outline in the project's Consortium Agreement. Since partners will be providing secure services to one another, each will have their own Certificate Authority. It will also be very clear which partners require access to data at any point in time as well as being able to verify where information has been transferred to and for what purpose.

```
:warrant {
   :warrant a swp:Warrant.
   :warrant swp:authority
   <mailto:erw@it-innovation.soton.ac.uk>.
   :warrant swp:sourceProject
   <http://http://www.ecs.soton.ac.uk/~erw/AcmeGrid/doap.rdf>.
   :warrant swp:signatureMethod
   swp:JjcRdfC14N-rsa-sha1.
   :warrant swp:signature
   "E2a...ylV"^^xsd:base64Binary.
   :warrant swp:assertedBy :warrant.
   :G1 swp:assertedBy :warrant.
   :G1 swp:digestMethod swp:JjcRdfC14N-sha1.
   :G1 swp:digest "YjR...hNz"^^xsd:base64Binary.
   :G2 swp:assertedBy :warrant.
   :G2 swp:digestMethod swp:JjcRdfC14N-sha1.
   :G2 swp:digest "NmM...2NW"^^xsd:base64Binary.
}
```

FIGURE 4.17: IST Warrant Graph that includes source project

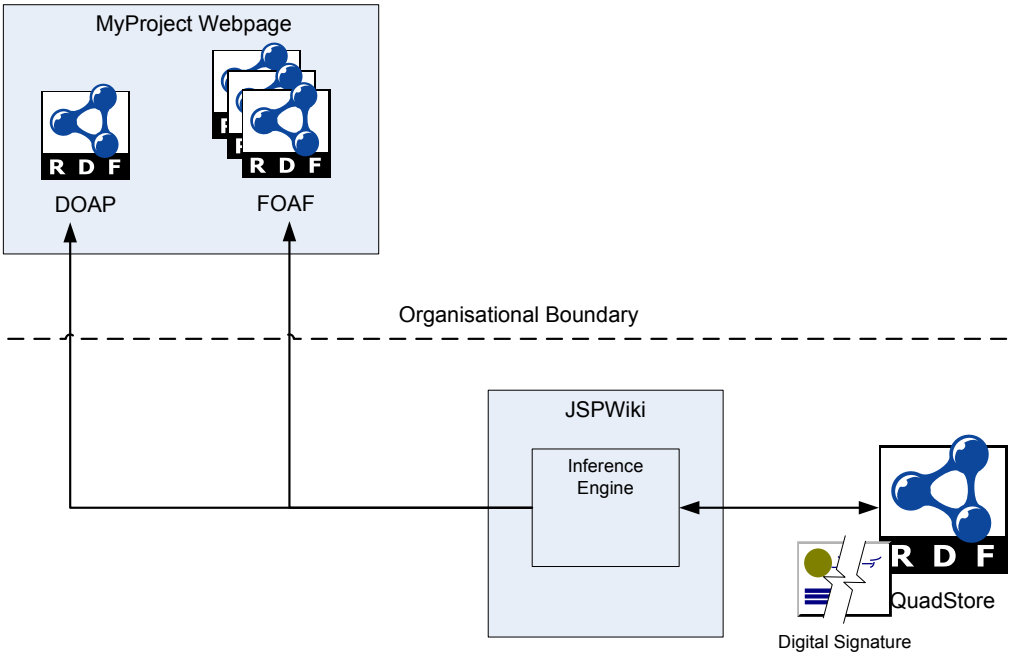

FIGURE 4.18: FLOSS Signature Recovery Scenario

Figure 4.19 gives an overview of the scenario which extends what we described in Section 4.7.2. The Integrating Partner has discovered a signature failure and needs to resolve the issue. The repository should perform the following steps:

1. Determine the author of commit with the broken signature;
2. Check if author's certificate is signed by a CA known to the project (Certificate);
3. Check if author is listed as working in the workpackage the document is part of (FOAF, DOAP);
4. Check if author has committed in local repository;
5. Request metadata about any commits in Contributing Partners' repository;
6. Generate report of author for repository admin;
7. Provide override option (new Digital Signature).

While many of the steps here are similar to those specified in Section 4.7.3, step seven is different. Once the repository administrator at the Integrating Partner has

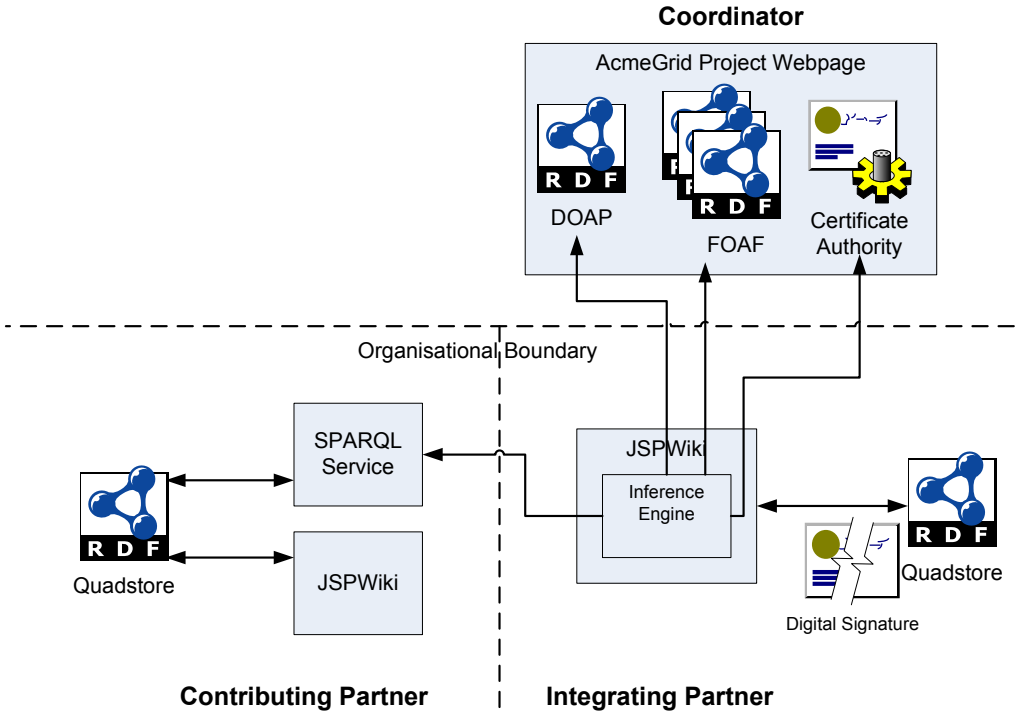

FIGURE 4.19: EC IST Signature Recovery Scenario

reviewed the integrity report, they have a choice to ignore the error and override with a new signature. Note that this is only possible for the local repository; if the offending signature failed in one of the Contributing Partners' repositories, then the override signature must be done by the Contributing Partner out-of-band. The Jena 2 inference rules used in this scenario can be found in Appendix D.1.2 with an example report in Figure D.8.

4.8 Summary

In this section we gave our rationale for Document Provenance and how it models version control. We also described its realisation as an OWL DL ontology. While Document Provenance forms the core of our DL framework it does not itself create provenance. For this we have used Named Graphs, an compatible extension to RDF that provides a natural way to record provenance in RDF. Named Graphs also provide a convenient mechanism that allow us to attach digital signatures of other RDF graphs. Digitally signed DP forms the core of a version control structure that not only enforces committer IPR attribution, but is also self-contained, non-repudiable, and immutable.

Our online collaborative tool, based on our work with Named Graphs and digital signatures, defines an example collaborative distributed software engineering environment. To show how this framework can be used to satisfy the the questions useful to our case studies in Section 3.5, we have defined as set concrete federation scenarios that attempt to demonstrate Semantic Web knowledge federation coupled with semantic reasoning, that goes beyond the capabilities of the modern RDBMS. Performance results of these scenarios can be found in Section 5.1.2.1.

Despite our successful work with Semantic Web technology, there still exist issues that need to be resolved in further work. While the *conservative canonicalisation* approach we employ for our RDF digital signature solution is viable, it limits the wider variety of RDF graphs that exist on the Semantic Web. The way RDF instances are decomposed and selected for signing still pose issues. In the next chapter we evaluate our online collaborative tool, RDF digital signature solution, and Semantic Web technology.

Chapter 5

Evaluation

In the previous chapter we described our DL-based provenance design, its security considerations, and its use in our online collaborative tool. We went on to define some federation scenarios that could be used to demonstrate the value of Semantic Web technology beyond the capabilities of a standard RDBMS. This chapter evaluates the research described in this thesis and includes quantitative and qualitative analysis of Semantic Web and RDBMS technology.

Quantitative experimental results included in this chapter were conducted on a Core^{TM}2Duo 2.00Ghz machine with 2GB RAM, running Windows XP SP2. The effective heap size available to Java was set to 1GB, which became essential when measuring results for the federation scenarios and OWL DL entailment performance.

5.1 Semantic Web Evaluation

During the course of our research we have been exposed to a wide array of Semantic Web concepts and techniques that we believe are valuable in improving distributed collaborative software development. Despite demonstrable advantages such as knowledge federation, there are still questions regarding the performance of Semantic Web toolkits against their modern RDBMS counterparts. If, for example, we were to select a simple RDBMS (embedded without optimisation or indexing), how does our Semantic Web-based version control approach compare?

5.1.1 Semantic Web Performance

In this section we measure various aspects of Semantic Web performance of based on representative Java™source documents that has been committed into our semantic version control system. We have used real-world source code from the Taverna project some of whose classes (Workbench[1]) provides a reasonably large version history (90 versions as of 20th February, 2007).

5.1.1.1 Data Models

RDBMS and Semantic Web technology take distinctly different approaches to data modelling. An RDBMS contains a collection of tuples of attributes whose structure is based on a *schema*. An RDBMS schema is a highly structured set of relationships between attributes that is also highly static; once a schema has been developed and deployed, it cannot be dynamically changed. Developing a schema takes a great deal of effort on the part of the developer, although a schema in third-normal form [Kent (1983)] should be extremely efficient when used in conjunction with indexes.

The RDF data model is, in contrast, more flexible than the RDBMS approach, primarily due to its so-called semi-structured nature. Its graph-based structure is therefore less strict than an RDBMS schema and more extensible. For example, it is quite simple to increase the size of an RDF graph simply by adding new relationships that form triples; any entailments on these triples will be based on an ontology which can also be changed since it also follows the same data model. As we noted in Section 2.5.3, the purpose of OWL ontology language is to give meaning to RDF. Typically an OWL ontology is used to process RDF and produce simple type classification. The main argument behind OWL DL inferencing is so that different vocabularies can be mapped together based on categorisation hierarchies. As we will see in Section 5.2, the additional classification produced by OWL DL entailments is a poor return on investment.

While the RDF data model is in principle more flexible and open to change, we will see that this flexibility comes a price. Its semi-structured nature makes it very difficult to store efficiently, which leads to performance issues in read and write operations.

[1] http://taverna.cvs.sourceforge.net/taverna/taverna1.0/taverna-workbench/src/main/java/org/embl/ebi/escience/scuflui/workbench/Workbench.java?view=log

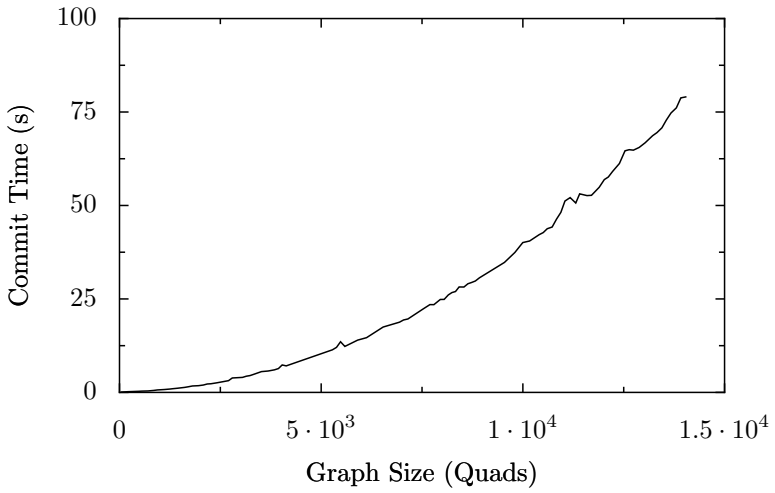

FIGURE 5.1: NG4J Commit Performance using HSQLDB

To determine the performance of our Semantic Web toolkit of choice, NG4J, we set out to measure its ability to store and retrieve RDF compared to a simple RDBMS. To ensure a like-for-like comparison, we used the same RDBMS in its "native" mode and as a backend to NG4J. With this in mind we chose HSQLDB[2] as the RDBMS, which is one of the many databases supported by NG4J. HSQLDB is an all-Java™ database engine that is extremely useful in embedded deployments.

5.1.1.2 Storage

Most Semantic Web toolkits process all RDF in memory, fetching and storing RDF from flat files located on the local file system or on the Web[3]. More advanced configurations use RDBMS-based persistent storage so that SQL-like languages for RDF can be used to reduce the amount of memory used to process RDF. For small amounts of RDF (100-1000s of triples) in-memory is still quite usable. Unfortunately, once more advanced SW concepts such as inferencing are introduced, memory usage increases, even if an RDBMS-triplestore is used; this is because all current inference strategies must be performed in memory, no matter the size of the underlying dataset.

Figure 5.1 shows the time taken for committing 90 successive DP instances describing the Taverna **Workbench** class. As can be seen from these results the taken is using NG4J is non-linear; this suggests that there is a performance hit compared to using an RDBMS natively which we would expect to be linear in

[2]http://hsqldb.org/.
[3]FOAF and DOAP descriptions are common examples where RDF will be serialised.

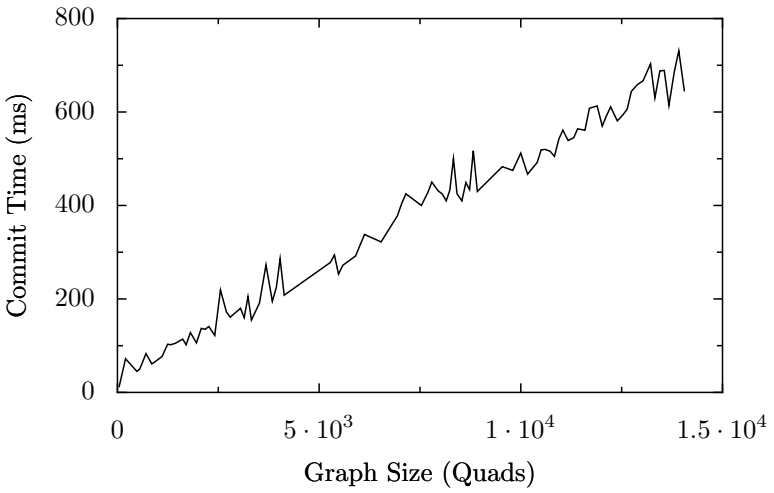

FIGURE 5.2: Commit Performance using HSQLDB Native

shape. If we were to "replay" the NG4J commits into HSQLDB natively we see a proportional relationship between the time taken and size of dataset (Figure 5.2). Given the maturity of RDBMS technology a linear result for HSQLDB is not unexpected and also shows that embedded small-scale RDBMSs are just as capable as their client-server counterparts (MySQL, PostgreSQL, Oracle).

It is quite obvious at this point that HSQLSB performs far better on its own than with NG4J at the application layer. It is highly likely that is in part due to the immaturity of NG4J and Semantic Web technology in general. RDBMS technology has had several decades of development to achieve the performance we see today. Another reason for NG4J's reduced performance is the schema that is used to represent the RDF data model in SQL. This is clearly sub-optimal and needs reviewing; however, strategies will be constrained by the semi-structured nature of RDF, making it difficult to be optimised in an RDBMS schema since it is too generic.

5.1.1.3 Querying

Querying is a vital feature of any information storage system, semantic or otherwise. At the time RDF and OWL were published as Recommendations by the W3C in early 2004, no standard query language for RDF existed; this left developers to create their own based on SQL or similar. RDQL has been one of the most popular languages, some parts having become part of the SPARQL RDF Protocol and Query and Language.

Both SQL and SPARQL can be seen as query languages that work on objects; SQL for querying attributes of objects in an RDBMS; SPARQL for querying relationships between objects in a knowledge-base Melton (2006). The closest equivalent to these relationships in SQL is the foreign key attribute in an RDBMS relation.

SPARQL is designed to query collections of triples known as *graph patterns* from one or more graphs. The SPARQL Dataset [Prud'hommeaux and Seaborne (2007)] defines the so-called *background graph* and zero, one or more Named Graphs. As a query language it has many advantages over SQL, some that derive from the underlying RDF data model, others from its ability to query from multiple logical data sources (different RDF graphs); it is the latter advantage that has been leveraged by us for knowledge federation.

Some recent work by Prud'hommeaux (2006) has investigated the possibility of mapping SPARQL directly onto MySQL, effectively turning SPARQL into a universal query language that supports both RDF and RDBMS data structures. Others are attempting to bridge the gap between OWL and relational databases [Motik et al. (2007)].

```
SELECT * FROM books
 WHERE title = 'book1';
```

FIGURE 5.3: Simple SQL Query

To illustrate the differences between SQL and SPARQL, consider the queries in Figure 5.3 and Figure 5.4. Both queries perform a select, however, the method for each is very different based on their respective data models. Rather than working against a strict schema, SPARQL allows the developer to write queries that work on relationships and unbound variables. Despite this SPARQL maintains a vaguely SQL syntax that makes writing queries relatively simple for those who know SQL and RDF. Queries are constructed based on graph patterns which can easily traverse relationships, something that is more difficult in SQL without the use of JOIN between different tables.

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

FIGURE 5.4: Simple SPARQL SELECT Query, taken from latest SPARQL Working Draft, March 2007

SPARQL is not limited to simple selects based on graph patterns. Entire graphs can be returned using the CONSTRUCT and DESCRIBE constructs (Figure 5.5), whilst existential tests can be made using ASK (5.6). While CONSTRUCT and DESCRIBE both return an RDF graph, in a DESCRIBE request the structure of the returned data is not prescribed by the SPARQL query, rather it is determined by the SPARQL query processor.

```
CONSTRUCT {?s ?p ?o}
WHERE
 {
   GRAPH ?g {
              ?s ?p ?o
            }
 }
```

FIGURE 5.5: Example SPARQL CONSTRUCT Query

It is important to note that the query in Figure 5.5 will produce a graph is a logical merge of all graphs that exist in the knowledge-base. Since a merge is performed, all information about the originating graph is lost without additional queries. This became particularly noticeable when we were implementing the federation scenarios in Section 4.7.

```
PREFIX dp: <http://grid.cx/dp/1.0/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
ASK WHERE
 {
   GRAPH ?g {
              ?doc rdf:type dp:Document
            }
 }
```

FIGURE 5.6: Example SPARQL ASK Query

One interesting but subtle aspect of the SPARQL ASK construct is the issue of monotonicity. As we noted in Section 2.4.5, the Semantic Web is considered a monotonic logic and follows the Open World assumption. It can be argued that SPARQL ASK simulates *negation as failure* and thus a non-monotonic operation; if it were monotonic then the return value for a null result would be *unknown*. It is likely that ASK was added to SPARQL for the convenience of yes or no questions, despite its implications for the underlying logic.

While SPARQL does not support nested queries like those found in SQL, it can emulate the same effect using the UNION construct. Prud'hommeaux (2006)

```
SELECT * FROM ng4j_quads;
```

FIGURE 5.7: SQL Select *

presents an interesting example of the use of the SPARQL UNION construct; whereas the SQL example used several subqueries SPARQL performs a number of UNIONs which reduce the size and complexity of the query.

5.1.1.4 Query Performance

```
SELECT * FROM ng4j_quads
where
subject='https://localhost:8443/webdav/taverna/
taverna-workbench/org/embl/ebi/escience/scuflui/
workbench/Workbench/1/45/Workbench.java';
```

FIGURE 5.8: SQL Select DP Instance

Our strategy for evaluating the performance of SQL and SPARQL is to use some simple queries to access the DP instances generated in Section 5.1.1.2. For SQL we have used the queries shown in Figure 5.7 and Figure 5.8; in the first query we select all quads from the knowledge-base, whilst in the second we select the metadata description for a specific source code document. To ensure a fair comparison, HSQLDB does not use indexes to increase SQL performance. Indexing is a notable feature that is lacking from Semantic Web toolkits.

```
SELECT *
WHERE
 {
  GRAPH ?g {
             ?s ?p ?o
           }
 }
```

FIGURE 5.9: SPARQL Select

For SPARQL we have used the queries shown in Figure 5.9, Figure 5.5, and Figure 5.6 since they are the three main SPARQL query types that we have used in our federation scenarios (Section 4.7).

Figure 5.10 and Figure 5.11 show results of HSQLDB SQL and NG4J SPARQL respectively, based on our DP instances. On the whole there is little difference in the query times between HSQLDB SQL and NG4J SPARQL. HSQLDB SQL is

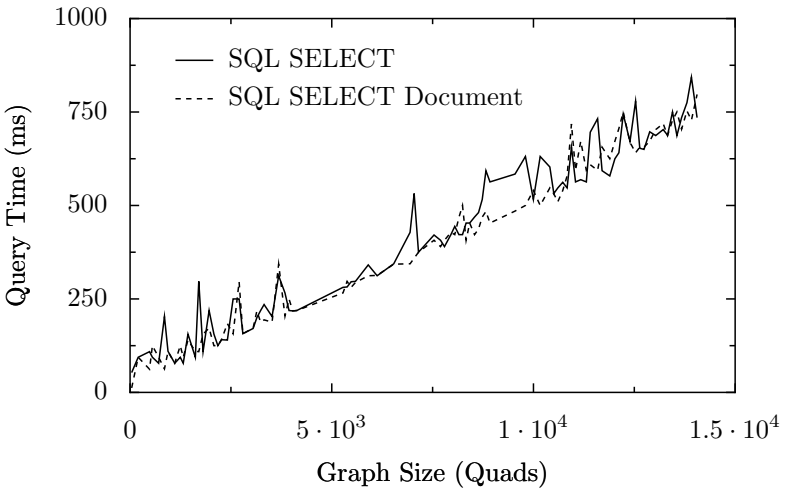

FIGURE 5.10: HSQLDB Native SQL Query Performance

clearly linear while NG4J SPARQL is non-linear which is expected given the RDF data model.

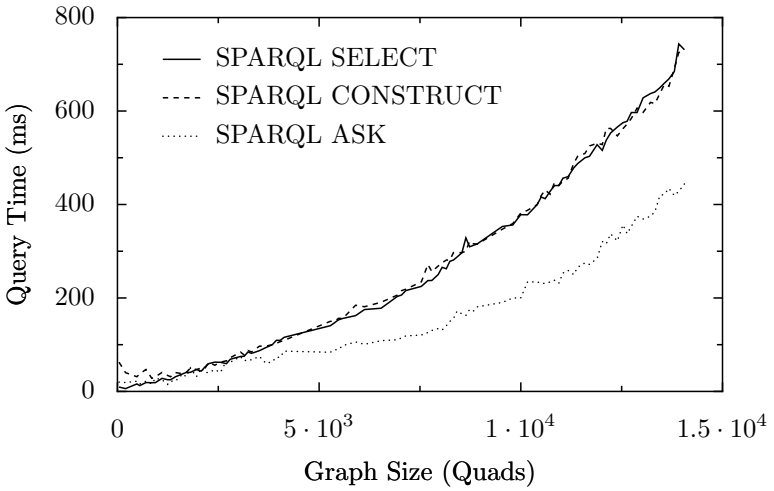

FIGURE 5.11: NG4J HSQLDB SPARQL Query Performance

5.1.2 Federation

Federation is a key enabler for collaboration in distributed environments. We have already noted that the ability to gather information from multiple sources is beneficial and crucial for our case studies (Section 3.1). Our approach in Section 4.7 attempts to demonstrate how Semantic Web federation satisfies the case studies and questions outlined in Section 3.5.

Below we present a set of SPARQL queries used in the federation and signature recovery scenarios. In our implementation, each query has been encapsulated as a Jena 2 builtin functor that can be called within a Jena 2 rule. Figure 5.12, Figure 5.13, and Figure 5.14 implement the builtins `doapAuthorKnown`, `remoteAuthorKnown`, and `listCommits` respectively. The declarative nature of the Jena 2 rule language makes introducing new functionality relatively simple. Managing rules can be problematic, however, since it is difficult to understand the data flow that causes rules to fire.

Figure 5.12 is designed to check a committer's membership of a project based on workpackage and CA, hosted on the project's DOAP description[4]. The OPTIONAL keyword optionally matches graph patterns that do have solutions, otherwise ignore and attempt to satisfy original pattern. This is a new feature that was not available in SPARQL's predecessor, RDQL. The use of OPTIONAL in our federation scenarios means that the same query can be used in both the FLOSS and IST federation scenarios that each query DOAP descriptions. This query is used in the `doapAuthorKnown` rule builtin.

```
PREFIX doap: <http://usefulinc.com/ns/doap#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX swp: <http://www.w3.org/2004/03/trix/swp-2/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT *
WHERE
 {
   ?author swp:certificate ?cert .
   OPTIONAL { ?author doap:workpackage ?wp . }
   OPTIONAL { ?s doap:knownCA ?ca . }
 }
```

FIGURE 5.12: SPARQL query on DOAP description

Figure 5.13 shows a SPARQL ASK query that verifies whether a committer (`swp:authority`) has committed a Java™ class based on matching against the digital signature. Here we see where the SPARQL GRAPH construct comes to into its own. Since each Warrant graph (Section 4.6.4) that holds the signature to original graph is distinct and logically separate, we must query for both graphs. Firstly, we find the Warrant graph that has the same signature creator (`swp:authority`); we can then find out which graph the Warrant graph *asserts* with the signature. Secondly, we query the second graph and test for its class description. This query is used in the `remoteAuthorKnown` rule builtin.

[4]http://www.ecs.soton.ac.uk/ erw/AcmeGrid/doap.rdf.

```
PREFIX swp: <http://www.w3.org/2004/03/trix/swp-2/>
PREFIX dp: <http://grid.cx/dp/1.0/>
ASK WHERE
 {
   GRAPH ?g
    {
     ?auth swp:authority
     <mailto:erw@it-innovation.soton.ac.uk> .

     ?g2 swp:assertedBy ?g .
    } .
   GRAPH ?g2
    {
     ?doc dp:hasClass
     <net.sf.taverna.tools.Bootstrap>
    }
 }
```

FIGURE 5.13: SPARQL ASK query used in federation scenarios

Figure 5.14 is designed to return a list of document descriptions committed by a known committer. This has been used in the `listCommits` Jena 2 builtin so that in the IST signature recovery scenario, the administrator can query contributing partners to see whether the creator of the broken signature has committed any other documents. Any additional documents are listed in the validity report using the `dp:knownCommits` relationship. This SPARQL query takes advantage of the DISTINCT construct which limits results to those that are unique only.

In addition to using SPARQL to query data in our knowledge federation scenarios, we have also used Horn clause-based inference rules to automate the execution of our signature recovery scenarios. Inference rules take a declarative approach that can also be found in languages such as Prolog and to some extent Java™ (RuleML). They are, however, difficult to author and difficult to manage. Depending on the algorithm used they can also be computationally expensive. We will discuss the performance of our custom rules in the next section and OWL DL in Section 5.2.3.

Another issue when accessing data from NG4J is that, unlike SPARQL, the Jena 2 rule engine does not understand the existence of Named Graphs. At present it still only works on standard RDF graphs, placing inferences into a separate graph that can be optionally added to the original graph if required.

```
PREFIX swp: <http://www.w3.org/2004/03/trix/swp-2/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dp: <http://grid.cx/dp/1.0/>
PREFIX java: <http://simile.mit.edu/2004/09/ontologies/java#>
SELECT DISTINCT ?doc
WHERE
 {
  GRAPH ?warrant
   {
    ?warrant swp:authority
    <mailto:erw@it-innovation.soton.ac.uk>
   } .
  GRAPH ?graph
   {
    ?doc dp:maker ?authority
   }
 }
```

FIGURE 5.14: SPARQL DISTINCT query used in federation scenarios

5.1.2.1 Federated Scenario Performance

To show the performance of our federation scenarios (Figure 4.15 and Figure 4.16), we used the same Taverna Workbench Java™ class dataset used in Section 5.1.1.2 and Section 5.1.1.3 and added the necessary RDF digital signatures. To ensure that it also worked with a different dataset, we also tested using the Taverna `net.sf.taverna.tools.Bootstrap` Java™ class. Results presented here are for the Taverna Workbench class.

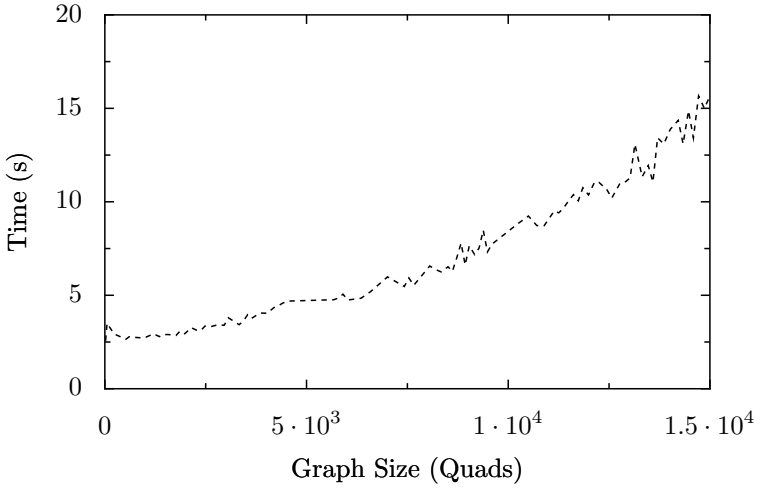

FIGURE 5.15: Federated Retrieve Document Metadata Performance

Figure 5.15 shows results for returning a single version of the Workbench class from our SPARQL GRIA service. We can see that the graph shape is clearly non-linear, although not exponential. The main reason for this, as can be seen by the time taken, is the cost of using secure SOAP to transport RDF and the inefficiency of NG4J SPARQL queries.

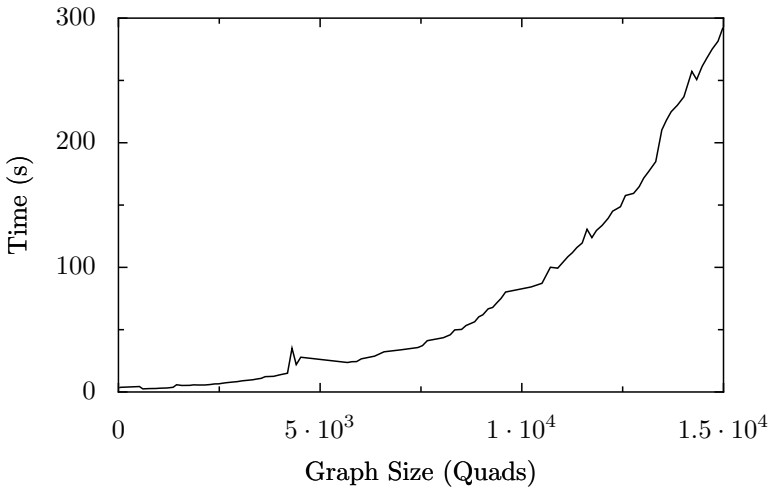

FIGURE 5.16: Federated Retrieve Document History Metadata Performance

Returning complete version histories is nearly ten times slower (Figure 5.16) more due to SPARQL query inefficiencies than the SOAP invocations. In this case, the web service must do much more work searching for all versions and packaging them for transport. The worse than linear shape, therefore, is expected.

Turning to our signature recovery scenarios, we measured the performance for the Jena 2 inference engine to process the rules defined in Appendix D.1.1.1 and Appendix D.1.2.1. While our rules are relatively simple, the nature of the RETE [Forgy (1982)] algorithm can still be computationally expensive.

Figure 5.17 shows the results for FLOSS signature recovery. While the results are somewhat erratic, even after several successive runs, the graphs shape can still be seen to as more than linear. It can also be seen that all inferences are complete within ten seconds. This is mainly due to the fact that the FLOSS signature recovery scenario does not attempt to access any remote SPARQL services located at another repository.

The IST signature recovery, on the other hand, is distinctively exponential in shape. There are two reasons for this, the first being the use of Jena 2 rule builtins that call a remote SPARQL web service to discover if the committer has committed the same class before (`remoteAuthorKnown`, Figure 5.13) and to

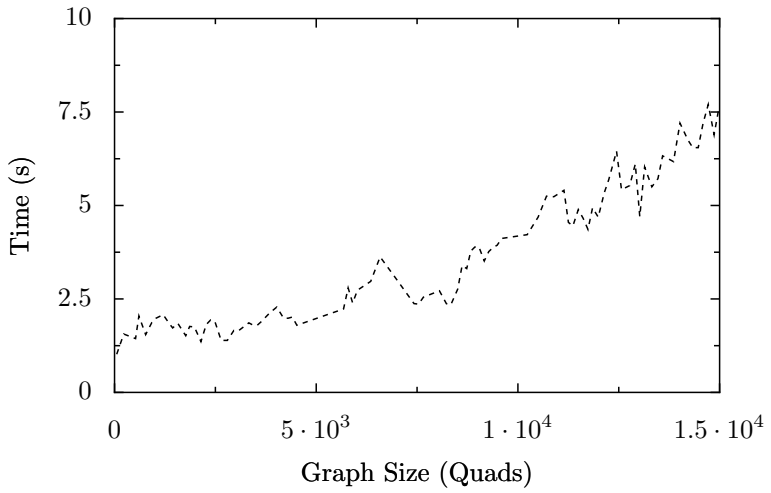

FIGURE 5.17: FLOSS Signature Recovery Performance

retrieve a list of other good commits (`checkDocument`, Figure 5.14). The second is the multiple SOAP invocations that the inference engine must make and the associated processing time at the SPARQL web service.

Despite the relatively slow performance of the federation and signature recovery scenarios, we can see that federation across different data sources using procedural (Java™) and declarative (Jena 2) rules is possible. The use of secure SOAP also definitely impedes performance, although in the case of IST projects it is crucial for effective collaboration between partners.

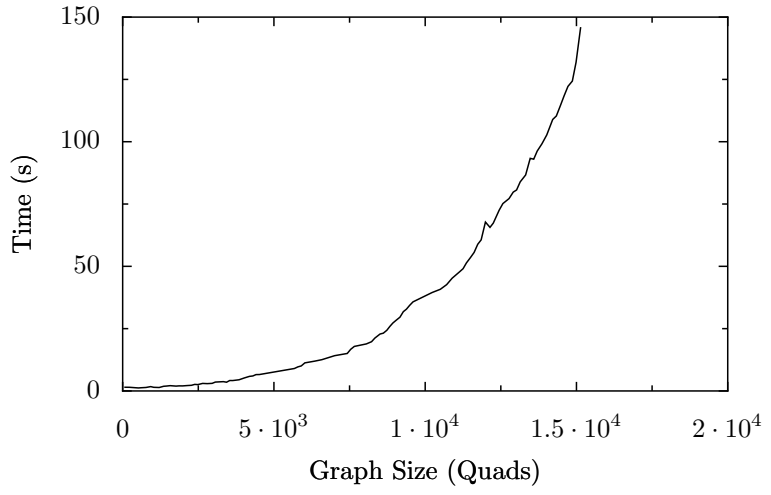

FIGURE 5.18: IST Signature Recovery Performance

5.1.3 Security

In Chapter 1 and Chapter 3 we argued that our case studies required a security approach where trust was vested in the software repository's internal structure (metadata) rather than the remote host. By taking this approach we could avoid the monopolies that exist in current FLOSS hosting providers, for example, SourceForge. Our solution to this problem has been to tightly couple trust into the version control structure so that its integrity is self-asserting, and that since developers must be part of the commit process, they are made accountable.

5.1.3.1 SWP Performance

Earlier implementations of SWP were based on DBin's RDF signature mechanism[5]. We initially tried the TriQL [Bizer (2004a)] query language, based on Jena 2's RDQL [Seaborne (2004)]. TriQL is a small and relatively efficient query engine that supports Named Graphs; unfortunately it does not scale well. The reference implementations of SPARQL in Jena 2, ARQ, showed some promise; the SPARQL draft specification[6] had only just been released and only spoke of the SOURCE of an RDF triple; Named Graphs did not arrive until January 2005[7].

The use of Named Graphs, an expressive query language (SPARQL) and inference puts our implementation apart from other work. Tummarello et al. (2005) claim that the use of RDF reification somehow causes the signature to be *closer* to the RDF graph it signs. XML Signature has a similar approach with *encapsulated* signatures, where the content is actually held as part of the signature structure.

Another well used method, used by Dumbill (2002) and XML Signature, and our work is *detached* signatures. Detached signatures are separate to the content they sign, for example, a file in a database. Our RDF digital signature solution is detached in that it is held in a different Warrant Named Graph; since there is a semantic connection which does not exist with RDF reification, our approach can be seen as flexible and extensible, something not possible with Tummarello et al. (2005).

Although digital encryption of data can be an expensive process it is relatively simple compared to generating digital signatures. The object to be signed must

[5] We removed the need for the RDF Reification vocabulary.

[6] SPARQL Query Language for RDF W3C, Working Draft 12 October, 2004, available at http://www.w3.org/TR/2004/WD-rdf-sparql-query-20041012/.

[7] Available at http://www.w3.org/2001/sw/DataAccess/rq23/#choosing.

have its digest recorded (SHA-1, SHA-224, SHA-384, or SHA-512); a manageable signature structure must be constructed that includes the digest of each RDF graph; this is then signed. As we noted in Section 4.4.1, RDF does not have a canonical form like XML, so digital signature generation is non-trivial. In Section 4.6.4 we described our RDF digital signature solution which is now part of the SWP Framework in NG4J. This solution uses a *conservative canonicalisation* approach based on the algorithm detailed in Carroll (2003), that forbids the use of blank nodes.

Here we present results for the performance of our RDF digital signature mechanism using Document Provenance instances based on the Taverna dataset we used in Section 5.1.1.2, Section 5.1.1.4,and Section 5.1.2.1. Results include the processing time for canonicalisation as well as generation and verification of digital signatures. Since our RDF digital signature mechanism is incapable of reliably canonicalising the Petersen Graph, we have not included any results given that such results cannot be validated properly.

5.1.3.2 DP Instances

DP instances provide an interesting dataset to test our RDF signature solution because of the varying size and structures that exist in each instance. While these structures include sub-graphs, RDF Semantics [Hayes (2004)] define how these sub-graphs merge together; this means the introduction of sub-graphs should not adversely affect the performance of our solution.

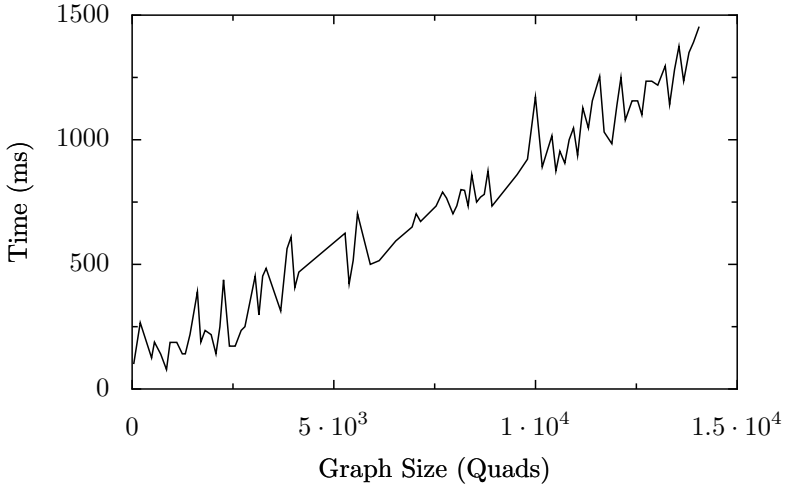

FIGURE 5.19: Carroll's Algorithm Performance

Figure 5.19 shows the performance of Carroll's algorithm over our DP instances. Results suggest that graph size and time taken are proportional, which is not surprising given that our *conservative canonicalisation* approach negates the use of blank nodes (Section 4.4.1.2).

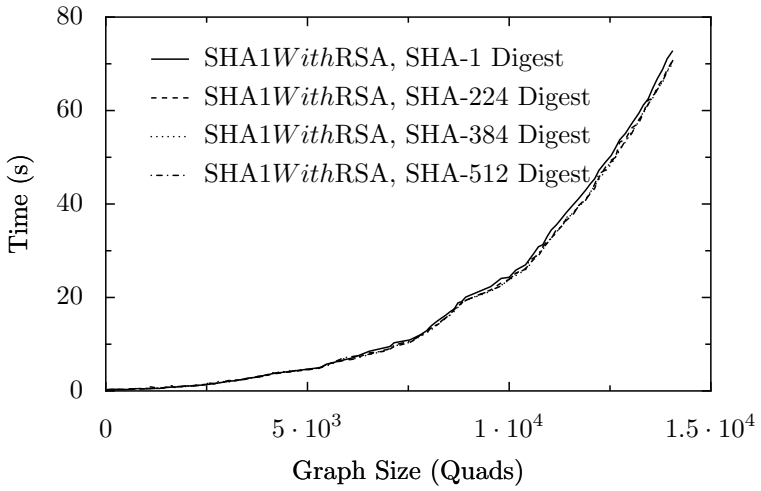

FIGURE 5.20: SWP SHA1*With*RSA Performance

While Carroll's algorithm follows a linear shape, the actual generation of digital signatures using this algorithm do not. As can be seen in Figure 5.20, over increasing graph sizes our RDF digital signature solution is at best non-linear. Since we have already see that Carroll's algorithm is linear, this is most likely due to the increased computation when generating the secure hashes and RSA signature.

5.2 Logic Evaluation

In this section we evaluate the various logics that we have encountered during our research namely, SQL, Description Logics and Prolog. While we have not directly worked with Prolog or Logic Programming, we believe that including them in this evaluation given its prevalence in artificial intelligence.

5.2.1 Differences

While on the face of it SQL, DL and Prolog are clearly different, they are all based on logics that are defined based on their expressiveness and computational complexity. Each approach is increasingly expressive, yet becomes more computationally complex as can be seen in Figure 5.21.

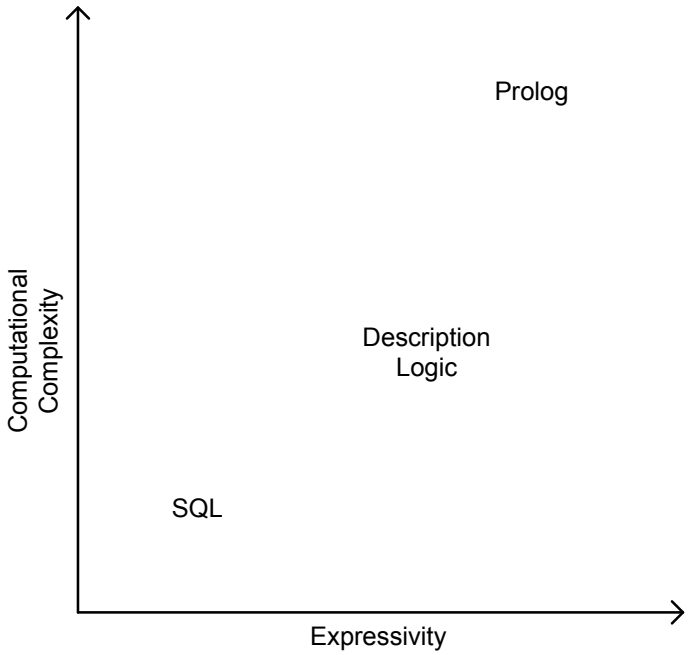

FIGURE 5.21: Logic Expressivity

SQL, which is the most common RDBMS query language is a simple and efficient language that is the least expressive logic. Like Codd's relational algebra and relational calculus it is decidable [Date (2000),Date (2006)], and capable of processing queries in linear time as we have seen in Section 5.1.1.3.

Description Logic sits comfortably between SQL and Prolog. While DL is a subset of FOL it uses a decidable subset that has made it an attractive logic for knowledge representation (see Section 2.4.1). Computationally complete, several algorithms exist to derive entailments on knowledge-bases. These algorithms, however, tend to be relatively inefficient as we shall see in Section 5.2.3. Examples of DL such as OWL DL provide an array of core entailments that provide automatic categorisation, a feature that is missing from SQL.

The last and most expressive logic related to our research is Prolog. Commonly used in artificial intelligence, it is made up entirely of Horn clauses [Horn (1951)]. These clauses can also be used to construct rules that perform a similar function to SQL when creating new relations[8]. The range of constructs that Prolog supports makes it an extremely versatile and expressive language; it can be used to make interesting programs based around business rules [Ross (2003); Walker (1990)] and solve relatively complex problems. Prolog can be used to implement DL

[8]http://cs.wwc.edu/KU/PR/Prolog.html.

entailments, and other FOL including F-Logic. This high expressivity, however, means Prolog and related Logic Programming (LP) languages are known to be undecidable [Covington et al. (1996)].

5.2.2 Application Domains

We have already seen the application of SQL in our online collaboration tool. SQL is primarily used in data intensive, real-time systems in the Enterprise. It is not difficult to find real-world examples of where RDBMS technology and SQL enabling more advanced systems whether for multi-tier web applications or web browser enhancements. Enterprise Java™ based on J2EE is still popular for multi-tier systems, while attempts to build stateful web services are making progress using WSRF and WS-ResourceTransfer. The Mozilla Foundation recently announced that Firefox 3 will include support for off-line web applications[9], including an SQLite database that will index bookmarks among other things.

As we have demonstrated with our online collaborative tool, one of the main uses of DL is in knowledge representation. Using a DL language for our DP ontology meant we had a way to not only have a machine readable way of describing the provenance of source code, but also have a way to *potentially* infer new knowledge from the software repository, an issue we raised in our research statement (see Section 1.3).

Ontologies are becoming more main stream, even if they are not strictly DL-compatible; FOAF and DOAP are obvious examples where popular ontologies are helping developers create machine readable metadata that can be federated. Industry is also starting to see the benefit of Semantic Web-based knowledge representation, particularly in the domain of bio-informatics [Stevens et al. (2003); Sabou et al. (2005)].

Prolog is mostly found in specialised systems related to artificial intelligence. Even today it is not in mainstream use since a lot of modern programming languages including Java™ able to replicate the declarative and rule-based innovations found in Prolog[10]; JSR-94 [Toussaint (2003)] has become an industry standard with implementations including Jess [Friedman-Hill (2003)].

[9] http://www.informationweek.com/software/showArticle.jhtml?articleID=198000591.

[10] http://today.java.net/pub/a/today/2004/08/19/rulingout.html.

Revival of a kind has taken place with the resurgence of F-Logic under initiatives such as the Web Services Modelling Ontology (WSMO), its sister languages Web Service Modelling Language and execution environment, Web Services Modelling eXecution (WSMX). As we mentioned in Section 2.4.3, F-Logic is an extension to Prolog that introduces an object-orientated approach to knowledge representation. While WSMO claims to be compatible to DL, it is difficult to see how two different approaches to knowledge representation can be reconciled [Horrocks et al. (2005)].

5.2.3 Inference Performance

In this section we evaluate the performance of several inference engines and their OWL implementations. As we shall see, various factors affect the performance of inferences including the algorithm used and the complexity of logic (OWL DL, Mini, Micro). We have not included performance measurements for Prolog given the declarative approaches that are now available in Java™. We have also not included SQL performance; as we have seen in Section 5.1.1.3 SQL performance with our semantic version control dataset is reasonably fast (linear time) despite the absence of indexing in HSQLDB. Value comes from relatively complex queries across different tables in the RDBMS.

In general, Prolog returns solutions in polynomial time, although a Prolog program cannot guarantee to complete. Prolog's inference strategy, sometimes known as "backward chaining" (see Section 2.4.2.1) or goal-seeking contributes to its computational complexity. Previous studies, however, have found that in general performance in Prolog can be increased by careful reordering of sub-goals [Escalante (1993)].

The potential for non-terminating queries makes Prolog not an ideal choice for providing inference facilities in distributed collaborative software development. Federation might be possible, but answering queries needs to be done quickly, otherwise users will go elsewhere.

5.2.3.1 DL

At present Description Logics and to a lesser extent F-Logics prevail on the Semantic Web. We have found that DL reasoners such as Pellet [Parsia and Sirin (2004)] and RacerPro [Haarslev and Mller (2003)] create an excessive number of triples; this may be partially alleviated by more restrictive OWL subsets such as

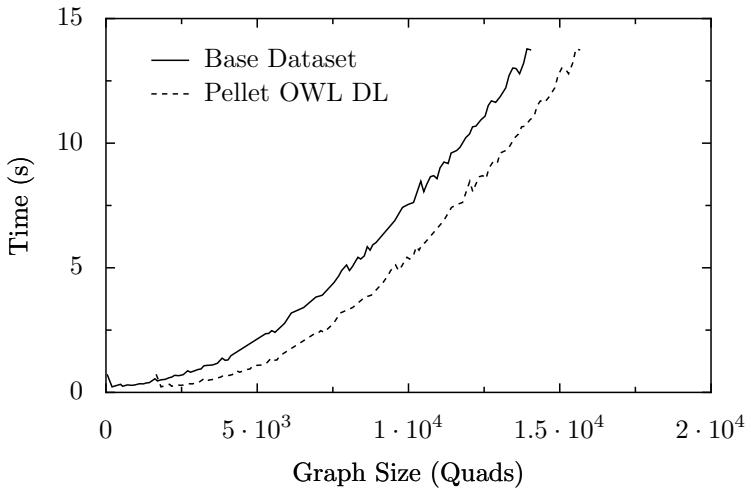

FIGURE 5.22: Pellet OWL DL Performance

OWL-Lite [Smith et al. (2004)], OWL-Mini, and OWL-Micro[11]. We have used Jena 2 because we are able to tailor its inference rules to suit our needs beyond OWL entailments.

DL is at best polynomial based on size of knowledge base. In most cases, even basic DL reasoning produces a vast amount of triples based on TBox classification. Whilst this may be useful in queries when you want to find the generic class of a resource, for a large dataset such information is not valuable.

The vast majority of inference algorithms including RETE and other JSR-94 rule engines consume enormous amounts of memory. Java™runs with a fixed heap range that means system resources can be still be used up if the knowledge base is moderately large (50,000 triples).

To test the relative performance DL based on the OWL DL sub-language, we evaluated two common implementations available to the public: Jena 2 and Pellet [Parsia and Sirin (2004)]. Jena 2 supports a subset of OWL DL based on a hybrid reasoner, whilst Pellet is a full OWL DL reasoner based on tableau algorithms [Baader and Sattler (2001); Horrocks and Sattler (2003)]. In both cases, we again used the DP instances generated in Section 5.1.1.2 and included the following OWL DL ontologies (see Section 5.3.2.1): DP, DOAP, FOAF, Simple Java, DCMI, DC Terms, DC Type.

[11] http://jena.sourceforge.net/inference/.

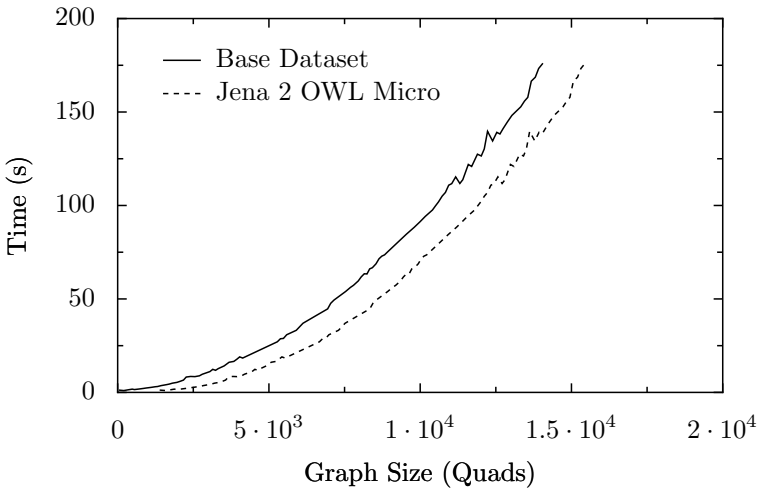

FIGURE 5.23: Jena 2 OWL Micro Performance

Figure 5.22 shows results for Pellet. It is fairly clear from the graph that the tableau algorithm used in Pellet is generally non-linear in shape given the different ontologies that we have included. Results published by the Pellet developers[12] show reasonable performance, there are no results on anything other than example ontologies; it seems OWL DL inferences on real data is somewhat rare. Fortunately, Pellet does not require a great deal of time to process the DP instances. As we shall see with the Jena 2 OWL DL, OWL Mini and OWL Micro, time can be a key issue.

The aim behind the Jena 2 OWL family inference engines is to provide a useful sub-set of OWL DL functionality[13] for Semantic Web applications. While some performance results have been provided, there is little information on how well the Jena 2 hybrid (forward and backward) inference engine works.

The Jena OWL (DL)[14] reasoner uses a Logic Programming engine to perform its inferences. In fact, the vast majority of rule languages, whether forward-based (e.g. RETE) or backward-based (Prolog, LP) use Horn clauses to describe rules.

OWL Micro is theoretically the least expressive of the Jena 2 OWL dialects, supporting little more than RDFS entailments as well as various property axioms. As can be seen in Figure 5.23 OWL Micro is non-linear, the main difference being the time scale; the performance hit incurred by using a RETE/LP hybrid engine is around a factor of ten. This is quite shocking given that the OWL Micro reasoner supports a minuscule sub-set of OWL DL in comparison to Pellet.

[12] http://www.mindswap.org/2003/pellet/performance.shtml.
[13] http://jena.sourceforge.net/inference/
[14] Jena 2 does not support the complete set of OWL DL entailments. See http://jena.sourceforge.net/inference/index.html for further details.

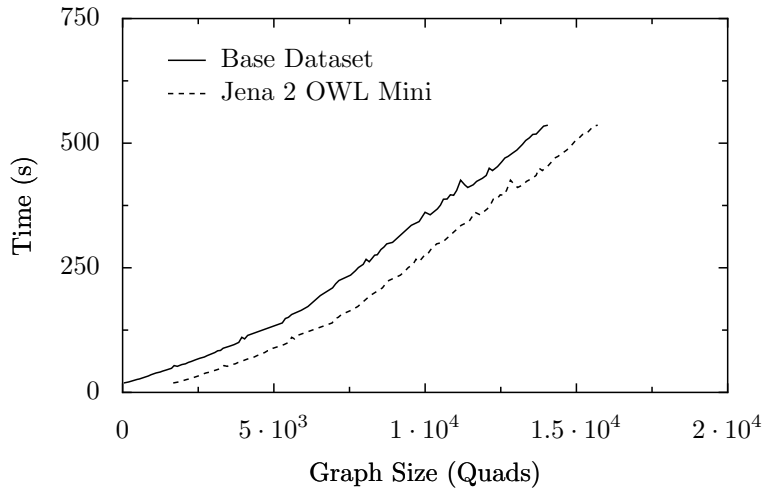

FIGURE 5.24: Jena 2 OWL Mini Performance

OWL Mini, another non-standard OWL DL sub-set fairs little better. While Figure 5.24 overall suggests linear relationship, this is unlikely given the results of Pellet and OWL Micro, and as we shall see, Jena 2 OWL DL. One difficultly that was found when taking results was the time taken for each iteration. As we can see, the time scale of OWL Mini has increased again, although only by a factor of five.

To get an idea of the memory usage of each implementation, we measured the memory (heap) used to before the entailments. Figure 5.26 shows a non-linear shape for OWL Micro, OWL Mini and Pellet. Unfortunately, the lengthy process of recording Jena 2 OWL DL performance leaves wildly sporadic memory usage results. What can be seen, however, is that Jena 2 OWL DL will eventually run out of heap (set to 1024MB) well before completing entailments on the DP instance data set.

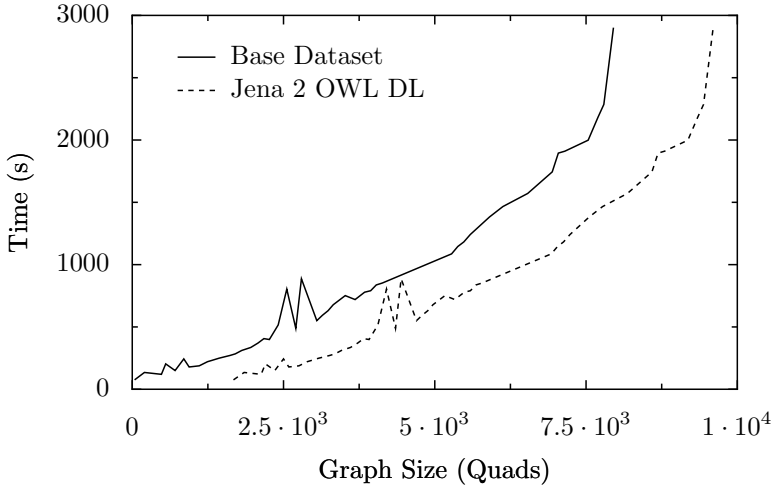

FIGURE 5.25: Jena 2 OWL DL Performance

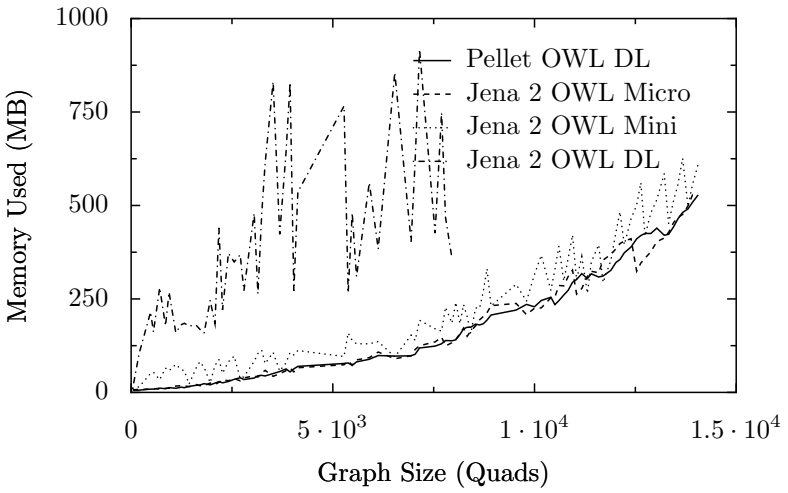

FIGURE 5.26: OWL Memory Performance

Depending on the OWL sub-language, entailments tend to generate a large number of triple instances of which only a few may be necessary to infer whether a class is a subclass within a class hierarchy. For example, a useful entailment would be to know that a **dp:Document** is also a **foaf:Document**. On the other hand, entailments that declare all DP instances to be **owl:Thing** is not as useful since *all* OWL instances will be an **owl:Thing**. If a large knowledge-base has an equally expressive OWL DL ontology, the vast majority of inferred triples will be useless and laborious to produce.

5.2.4 Rule Language Standardisation

In an ideal world, inference strategy should be irrelevant; forward or backward-chaining reasoners should produce the same results. If the Semantic Web is to progress beyond OWL DL classification, a common rule language needs to exist that can produce a consistent result across different logics. Unfortunately, this does not appear to be a priority based on the Reasoning on the Web Workshop at WWW 2006 in Edinburgh[15].

Barkmeyer (2006) argues that a common rule language is premature and will require considerable compromises. Even the existence of the Rule Interchange Format (RIF) at the W3C[16] has made little progress. Others are more positive, continuing to state, "a little semantics goes a long way". Even recent OWL Mini,

[15]http://www.aifb.uni-karlsruhe.de/WBS/phi/RoW06/.
[16]http://www.w3.org/2005/rules/.

however, has its performance problems as we discovered in Section 5.2.3[Hendler (2006)].

5.3 Document Provenance Evaluation

Here we evaluate the cost of developing an ontology for semantic version control using Semantic Web languages. We note the difficulties that exist in developing ontologies in general, as well as the importance of understanding the consequences of the mixing of ontologies with different levels of expressivity. We also comment on the challenges when using Named Graphs in conjunction with our DP ontology.

5.3.1 Ontology Design

Ontology design is quite possibly one of the most misunderstood activities in the Semantic Web. Design guides such as Noy and McGuinness (2001) give a general overview of what is meant by an ontology and state that there is no definitive methodology. Grüninger and Fox (1995) attempt to define and evaluate ontologies based on formal logic, although this is unlikely to be accessible to the average user, even if they do understand the principles of RDF[17]. Researchers have been working with ontologies for at least a decade, and many companies have invested a lot of money in an attempt to adopt ontologies in the real world.

Unfortunately, ontologies are expensive to create (low return on investment) and do not always reflect the way people think. If we consider OWL, we have a tree-like structure that, given the Open World Assumption (see Section 2.4.4), does not intuitively represent the real world and how we relate to it. Shipman and Marshall (1999) note some problems associated with ontology design from a user perspective. Users tend to be unwilling or unable to express concepts *explicitly* for system designers. Jones and Paton (1998) give some views on the technical perspectives of ontology design. They highlight five types of problem that can be encountered during the design process and offer some solutions based on extensive domain analysis. There also exist some theoretical limits to what an ontology can describe based on the computational complexity of expressive logics.

Ontologies are generally useful in describing a particular domain of knowledge. Potential problems come from who is generating terms for the ontology. In most

[17] http://www.dzr-web.com/people/darren/blog/2006/07/12/is-the-w3c-failing-us/.

cases a domain specialist is necessary to populate the terminology based on professional knowledge and other experience. On the other hand, Sabou et al. (2005) is a very good example of where software strategies can be used to improve domain ontologies. Apart from being self-descriptive, ontologies are of course useful as the basis of instances in the sense of DL. Based on our experience with OWL DL performance (Section 5.2.3), it is not clear how useful DL reasoning is in the long term. Certainly some value can be gained, unfortunately until DL reasoners become fast and efficient, they are unlikely to find themselves used beyond research.

Like a modern language dictionary, an ontology needs to be maintained (depending on use). This is especially true in open domains such as the Internet where users and developers find new ways to use a vocabulary or find use cases to extend it that previously did not exist. FOAF is an example which has slowly evolved based on a consensus in the Semantic Web community. WordNet is another example that follows more closely to our initial dictionary example; as the underlying *corpus* develops, so must the ontology that annotates it.

5.3.2 Expressiveness and Complexity

Knowledge-based systems suffer from a problem of expressiveness versus complexity; the more complex the constraints that exist between concepts and roles in a knowledge-base, the more computationally complex class subsumption and consistency checks become. The designers of OWL partially solved this problem by creating three levels of increasing complexity. OWL DL, which is equivalent to $\mathcal{SHION}$(D), is the most expressive these ontology languages which is guaranteed to be decidable.

From the outset of the design of the Document Provenance ontology it was clear that a considerable amount of instance data would be generated over time. Any subsumption or general purpose inference would be performed over this large dataset. If any system based on our design to be sufficiently usable, this would require us only being able to guarantee subsumption and general inference satisfiability, but also reduce computational complexity. Only two types of DL satisfy these requirements: $\mathcal{SHION}$(D) and $\mathcal{SHIOQ}$(D) which are approximated by OWL DL and OWL Lite respectively.

We found a minimalist OWL DL ontology would serve our requirements. Debugging based on work by Parsia et al. (2005) was essential and frequent but

highlighted the need for effective ontology evolution management. Each time we discovered we were defining concepts and roles that already existed, these ontologies were imported instead. Examples of this included ontologies for describing projects, people, and Java™ classes. Writing a new ontology that described all these concepts would not only have been wasteful, but also would reduce the possibility of semantic interoperability. However, due to semantic constraints imposed by OWL, it was vital we found versions of these ontologies that were no more expressive than OWL DL.

5.3.2.1 Ontology Interaction

During the design of our DP ontology we were careful to include common ontologies available that support various application domains such as people (FOAF) and projects (DOAP). The idea behind *importing* an ontology is to extend and be able to maintain semantic interoperability through the use of OWL entailments. As we demonstrated with our federation scenarios, multiple ontologies are also useful for data federation even though OWL (DL, Mini, Micro) entailments are remarkably slow.

Unfortunately, simply importing an ontology is not enough. Depending on the language used, an ontology has a particular logic "strength" that affects the overall expressivity of the ontology importing it. Understanding this important yet subtle issue is one of the great problems with developing ontologies and the Semantic Web in general. Core references on RDF [Calvanese and Giacomo (2003); Klyne and Carroll (2004); Hayes (2004)] should become required reading for any appreciation of the foundations of the Semantic Web.

Rector et al. (2004) provides an enlightening overview of the common errors that can be introduced when working with OWL DL. Although they do not deal with the issues surrounding mixing logic strengths, they do highlight the issues of understanding Open World Reasoning, which is quite rightly a difficult hurdle to pass. If an ontology developer is to successfully write a sound ontology they must understand the consequences of Open World Reasoning.

On the other hand, if an ontology developer wants to effectively reason over their new ontology and associated instances, they must choose the ontology language carefully. Table 5.1 gives an overview of each imported ontology, its language and complexity level. The Dublin Core ontologies are currently RDFS[18]; FOAF is

[18]OWL DL version available at http://protege.stanford.edu/plugins/owl/dc/.

OWL Full[19]; DOAP is also OWL Full since it imports FOAF[20]; the Simile project's Simple Java Ontology is RDFS in Notation 3 (N3) syntax.

Ontology	**Language**	**Complexity Level**[21]
Dublin Core Elements 1.1	RDFS	OWL Full
Dublin Core Terms	RDFS	OWL Full
Dublin Core DCMITypes	RDFS	OWL Full
Friend of a Friend (FOAF	RDFS/OWL	OWL Full
Description of a Project (DOAP)	OWL	OWL Full
Simple Java Ontology	RDFS	OWL Full

TABLE 5.1: Imported Ontology Complexity without Modification

One of the reasons FOAF is classified as OWL Full is the inclusion of a property that is defined as `owl:inverseFunctionalProperty` with a range `xsd:string`. Inverse functional roles are useful in that they can *uniquely* identify the subject of a triples (see Section 2.4.1). In the case of `foaf:mbox_sha1sum`, however, the use of `xsd:sting` causes the ontology to become OWL Full.

As we noted in Section 4.1.3, while it is permissible to freely mix ontology languages together, OWL rules state that importing a non-OWL ontology *automatically* requires the new ontology to be OWL Full. FOAF is a prime example as it uses a combination of RDFS and OWL constructs. All three DCMI ontologies and the Simple Java Ontology are similarly OWL Full since they are pure RDFS. DOAP is itself OWL DL; however, since it imports the default FOAF OWL Full ontology, it must also be OWL Full.

Ontology	**Language**	**Complexity Level**[22]
Dublin Core Elements 1.1	OWL	OWL DL
Dublin Core Terms	OWL	OWL DL
Dublin Core DCMITypes	OWL	OWL DL
Friend of a Friend (FOAF)	OWL	OWL DL
Description of a Project (DOAP)	OWL	OWL DL
Simple Java Ontology	OWL	OWL DL

TABLE 5.2: Imported Ontology Complexity after Modification

By using the OWL DL version of these ontologies, we produced an ontology that was OWL DL, with an expressivity equivalent to $\mathcal{SHIOF}$ (see Table 5.2). This ontology provides a minimal set of concepts and roles we have found necessary to describe the principles of version control.

[19]OWL DL version available at http://www.mindswap.org/2003/owl/foaf/.

[20]Version available at http://www.usefulinc.com/doap/ becomes OWL DL once OWL DL FOAF is imported.

5.3.3 Temporal Restrictions

While ontologies can be seen as highly extensible schemata, concrete languages including OWL are intrinsically static in nature. This limitation stems from the monotonic foundations of formal logics such as FOL (Section 2.4.1). For example, it is difficult to represent processes or anything that represent a sequence of events. While proposals to extend DLs with temporal semantics exist [Artale and Franconi (2000)], there are no mainstream temporal DL languages that we are aware of.

Our Document Provenance ontology does to a certain extent represent a sequence of events, although we have been careful never to negate knowledge; our knowledge-base always grows as new commits are added. In fact, our ontology and its underlying logic are well suited to version control since all information is stored indefinitely. This means we can take full advantage of our logic of choice yet minimise most of the inadequacies of the approach.

The end result has been an ontology that is compact and easy to manage. We have found ontology design to be subjective and despite efforts by authors [Noy and McGuinness (2001)], tends either to be application specific, or too general to be useful without creating subclasses or sub-properties. FOAF and Dublin Core are common examples. FOAF is highly unstable (subject to regular change) and targeted towards social relationships; Dublin Core is highly stable (controlled public releases) and targeted towards resource cataloguing with a set of all-purpose, standardised elements. When suitable properties cannot be found, Dublin Core Elements can be used as base properties that provide some semantic compatibility.

5.3.4 Provenance Mechanism

Throughout the duration of this thesis there has been no clear consensus in the RDF community on how provenance should be tackled. It was clear that while we had developed an ontology that provided descriptive annotation that could be readily rederived from the source document, there was a need for a mechanism that could reliably bind provenance to source knowledge. The emergence of Named Graphs as a contender for this task meant we had a way to not only bind provenance to source knowledge represented in DP, but also cause the author of that provenance to be accountable through the non-repudiation of digital signatures.

Prior to being endorsed in SPARQL draft specification, Named Graphs were a niche research topic, while quads and contexts had too many toolkit and library

constraints, and RDF Reification was not living up to expectations. The potential of Named Graphs for provenance annotation gave us the impetus to use it and it also gave us some influence on the future development of the NG4J API; our work with Named Graphs has meant that it is now better known and now supports basic digital signatures in RDF.

We have found that working with Named Graphs has been both challenging and rewarding. Management of Named Graphs is still in its infancy but possible with the SPARQL query language. Use of Named Graphs with inferences is also a challenge since rule languages such as the one used in Jena 2 cannot match against the graph name. With the knowledge we have gained, however, we are well placed as SPARQL and new inference engines become mainstream.

5.4 Research Evaluation

The main thrust of this thesis has been to determine the viability of using Semantic Web technology as an alternative approach to the RDBMS that improves version control in distributed collaborative software development. To this end, we investigated the merits of using Semantic Web technology as described in Chapter 3, namely knowledge federation, explicit trust of servers and new facilities. We also investigated the approaches that bind provenance to source knowledge that we could subsequently reason over.

In the case of provenance binding, our research has led us to develop an ontology that introduces a novel approach to version control yet captures the principles behind early provenance recording strategies. This has been integrated with a cryptographic integrity mechanism based upon Named Graphs and a PKI, a well established approach to trust in business-to-business environments.

Based on the federation scenarios in Section 4.7 we argue that our case studies as described in Chapter 3 can be *better* supported by our semantic version control approach. Despite immature performance, we see the key enablers of our approach as trusted metadata, federated collaboration, and to some extent semantic inferencing.

5.4.1 Trusted Metadata

We argued in Chapter 1 that software development hosting platforms such as SourceForge force developers to implicitly trust the server without a way to verify the integrity of the server's content. Few version control systems in a hosted environment support anything that includes secure hashes on version controlled resources or even digital signatures. Our solution to this has been to develop an approach to version control where integrity is vested in the metadata itself. Rather than simply starting the commit process, software developers become part of the process with the result that the signed metadata becomes part of a trusted audit trail. This also means that they become more accountable for their actions; administrators can query the repository and other repositories in the event a signature fails.

Confidence in the metadata is only one part of the problem. Organisations that collaborate together as they do in our case studies need a mechanism to dynamically create trusted collaborations with minimal fuss. Our approach to this has been to use well-established grid middleware in the form of GRIA to transport metadata from trusted sources into local repositories.

Performance results for our choice of *conservative canonicalisation* is promising, even though large scale signing of graphs is expensive. We have shown though that this is not that much of a boundary; DP instances are generally small in size, in the order of hundreds rather than thousands of triples per graph. It is more likely that problems will occur in preparing graphs for signature verification: querying of large datasets to extract original graph and associated Warrant graph.

5.4.2 Federated Collaboration

Knowledge federation has often been touted as a genuine advantage of Semantic Web technology [Schraefel et al. (2003); Jaén et al. (2005); Park (2006)]. The ability to merge disparate data sources using the RDF data model is appealing, especially when additional information is required. Analysis of RDBMS technology has shown that knowledge federation is not readily possible and that the static nature of the RDBMS schema makes it difficult to federate new data structures in a dynamic manner. Our approach to addressing the data federation issues in our case studies was to investigate what kind of scenarios we could use to demonstrate the use of Semantic Web federation. The result of this was a set of scenarios

in Section 3.5 that we implemented in Section 4.7 and performance tested in Section 5.1.2.1.

Rather than using the term knowledge federation, it might be more appropriate to use *federated collaboration*. Throughout this research we have been investigating the use of Semantic Web technology in distributed collaboration. Our scenarios demonstrate the federation of data between collaborating parties, both FLOSS developers and IST project partners. Since much of this collaboration is between different software repositories, we have already gone beyond what is possible with current version control systems.

5.4.3 Semantic Inferencing

We have shown that inferences over semantic repositories is inevitably a slow process. If we consider the results collected in Section 5.2.3, we found any OWL DL, Mini or Micro subsumption to be overly expensive and largely un-useful. The possibility of using such technology in a production environment with slow and unscalable performance is highly unlikely. Performance was compounded by our DP ontology importing several other ontologies. Since rule engines like the Jena 2 RETE engine match triple patterns in main memory, large knowledge-bases will quickly use up all available heap space. Until more scalable algorithms are used or a new alternative is proposed for Semantic Web inferencing, performance will always be an issue. The inference rules we used in our signature recovery scenarios were made relatively simple so that they would complete in a reasonably short period of time.

The declarative programming approach we used for semantic inferencing has its advantages and disadvantages. On the one hand, they provide a certain amount of flexibility when introducing new functionality. We could, for example, quite easily extend our federation scenarios with more elaborate rules. Unfortunately, management and maintenance of rules is problematic because it is difficult to get an overall picture of the data flow between different rules. We have already seen in Appendix D that to represent a logical *OR*, we must write a separate rule that tests for each different condition. We anticipate that future work will investigate new strategies to improve performance, management, and maintenance of rules.

5.5 Summary

Results from our research largely support the notion that Semantic Web technology *can* is a viable alternative to the RDBMS in distributed collaborative software development. Our federation and signature recovery scenarios show that knowledge federation is possible across different trust domains; our trusted metadata approach demonstrates that it is no longer necessary to implicitly trust the host server; our choice of OWL DL means that other developers can improve on our approach in an open manner to promote federated collaboration.

While reliably binding provenance to source knowledge has been a challenging task, we believe that our approach is an appropriate solution that is scalable at the provenance level. The use of semantic inferences have been somewhat successful, although future iterations might include more efficient approaches to declarative programming.

All this effort, however, comes at a price: performance and scalability. Our choice of RDF as the data model has been both a blessing and a curse; its semi-structured nature is highly flexible and extensible at the expense of efficiency compared to an RDBMS schema. The absence of an efficient indexing strategy in Semantic Web toolkits such as Jena 2 or NG4J means that both performance and scalability is a long term issue for the sustained use of Semantic Web technology. We have already seen in our results that accessing our trusted metadata is expensive and becomes even more expensive as remote access is introduced.

In the last chapter we evaluate our contributions, review related work, propose future work and conclude.

Chapter 6

Summary

The advent of Service Orientated Architectures (SOA) [Erl (2005)] and web services has brought about a fundamental change in the development of distributed systems. Rather than rely on proprietary and incompatible network protocols, most systems now build upon HTTP. Recent initiatives such as the Semantic Web build upon the WWW and even SOA, where we see the emergence of the semantic web services and the Semantic Grid. Our work intersects many if not all of these domains where we have investigated the viability of modelling version control using Description Logics rather than older RDBMS technologies and new ways to reliably bind provenance to source knowledge.

To validate our approach, we analysed the need for additional facilities supported by the Semantic Web using two case studies: FLOSS and EC IST project development. We argued that each case study could be better supported using a semantic version control system, based on the advantages of knowledge federation and trusted metadata. To our knowledge, the level of integration we have achieved with Semantic Web, digital signature and grid technology is both unique and novel.

Early on in our research, it became very clear to us the value and potential of Named Graphs in the role of provenance. We recognised that for semantic version control, we required two types of provenance: descriptive (annotation) and assertive (relationships). We found initial investigations with RDF Reification, RDF molecules and MSGs to be inadequate, none of which were able to effectively make assertions about RDF graphs. In Named Graphs, however, we found an approach that could not only make assertions about other graphs, that is, reliably bind provenance to source knowledge at the graph level, but also attach more

complicated data structures like digital signatures. Based on work by Tummarello et al. (2005) and Carroll et al. (2005), we were able to write an RDF digital signature mechanism that generated signatures that were distinct from the source graph and whose logic is capable of being reasoned over [Watkins and Nicole (2006)].

Our work on RDF digital signatures has been reasonably successful. Our *conservative canonicalisation* approach for signing DP instances has meant that the integrity of metadata is vested in the metadata itself rather than the environment where it is stored. This means we have been able to remove the reliance on trusted servers in an otherwise untrustworthy environment. We have also been able to leverage this integrity to support non-repudiation and IPR attribution, which we argue has been essential for our case studies. As part of our collaboration with international partners in Bizer et al. (2005a), our RDF digital signature solution has been published in NG4J as part of the SWP toolkit. We will later look at ways our approach can be improved.

Fortunately, our decision to use Named Graphs has been validated by an unlikely source: the SPARQL query language. This has brought Named Graphs much closer to mainstream usage, where Semantic Web developers are now being exposed to the possibility of querying more than one RDF graphs. SPARQL has, of course, made our research easier by providing a query language that supplies many of the necessary features for our federation scenarios. SPARQL is now a W3C Last Call Working Draft and should become a W3C recommendation in the near future.

Our work with the Semantic Web, however, has been more than just recording provenance and generating digital signatures. We have also been interested in the additional facilities that it would provide, namely knowledge federation and semantic reasoning. While knowledge federation has proved to be valuable in our federation scenarios in Section 4.7, our experience of inferencing, in particular OWL DL, has been less positive.

Further analysis of OWL DL and its performance has shown that it is not as useful as it first appears. Firstly, quantitative evaluation found it to be computationally expensive, even for small scale datasets. Secondly, the value of the entailments provided by even the weakest non-official OWL sub-language, OWL Micro, is called into question based on the cost to produce the entailment in the first place. Even general purpose reasoning is expensive for simple tasks such as knowledge federation orchestration. Management and maintenance is also an issue since there is no clear way to track the data flow between rules since rule execution in a RETE-based engine is not sequential.

Unfortunately, despite the emergence of a common query language for RDF, different logics are starting to be used to realise Semantic Web. OWL, which represents the prevalent approach, builds upon early Description Logic research that is known to be decidable. Other approaches, such as F-Logic are starting to gain ground in WSMO [Dumitru Roman and Fensel (2005); Fensel et al. (2006)], even though they are incompatible with OWL. Horrocks et al. (2005) warns of the dangers in a Two Tower approach to the Semantic Web. Our own experience from WWW 2006 has also shown that the DL-Datalog semantic rift also applies to inferencing. This means that there is a real risk that rule authors will not be able to write rules independent of their choice of logic.

While we have achieved a great deal during this research, many issues still remain. In the next section we evaluate our approach and consider what aspects of our research has not been fully realised.

6.1 Self Evaluation

The effective use of Semantic Web technology is deceptively difficult, requiring the appreciation of several different topics areas. During our research in the use of the Semantic Web, we discovered the importance in understanding Description Logics, the Open World Assumption, monotonic logics, and what is meant by expressivity. Many Semantic Web advocates who have espoused the virtues of RDF and OWL, and how they are essential to the next iteration of the WWW fail to highlight the importance of these concepts. This is a pity since not understanding these core concepts can lead to incorrect decisions when developing ontologies and developing inference-based systems. We see our research as not only demonstrating the value of the Semantic Web technology in distributed collaborative software development, but also highlighting the extent to which the technology can be used in practical applications.

We believe that our choice to use Semantic Web technology, rather than more conventional approaches has been fruitful despite the hurdles we have faced. Most of the additional features we expected to be made available from our design have been validated in our federation scenarios: the use of trusted metadata and knowledge federation. Knowledge federation is key example where Semantic Web technology is becoming better understood and showing real results; this is especially true when providing access to legacy RDBMSs [Wilson and Dardailler (2003); Bizer and Seaborne (2004); Hawke (2002); Jaén et al. (2005)]. Our own work continues

to validate this approach, which is a positive sign that the RDF data model has an advantage over RDBMS schemata. Semantic Web inferencing, however, has not produced the results we expected and has shown OWL DL to be not as useful as first thought. We will discuss the performance issues later in this section.

Trust is a another hard issue that plagues the Semantic Web. People are starting to realise that the quality of information on the WWW is not quite what they thought is was; Wikipedia has fast become a battle ground over information quality [Carr (2006)]. Others argue that the Semantic Web is no better. Doctorow (2001) provides a strong critique on "metacrap", arguing that the same problem can only get worse; even Tim Berners-Lee understands the problem, although he does not provide many answers [Rowland (2007)].

On the other hand, recent research by Wilkinson and Huberman (2007) shows Wikipedia to be a successful *collaborative* effort where quality generally improves with the number of edits. This suggests that the "soft" peer review approach at Wikipedia does work to some extent[1]. Unfortunately, until Wikipedia provides a mechanism for accountability of authors, it will remain a social collaborative effort with no real trust. Our use of digital signatures binds the author to the commit process, and makes them accountable through the PKI.

While our research has produced some positive results, there are several areas where we think improvement can be made on our research. These include the canonicalisation algorithm used in our RDF signature approach, and Semantic Web toolkit performance.

6.1.1 RDF Canonicalisation

One of the key challenges in this research has been the development of an RDF digital signature mechanism that was fast, efficient, and scalable. Our initial approach was to sign serialised RDF/XML using XML Digital Signature. Since this approach did not enable us to store the digital signature in a triple store we investigated other approaches. Work by Carroll et al. (2005) provided a new direction that would mean including the digital signatures within an RDF data model.

The RDF digital signature mechanism used in our research relies on the constrained usage of an algorithm described in Carroll (2003), which we have called

[1] http://en.wikipedia.org/wiki/Wikipedia:General_disclaimer.

conservative canonicalisation. In this approach, we restrict the RDF graphs that we sign to only fully labelled graphs; we therefore forbid blank nodes. Consequently, this has meant that all metadata used in our online collaborative tool and federation scenarios have been based on fully labelled RDF graphs.

While our *conservative canonicalisation* approach has performed well in our research, it is not practical to attempt to limit all RDF graphs to being fully labelled. This means if RDF digital signature is to be used in different domains, a better canonicalisation method must be found. As we have demonstrated in Appendix C, since Carroll's algorithm and **nauty** use wildly different techniques, they cannot be used together. **nauty** on its own is an option, although it would have to be updated to understand labelled edges and Named Graphs. We have limited our activity in this area, because the topic of graph isomorphism is beyond the scope and competency of this thesis.

6.1.2 Trust

The only trust model we have used in our research has been the classic PKI model that relies on a trusted third-party, the Certificate Authority. While this model is well tried and tested, it does not necessary enable us to leverage the full potential of RDF digital signatures on the Semantic Web.

If we consider the inference rules we defined in Appendix D we can envisage the use of more complex trust models like those used in Golbeck and Hendler (2004a) or Bizer and Oldakowski (2004). A combination of different trust models would be advantageous in our federation scenarios where consumers could use PKI, semantic trust metrics and SPARQL-based queries to limit what kind of information they are will to accept as genuine.

6.1.3 Performance

We have already seen that less than optimal performance from Semantic Web technology in our research. However, we can argue Semantic Web toolkits including Jena 2 and NG4J rightly choose novel features over low performance. The vast majority of the technology we use today came from humble research projects; the WWW is a prime example. It is often typical that initial releases of a new technology are slower than the previous generation, although new features and

advancements tend to overcome the perceived disadvantages. As the technology matures, features may be dropped (semantic inferencing?), but features with a large market will survive and flourish.

Results taken in Section 5.1.1 suggest that the RDF data model is the likely cause for the lack of performance in RDF toolkits. The semi-structured, flexible manner of RDF makes it difficult to optimise in an RDBMS, making it extremely difficult to index. A first step to improving this situation is to investigate efficient ways to index RDF, preferably with an RDBMS or using an object-orientated approach. The next step would be to involve industry; this is essential for industry-wide take up of the Semantic Web. Companies will not adopt technologies that they cannot have a vested interest in. Perhaps following Oracle's lead might help; the recently released Oracle 10g supports the RDF data model.

A similar approach should also be applied to semantic inferencing. Current inference strategies are over complicated, slow and could be complemented current adoption of process-based workflow technology like BPEL 2.0 [Jordan and Evdemon (2007)] and Windows Workflow Foundation [Scribner (2007)]. New, more practical algorithms should be investigated that can be integrated with common business processes. The development of a common rule language could also be developed in conjunction, helping to break the logic-split and make RIF a reality.

6.1.4 Achievements

Achievements of the work presented in this thesis can be summarised as follows:

1. The design of a DL framework based around Named Graphs called "Document Provenance", used as the basis for our online collaborative tool and federation scenarios.

2. Development of an RDF digital signature mechanism based on our work with Named Graphs. This work has subsequently been integrated into a toolkit in collaboration with international partners which is now part of the Semantic Web Publishing Framework, a sub-project of the NG4J. Our mechanism is one of the few examples that satisfies the Digital Signature portion of the Semantic Web stack (see Figure 3.1).

3. Development of an online collaborative tool that enables distributed collaborative software development and supports two different case studies: FLOSS

and EC IST projects. We used this tool to enhance the state-of-the-art, demonstrated with a set of federation and signature recovery scenarios.

4. Used our online collaborative tool to mine information from existing software repositories, such as the Taverna workflow platform.

5. Evaluated the use of Semantic Web technology through quantitative and qualitative analysis. Our evaluation included performance comparisons against RDBMS technology, as well as performance analysis of our RDF digital signature mechanism and various OWL DL inference engines. We further analysed the performance of our knowledge federation and digital signature recovery scenarios.

These achievements serve to reinforce our original research statement made in Section 1.3, chiefly that this thesis investigates new and novel strategies to improve version control in distributed software development. Our approach not only provides a trusted metadata approach that can reliably bind provenance to source knowledge, but also federation facilities not available in current version control systems.

6.2 Related Work

In this section we present research that is related to the key issues we have been investigating in this thesis namely, provenance mechanisms, trusted metadata, RDF digital signatures, and semantic knowledge federation.

6.2.1 Provenance Frameworks

Recent work by Miles et al. (2007) presents a platform-independent framework for validating workflow executions using a mixture of OWL descriptions and XML-based provenance. OWL and web service descriptions are used during the validation process based on a set of inference rules implemented in Jena 2.

Several other approaches exist for employing semantics and provenance in service-orientated environments. Chen et al. (2006b,c) describe a hybrid provenance approach, defining a new kind of provenance, *augmented provenance*. They distinguish augmented provenance as an enhancement of existing provenance using

extensive metadata and semantic. Chen et al. (2006a) envisage the use of service-based semantic for use in service discovery and composition, previously an aim of OWL-S.

Liang (2006) seeks to address the issues surrounding ontology change management using a *Log Ontology* to capture the ontology change information. They see two approaches to ontology versioning: passive and active analysis. Passive analysis compares the current version to previous versions, whereas active analysis records all change events as they occur in a similar fashion to a logging utility.

6.2.2 RDF Digital Signatures

Tummarello et al. (2005) and Cloran and Irwin (2005) offer the only real alternatives to signing RDF. The approach taken by Tummarello et al. (2005) is to use RDF Reification to attach the signature to the graph. While this approach appears sound, we have argued that it is semantically flawed (Section 3.3.2.2). Cloran and Irwin (2005) take a more conventional route, simply signing serialised RDF for later verification. The use of XML Digital Signature is a good approach, since it means developers who have existing toolkits based on XML Digital Signature can integrate easily. The only disadvantage is that managing signed RDF becomes a problem; they have yet to address this in their approach.

Following on from their PASOA work, Tan et al. (2006b) have investigated the use of XML Digital Signature to enable accountability and non-repudiation for p-assertions [Groth et al. (2004)]. P-assertions are placed in the WS-Addressing header of a SOAP message, then signed using XML Digital Signature. While this approach has the potential to support non-repudiation, this is only applicable if an asymmetric algorithm is used; symmetric algorithms such as Kerberos and SAML token profiles do not support non-repudiation in WS-Security, which relies on XML Digital Signature.

Tan et al. (2006a) look at a more wide range of security issues of when using p-assertions. Rather than being limited to non-repudiation and digital signatures, they investigate access control, trust models, confidentiality, and archival of p-assertions. It is important to note here that Tan et al. (2006a), like Braun and Shinnar (2006) have an interest to potentially protect access to provenance, an issue that we have not explicitly addressed.

6.2.3 Semantic Knowledge Federation

Bizer and Cyganiak (2006) provide a good example of where Semantic Web technology is being used to integrate legacy databases using SPARQL queries. They define a declarative mapping language based on earlier work in Bizer and Seaborne (2004). This can be compared to Joseki[2], an RDF server that only supports a native RDF dataset. While our work does not envisage the use of a legacy database, it would be interesting to investigate how our semantic version control approach can be mapped onto existing version control systems.

Newer toolkits now support distributed SPARQL-based federation. DARQ[3] is one such example. It can be used to query multiple graphs which could include Joseki and D2R servers, as if they were a single RDF graph. The ability to perform distributed querying is a useful feature, even though DARQ is unable to perform DESCRIBE or GRAPH constructs which would be required in our federation scenarios. We expect that future versions of DARQ will support these features.

6.3 Future Work

Future work outlines areas where we can perform further research based on the issues discussed in our self-evaluation and related work. Here we identify several potential and interesting areas for future work which include: architecture, ontology, logic, and federation extensions. We also briefly discuss how performance might be increased.

6.3.1 Architectural Improvements

6.3.1.1 GRIA

The use of GRIA in our federation scenarios has shown how easy it is to develop a SPARQL service for querying our version control metadata. While our wiki interface is useful for viewing documents and their histories online, third-party access is an obvious advantage. Expanding our use of GRIA will enable us to create more complex trust relationships between developers and service providers

[2] http://www.joseki.org/.
[3] http://darq.sourceforge.net/.

who store the trusted metadata. One example would be to include constraints on accessing our SPARQL service using Service Level Agreements (SLAs).

6.3.1.2 Maven 2

Another extension might include the integration of the Maven 2 project management system [Massol et al. (2006)]. Projects hosted on our online collaborative tool could use Maven 2[4] for full building, testing and deployment services. This would potentially make the online collaborative tool a *trusted compilation* platform.

6.3.2 RDF Digital Signature Improvements

There are two key areas where we can improve on our RDF digital signature mechanism: the canonicalisation algorithm and the use of trust metrics. Improvements in both of these areas would increase the reliability of our online collaborative tool and create a solution that could be reused elsewhere.

As we noted in Appendix C, **nauty** is perfectly capable of canonicalising complex unlabelled graphs like the Petersen graph. One option would be to extend **nauty** so that it can understand labelled edges and Named Graphs. This would mean our RDF digital signature mechanism would be able to sign over arbitrary RDF graphs, removing the need for our current *conservative canonicalisation* approach.

It appears that recent work by Tan et al. (2006b) is particularly relevant to our research. It would be advantageous to collaborate in future work to establish how PASOA-based provenance and our Named Graph approach can be integrated. The interest in trust metrics in Tan et al. (2006a) is also timely, since we believe that by expanding our RDF digital signature mechanism to support trust metrics, we can leverage existing approaches that have been developed in parallel to our own work. It is highly likely that integrating these approaches with research by Dimitrakos et al. (2001); Bizer (2004b) and Golbeck and Hendler (2004a,c) will open up new avenues of collaboration.

[4] http://maven.apache.org/.

6.3.3 Ontology Extensions

Just like the documents we put under version control in our online collaborative tool, ontologies develop over time as requirements change [Noy and Klein (2004)]. While it is not envisaged that the ontology used as the basis for version control be managed by the same system, there may come a point where extensions to the ontology are vital for future development. Other developers, for example, may want to improve the ontology, which will require change management.

6.3.3.1 Advanced Software Project Management

Software project management systems, for example Maven 2 [Massol et al. (2006)], have become a fast and efficient method to manage and automate the build process of simple and complex projects. If we were to consider Maven 2 as part of the core architecture it would be necessary to model the Maven 2 build life-cycle so to capture the progress of a build.

6.3.3.2 Intellectual Property Rights Management

While our RDF digital signature mechanism supports non-repudiation when using a PKI and can help enforce Intellectual Property Rights, we have not written an ontology to represent these rights. González (2005) suggests a interesting approach to developing an OWL ontology for Digital Rights Management. This ontology approach could be used as a first step to creating a generic ontology for IPR attribution.

6.3.4 Logic Extensions

6.3.4.1 Non-monotonic Reasoning

One key advantage we have noted during this research of DL over relational database systems is the ability to leverage *explicit* knowledge and generate *implicit* knowledge using inference rules. Inferences are not limited to just RDF, RDFS and OWL entailments; our work has demonstrated that useful information can be generated with custom inference rules. These inferences, however, operate

under monotonic, open world semantics in line with Semantic Web "best practices". As SWRL becomes the mainstream language for rule composition, some researchers are beginning to advocate non-monotonic extensions to OWL [Katz and Parsia (2005); Hitzler et al. (2005)]. Others suggest combining the use of open world reasoning with closed world reasoning at a local level [Grimm and Motik (2005); Kolovski et al. (2005); Ng (2005)].

Web service description languages such as the Web Service Modelling Ontology (WSMO)[5] define a set of non-monotonic extensions to an otherwise monotonic framework. Similar non-monotonic extensions could also be applied to our Named Graph work.

6.3.5 Federation Extensions

6.3.5.1 Process-based Workflow

We have noted some of the management and maintenance issues regarding Jena 2-based inference rules. Although the declarative approach used is flexible, it makes data flow difficult to track, and can be computationally expensive depending on the expressivity of the rules and procedural builtins used.

Another approach that could be used to complement declarative rule languages is process-based workflow. Process-based workflow, while sequential in its execution, can easily track data flow and has several industry standards (BPEL 2.0), currently lacking in the rule domain. Microsoft has gone some way toward this integration with Windows Workflow Foundation (WF), which is capable of firing rules sequentially [Young (2005)]. Workflows could be used for data flow and orchestration, firing rules as they are required. It is also conceivable that further work here could produce useful results that work across different platforms.

6.3.5.2 SPARQL Query Protocol

We have taken advantage of only a small subset of SPARQL's language. SPARQL also defines a query protocol which might be useful to employ, especially through a SOAP interface like that of Joseki[6]. Since SPARQL is now in its last call, all features should now be stable; this will encourage adoption.

[5] http://www.wsmo.org/.
[6] Available at http://joseki.sourceforge.net/.

It would be reasonable to take the SPARQL query service we developed for our federation scenarios and develop it into a complete SPARQL protocol service. A client should also be developed that leverages DARQ.

6.3.5.3 Natural Language Processing

Unfortunately, the interface between application and Semantic Web query mechanisms is such that it is difficult to dynamically create queries at runtime. This needs to be improved, especially when queries are used in conjunction with semantic inferences. Natural Language Processing (NLP) is one approach that maps basic English onto ontology concepts and roles.

6.3.6 Performance Enhancements

At present there are two issues that need to be addressed before Semantic Web toolkits will improve in their performance: triplestore database schemata and indexing. Unfortunately, efficient indexing is linked to the schema used to represent the RDF. It may be that rather than using SQL to perform the indexing, it will be necessary to let a higher level library perform this task.

To our knowledge none of the major Semantic Web toolkits share the same database schema for persistent storage. Each take their own approach, which can mean different performance depending on the toolkit used. NG4J, for example, takes a naïve approach to persistent storage, keeping all components of a quad (graph-name, subject, predicate, object) in the same table. It might be more productive to investigate the Oracle approach where different components of an RDF triple are kept in different tables.

6.4 Conclusions

In this study we investigated new and novel strategies to improve version control in distributed software development. We posed two questions that would form the core of this thesis: firstly, we considered the use of Semantic Web technology as an alternative to the traditional relational database used in Subversion. Secondly, we attempted to discover whether we could reliably bind provenance to source

knowledge contained within a semantic version control repository and use it to infer new knowledge.

We believe that our approach with Semantic Web technology has been relatively successful. Our experience with ontologies and knowledge management has helped us develop a DP ontology that made a minimal set of extensions based on popular ontologies. This DP ontology was used to enhance a basic Wiki with semantic content, capable of recording the version history of Java™ documents. Since our DP ontology used OWL DL, we were able in principle, to incorporate DL reasoning into online collaborative tool. Experiments with a set of federation scenarios revealed to us, however, that knowledge federation was of greater value than DL entailments, which showed poor performance. Experience in the use of Semantic Web technology has shown us how far we can push it (provenance binding, federation), and where it fails to deliver (performance).

Our work on binding provenance to source knowledge has been more mixed. We have been able to create reliable provenance using Named Graphs which we see as a natural way to record provenance. We have also been able to digitally sign DP instances to create *trusted metadata* that can maintain its own integrity. This is a useful result from our work, since the reliance on server integrity that we were trying to avoid has been reduced. On the other hand, to achieve a reliable RDF digital signature mechanism, we have had to constrain the type of RDF that we sign to fully labelled graphs only. We anticipate that future work will see this constrained use of RDF disappear.

As a demonstration of how our work supported our case studies we developed a set of federation and signature recovery scenarios that used a combination of SPARQL and inference rules. We found our inference rule approach to be interesting, but ultimately difficult to manage and maintain. We found management and maintenance of declarative rules required intimate knowledge of the application's data flow; rules can be written in any order independent of execution, which makes it difficult to track and debug rules. Representing different conditions requires separate rules since it is not possible to simulate logical ORs in the Jena 2 rule language. Another issue with using inference rules is that if someone else wants to perform same inferences with their own legacy tools, there is no common rule language to maintain consistent view in a distributed federated environment.

While this work has reached an end point, we realise that further work can be done on what has already been achieved. With this in mind we have listed several areas where our work can be extended.

Appendix A

A.1 Document Provenance Ontology

```
<?xml version="1.0"?>
<!DOCTYPE owl [
<!ENTITY owl "http://www.w3.org/2002/07/owl#">
<!ENTITY java "http://simile.mit.edu/2004/09/ontologies/java#">
<!ENTITY dp "http://grid.cx/dp/1.0/">
<!ENTITY foaf "http://xmlns.com/foaf/0.1/">
<!ENTITY terms "http://purl.org/dc/terms/">
<!ENTITY doap "http://usefulinc.com/ns/doap#">
<!ENTITY swp-2 "http://www.w3.org/2004/03/trix/swp-2/">
<!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#">
<!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">
<!ENTITY dc "http://purl.org/dc/elements/1.1/">
]>
<rdf:RDF
 xmlns:owl="http://www.w3.org/2002/07/owl#"
 xmlns:java="http://simile.mit.edu/2004/09/ontologies/java#"
 xmlns:dp="http://grid.cx/dp/1.0/"
 xmlns:foaf="http://xmlns.com/foaf/0.1/"
 xmlns:terms="http://purl.org/dc/terms/"
 xmlns:doap="http://usefulinc.com/ns/doap#"
 xmlns:swp-2="http://www.w3.org/2004/03/trix/swp-2/"
 xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
```

```
 xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
 xmlns:dc="http://purl.org/dc/elements/1.1/"
 xml:base="http://grid.cx/dp/1.0/"
>
<owl:Ontology rdf:about="http://grid.cx/dp/1.0/">
<owl:versionInfo>1.0</owl:versionInfo>
<owl:imports rdf:resource="http://purl.org/dc/elements/1.1/"/>
<owl:imports rdf:resource="file:/C:/Projects/Personal/phd/
ontologies/swp-3.rdf"/>
<owl:imports rdf:resource="file:///c:/Projects/Personal/phd/
terms.owl"/>
<owl:imports rdf:resource="file:///c:/Projects/Personal/phd/
dcmitype.owl"/>
<owl:imports rdf:resource="file:///c:/Projects/Personal/phd/
ontologies/java-simple.owl"/>
<owl:imports rdf:resource="http://usefulinc.com/ns/doap"/>
<owl:imports rdf:resource="http://www.mindswap.org/2003/owl/foaf"/>
</owl:Ontology>
<owl:Class rdf:about="http://grid.cx/dp/1.0/Document">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="http://xmlns.com/foaf/
       0.1/sha1" />
      <owl:someValuesFrom>
        <rdfs:Datatype rdf:about="http://www.w3.org/2001/
         XMLSchema#string"/>
      </owl:someValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="http://purl.org/dc/elements/
       1.1/title" />
      <owl:someValuesFrom>
        <rdfs:Datatype rdf:about="http://www.w3.org/2001/
         XMLSchema#string"/>
```

```
      </owl:someValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="http://grid.cx/dp/
       1.0/version" />
      <owl:someValuesFrom>
        <rdfs:Datatype rdf:about="http://www.w3.org/2001/
         XMLSchema#int"/>
      </owl:someValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="http://grid.cx/dp/1.0/
       isReplacedBy" />
      <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">
      1</owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="http://grid.cx/dp/1.0/
       dateSubmitted" />
      <owl:someValuesFrom>
        <rdfs:Datatype rdf:about="http://www.w3.org/2001/
         XMLSchema#dateTime"/>
      </owl:someValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="http://purl.org/dc/elements/
       1.1/format" />
      <owl:someValuesFrom>
```

```
        <rdfs:Datatype rdf:about="http://www.w3.org/2001/
         XMLSchema#string"/>
      </owl:someValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="http://grid.cx/dp/
       1.0/replaces" />
      <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">
       1</owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="http://grid.cx/dp/
       1.0/branch" />
      <owl:someValuesFrom>
        <rdfs:Datatype rdf:about="http://www.w3.org/2001/
         XMLSchema#int"/>
      </owl:someValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="http://grid.cx/dp/
       1.0/replaces" />
      <owl:allValuesFrom>
        <owl:Class rdf:about="http://grid.cx/dp/1.0/Document">
        </owl:Class>
      </owl:allValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="http://grid.cx/dp/
```

```
       1.0/hasClass" />
      <owl:someValuesFrom>
        <owl:Class rdf:about="http://simile.mit.edu/2004/09/
         ontologies/java#Class">
        </owl:Class>
      </owl:someValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Class rdf:about="http://xmlns.com/foaf/0.1/Document">
    </owl:Class>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="http://grid.cx/dp/1.0/
       isReplacedBy" />
      <owl:allValuesFrom>
        <owl:Class rdf:about="http://grid.cx/dp/1.0/Document">
        </owl:Class>
      </owl:allValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="http://grid.cx/dp/
       1.0/maker" />
      <owl:someValuesFrom>
        <owl:Class rdf:about="http://grid.cx/dp/1.0/Person">
        </owl:Class>
      </owl:someValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="http://grid.cx/dp/1.0/Person">
  <rdfs:subClassOf>
    <owl:Restriction>
```

```
      <owl:onProperty rdf:resource="http://xmlns.com/foaf/
       0.1/mbox" />
      <owl:someValuesFrom>
        <owl:Class rdf:about="http://www.w3.org/2001/
         XMLSchema#anyURI">
        </owl:Class>
      </owl:someValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Class rdf:about="http://xmlns.com/foaf/0.1/Person">
    </owl:Class>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="http://xmlns.com/foaf/
       0.1/name" />
      <owl:someValuesFrom>
        <rdfs:Datatype rdf:about="http://www.w3.org/2001/
         XMLSchema#string"/>
      </owl:someValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="http://grid.cx/dp/1.0/ValidityReport">
  <rdfs:subClassOf>
    <owl:Class rdf:about="http://xmlns.com/foaf/0.1/Document">
    </owl:Class>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="http://grid.cx/dp/1.0/Wikipage">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="http://grid.cx/dp/
       1.0/isPartOf" />
      <owl:someValuesFrom>
```

```
        <owl:Class rdf:about="http://usefulinc.com/ns/
         doap#Project">
        </owl:Class>
      </owl:someValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Class rdf:about="http://xmlns.com/foaf/0.1/Document">
    </owl:Class>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="http://grid.cx/dp/
       1.0/module" />
      <owl:someValuesFrom>
        <owl:Class rdf:about="http://www.w3.org/2001/
        XMLSchema#string">
        </owl:Class>
      </owl:someValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="http://grid.cx/dp/1.0/
       firstVersion" />
      <owl:someValuesFrom>
        <owl:Class rdf:about="http://grid.cx/dp/1.0/Document">
        </owl:Class>
      </owl:someValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="http://grid.cx/dp/1.0/
      firstVersion" />
      <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">
```

```
      1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="http://purl.org/dc/
       terms/created" />
      <owl:someValuesFrom>
        <rdfs:Datatype rdf:about="http://www.w3.org/2001/
         XMLSchema#dateTime"/>
      </owl:someValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="http://simile.mit.edu/2004/09/ontologies/
java#Class">
</owl:Class>
<owl:Class rdf:about="http://usefulinc.com/ns/doap#Project">
</owl:Class>
<owl:Class rdf:about="http://www.w3.org/2001/XMLSchema#anyURI">
</owl:Class>
<owl:Class rdf:about="http://www.w3.org/2001/XMLSchema#string">
</owl:Class>
<owl:Class rdf:about="http://www.w3.org/2004/03/trix/swp-2/
Authority">
</owl:Class>
<owl:Class rdf:about="http://xmlns.com/foaf/0.1/Document">
</owl:Class>
<owl:Class rdf:about="http://xmlns.com/foaf/0.1/Person">
</owl:Class>
<owl:ObjectProperty rdf:about="http://grid.cx/dp/1.0/
firstVersion">
  <rdf:type rdf:resource="&owl;FunctionalProperty" />
  <rdfs:domain>
    <owl:Class rdf:about="http://grid.cx/dp/1.0/Wikipage">
    </owl:Class>
```

```
  </rdfs:domain>
  <rdfs:range>
    <owl:Class rdf:about="http://grid.cx/dp/1.0/Document">
    </owl:Class>
  </rdfs:range>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="http://grid.cx/dp/1.0/hasClass">
  <rdfs:domain>
    <owl:Class rdf:about="http://grid.cx/dp/1.0/Document">
    </owl:Class>
  </rdfs:domain>
  <rdfs:range>
    <owl:Class rdf:about="http://simile.mit.edu/2004/09/
     ontologies/java#Class">
    </owl:Class>
  </rdfs:range>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="http://grid.cx/dp/1.0/
 isPartOf">
  <rdfs:domain>
    <owl:Class rdf:about="http://grid.cx/dp/1.0/Wikipage">
    </owl:Class>
  </rdfs:domain>
  <rdfs:range>
    <owl:Class rdf:about="http://usefulinc.com/ns/
     doap#Project">
    </owl:Class>
  </rdfs:range>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="http://grid.cx/dp/1.0/
isReplacedBy">
  <rdfs:domain>
    <owl:Class rdf:about="http://grid.cx/dp/1.0/Document">
    </owl:Class>
  </rdfs:domain>
  <rdfs:range>
```

```
    <owl:Class rdf:about="http://grid.cx/dp/1.0/Document">
    </owl:Class>
  </rdfs:range>
  <owl:inverseOf rdf:resource="http://grid.cx/dp/
   1.0/replaces" />
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="http://grid.cx/dp/
 1.0/knownCommit">
  <rdfs:domain>
    <owl:Class rdf:about="http://grid.cx/dp/1.0/
     ValidityReport">
    </owl:Class>
  </rdfs:domain>
  <rdfs:range>
    <owl:Restriction>
      <owl:onProperty rdf:resource="http://grid.cx/dp/1.0/
       knownCommit" />
      <owl:someValuesFrom>
        <owl:Class rdf:about="http://grid.cx/dp/1.0/Document">
        </owl:Class>
      </owl:someValuesFrom>
    </owl:Restriction>
  </rdfs:range>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="http://grid.cx/dp/1.0/maker">
  <rdfs:domain>
    <owl:Class rdf:about="http://grid.cx/dp/1.0/Document">
    </owl:Class>
  </rdfs:domain>
  <rdfs:domain>
    <owl:Restriction>
      <owl:onProperty rdf:resource="http://grid.cx/dp/1.0/maker" />
      <owl:someValuesFrom>
        <owl:Class rdf:about="http://grid.cx/dp/1.0/Person">
        </owl:Class>
      </owl:someValuesFrom>
```

```
    </owl:Restriction>
  </rdfs:domain>
  <rdfs:range>
    <owl:Class rdf:about="http://grid.cx/dp/1.0/Person">
    </owl:Class>
  </rdfs:range>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="http://grid.cx/dp/1.0/module">
  <rdfs:domain>
    <owl:Class rdf:about="http://grid.cx/dp/1.0/Wikipage">
    </owl:Class>
  </rdfs:domain>
  <rdfs:range>
    <owl:Class rdf:about="http://www.w3.org/2001/
     XMLSchema#string">
    </owl:Class>
  </rdfs:range>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="http://grid.cx/dp/1.0/
 recommendation">
  <rdfs:domain>
    <owl:Class rdf:about="http://grid.cx/dp/1.0/
    ValidityReport">
    </owl:Class>
  </rdfs:domain>
  <rdfs:range>
    <owl:Class rdf:about="http://www.w3.org/2001/
     XMLSchema#anyURI">
    </owl:Class>
  </rdfs:range>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="http://grid.cx/dp/1.0/
 replaces">
  <rdfs:domain>
    <owl:Class rdf:about="http://grid.cx/dp/1.0/Document">
    </owl:Class>
```

```
  </rdfs:domain>
  <rdfs:range>
    <owl:Class rdf:about="http://grid.cx/dp/1.0/Document">
    </owl:Class>
  </rdfs:range>
  <owl:inverseOf rdf:resource="http://grid.cx/dp/1.0/
   isReplacedBy" />
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="http://grid.cx/dp/1.0/
 target">
  <rdfs:domain>
    <owl:Class rdf:about="http://grid.cx/dp/1.0/
    ValidityReport">
    </owl:Class>
  </rdfs:domain>
  <rdfs:range>
    <owl:Restriction>
      <owl:onProperty rdf:resource="http://grid.cx/dp/1.0/
       branch" />
      <owl:someValuesFrom>
        <rdfs:Datatype rdf:about="http://www.w3.org/2001/
         XMLSchema#anyURI"/>
      </owl:someValuesFrom>
    </owl:Restriction>
  </rdfs:range>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="http://purl.org/dc/terms/
 isReplacedBy">
  <owl:inverseOf rdf:resource="http://purl.org/dc/terms/
  replaces" />
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="http://purl.org/dc/terms/
replaces">
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="http://xmlns.com/foaf/0.1/maker">
</owl:ObjectProperty>
```

```
<owl:ObjectProperty rdf:about="http://xmlns.com/foaf/0.1/mbox">
  <rdfs:domain>
    <owl:Class rdf:about="http://grid.cx/dp/1.0/Person">
    </owl:Class>
  </rdfs:domain>
</owl:ObjectProperty>

<owl:DatatypeProperty rdf:about="http://grid.cx/dp/1.0/branch">
  <rdfs:domain>    <owl:Class rdf:about="http://grid.cx/dp/1.0/
                    Document">
    </owl:Class>
</rdfs:domain>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="http://grid.cx/dp/1.0/caCheck">
  <rdfs:domain>    <owl:Class rdf:about="http://www.w3.org/2004/03/
                    trix/swp-2/Authority">
    </owl:Class>
</rdfs:domain>
  <rdfs:range>    <rdfs:Datatype rdf:about="http://www.w3.org/2001/
                   XMLSchema#anyURI"/>
</rdfs:range>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="http://grid.cx/dp/1.0/content">
  <rdfs:domain>    <owl:Class rdf:about="http://grid.cx/dp/1.0/
                    Wikipage">
    </owl:Class>
</rdfs:domain>
  <rdfs:domain>    <owl:Restriction>
      <owl:onProperty rdf:resource="http://grid.cx/dp/1.0/
       content" />
      <owl:someValuesFrom>
        <rdfs:Datatype rdf:about="http://www.w3.org/2001/
         XMLSchema#string"/>
      </owl:someValuesFrom>
    </owl:Restriction>
</rdfs:domain>
```

```
  <rdfs:range>    <rdfs:Datatype rdf:about="http://www.w3.org/2001/
                   XMLSchema#string"/>
</rdfs:range>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="http://grid.cx/dp/
 1.0/dateSubmitted">
  <rdfs:domain>    <owl:Class rdf:about="http://grid.cx/dp/
                    1.0/Document">
    </owl:Class>
</rdfs:domain>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="http://grid.cx/dp/1.0/
 localCommitStatus">
  <rdfs:domain>    <owl:Class rdf:about="http://www.w3.org/2004/03/
                    trix/swp-2/Authority">
    </owl:Class>
</rdfs:domain>
  <rdfs:range>    <rdfs:Datatype rdf:about="http://www.w3.org/2001/
                   XMLSchema#anyURI"/>
</rdfs:range>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="http://grid.cx/dp/1.0/
 module">
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="http://grid.cx/dp/1.0/
 recommendation">
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="http://grid.cx/dp/1.0/
 remoteCommitStatus">
  <rdfs:domain>    <owl:Class rdf:about="http://www.w3.org/2004/03/
                    trix/swp-2/Authority">
    </owl:Class>
</rdfs:domain>
  <rdfs:range>    <rdfs:Datatype rdf:about="http://www.w3.org/2001/
                   XMLSchema#anyURI"/>
</rdfs:range>
```

```
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="http://grid.cx/dp/1.0/version">
  <rdfs:domain>    <owl:Class rdf:about="http://grid.cx/dp/1.0/
                    Document">
    </owl:Class>
</rdfs:domain>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="http://grid.cx/dp/1.0/wpCheck">
  <rdfs:domain>    <owl:Class rdf:about="http://www.w3.org/2004/03/
                    trix/swp-2/Authority">
    </owl:Class>
</rdfs:domain>
  <rdfs:range>    <rdfs:Datatype rdf:about="http://www.w3.org/2001/
                   XMLSchema#anyURI"/>
</rdfs:range>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="http://purl.org/dc/elements/1.1/
 description">
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="http://purl.org/dc/elements/
 1.1/format">
  <rdfs:domain>    <owl:Class rdf:about="http://grid.cx/dp/
                    1.0/Document">
    </owl:Class>
</rdfs:domain>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="http://purl.org/dc/elements/
 1.1/title">
  <rdfs:domain>    <owl:Class rdf:about="http://grid.cx/dp/
                    1.0/Document">
    </owl:Class>
</rdfs:domain>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="http://purl.org/dc/terms/
 created">
  <rdfs:domain>    <owl:Class rdf:about="http://grid.cx/dp/1.0/
```

```
                    Wikipage">
    </owl:Class>
</rdfs:domain>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="http://usefulinc.com/ns/
 doap#revision">
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="http://xmlns.com/foaf/
 0.1/maker">
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="http://xmlns.com/foaf/
 0.1/mbox">
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="http://xmlns.com/foaf/
 0.1/name">
  <rdfs:domain>    <owl:Class rdf:about="http://grid.cx/dp/
                    1.0/Person">
    </owl:Class>
</rdfs:domain>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="http://xmlns.com/foaf/0.1/sha1">
  <rdfs:domain>    <owl:Class rdf:about="http://grid.cx/dp/
                    1.0/Document">
    </owl:Class>
</rdfs:domain>
</owl:DatatypeProperty>


<owl:Class rdf:about="http://grid.cx/dp/1.0/Document">
  <owl:disjointWith>
  <owl:Class rdf:about="http://grid.cx/dp/1.0/Wikipage">
  </owl:Class>
  </owl:disjointWith>
</owl:Class>
<owl:DatatypeProperty rdf:about="http://grid.cx/dp/1.0/branch">
  <rdfs:subPropertyOf>
```

```
  <owl:DatatypeProperty rdf:about="http://usefulinc.com/ns/
    doap#revision">
  </owl:DatatypeProperty>
  </rdfs:subPropertyOf>
</owl:DatatypeProperty>

<owl:ObjectProperty rdf:about="http://grid.cx/dp/1.0/replaces">
  <rdfs:subPropertyOf>
  <owl:ObjectProperty rdf:about="http://purl.org/dc/
   terms/replaces">
  </owl:ObjectProperty>
  </rdfs:subPropertyOf>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="http://grid.cx/dp/1.0/
 isReplacedBy">
  <rdfs:subPropertyOf>
  <owl:ObjectProperty rdf:about="http://purl.org/dc/terms/
   isReplacedBy">
  </owl:ObjectProperty>
  </rdfs:subPropertyOf>
</owl:ObjectProperty>

<owl:DatatypeProperty rdf:about="http://grid.cx/dp/1.0/content">
  <rdfs:subPropertyOf>
  <owl:DatatypeProperty rdf:about="http://purl.org/dc/elements/
   1.1/description">
  </owl:DatatypeProperty>
  </rdfs:subPropertyOf>
</owl:DatatypeProperty>

<owl:ObjectProperty rdf:about="http://grid.cx/dp/1.0/maker">
  <rdfs:subPropertyOf>
  <owl:ObjectProperty rdf:about="http://xmlns.com/
   foaf/0.1/maker">
  </owl:ObjectProperty>
```

```
  </rdfs:subPropertyOf>
</owl:ObjectProperty>

<owl:DatatypeProperty rdf:about="http://grid.cx/dp/1.0/version">
  <rdfs:subPropertyOf>
  <owl:DatatypeProperty rdf:about="http://usefulinc.com/
   ns/doap#revision">
  </owl:DatatypeProperty>
  </rdfs:subPropertyOf>
</owl:DatatypeProperty>

</rdf:RDF>
```

Appendix B

B.1 Instance Examples

This appendix gives a complete overview of a instance of the DP ontology, including DOAP (FOAF), and Simple Java Ontology instances. Each section has been annotated for easy cross-referencing. Since all RDF is decomposed into Named Graphs, we have chosen to use the TriG concrete syntax for its simplicity.

B.2 DP

```
@prefix swp:     <http://www.w3.org/2004/03/trix/swp-2/> .
@prefix ns0:     <http://simile.mit.edu/2004/09/ontologies/java#> .
@prefix ns1:     <http://grid.cx/dp/1.0/> .
@prefix dcterms:  <http://purl.org/dc/terms/> .
@prefix dc:      <http://purl.org/dc/elements/1.1/> .
@prefix doap:    <http://usefulinc.com/ns/doap#> .
@prefix rdfs:    <http://www.w3.org/2000/01/rdf-schema#> .
@prefix foaf:    <http://xmlns.com/foaf/0.1/> .
@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix :        <#> .

<urn:uuid:C9180AE0-21A8-11DB-8270-8FCC55490BC5> {
   <java:org.embl.ebi.escience.scuflui.workbench>
      a       ns0:Package .

   <java:org.embl.ebi.escience.scuflui.workbench.Workbench>
```

```
    a       ns0:Class ;
    ns0:contained
      <java:org.embl.ebi.escience.scuflui.workbench> ;
    ns0:located
      <https://localhost:8443/webdav/taverna/taverna/
      org/embl/ebi/escience/scuflui/workbench/Workbench/
      1/1/Workbench.java> ;
    ns0:uses
      <java:java.awt.Dimension> ,
      <java:java.awt.event.WindowEvent> ,
      <java:org.embl.ebi.escience.scufl.ScuflModel> ,
      <java:java.awt.event.ActionEvent> ,
      <java:java.lang.String> ,
      <java:org.embl.ebi.escience.scuflui.workbench.XScuflFrame> ,
      <java:java.lang.Exception> ,
      <java:org.embl.ebi.escience.scuflui.workbench.ExplorerFrame> ,
      <java:org.embl.ebi.escience.scuflui.workbench.DiagramFrame> ,
      <java:org.embl.ebi.escience.scufl.parser.XScuflParser> ,
      <java:java.awt.Toolkit> ,
      <java:javax.swing.*> ,
      <java:java.awt.event.WindowAdapter> ,
      <java:java.io.File> ,
      <java:java.awt.event.ActionListener> ,
      <java:java.lang.System> ;
    swp:inGraph
      <urn:uuid:C9180AE0-21A8-11DB-8270-8FCC55490BC5> .

<https://localhost:8443/webdav/taverna/taverna/org/embl/ebi/
    escience/scuflui/workbench/Workbench/1/1/Workbench.java>
    a       ns1:Document ;
    ns1:branch "1"^^<http://www.w3.org/2001/XMLSchema#int> ;
    ns1:hasClass
      <java:org.embl.ebi.escience.scuflui.workbench.Workbench> ;
    ns1:maker <mailto:jjc@hpl.hp.com> ;
    ns1:revision "1"^^<http://www.w3.org/2001/XMLSchema#int> ;
    dc:format "text/x-java-source" ;
```

```
      dc:title "Workbench.java" ;
      dcterms:dateSubmitted
        "Tue Aug 01 22:57:55 BST 2006"^^
        <http://www.w3.org/2001/XMLSchema#dateTime> ;
      dcterms:isPartOf <http://www.taverna.sourceforge.net> ;
      dcterms:modified "Mon May 19 16:58:57 BST 2003"^^
        <http://www.w3.org/2001/XMLSchema#dateTime> ;
      doap:module "taverna" ;
      swp:inGraph
        <urn:uuid:C9180AE0-21A8-11DB-8270-8FCC55490BC5> ;
      foaf:sha1 "85073015fc760ceda638fa03cbaae301256ed462" .
}

<urn:uuid:CA2CAF30-21A8-11DB-8270-9859210973A2> {
   <https://localhost:8443/JSPWiki/Wiki.jsp?page=
  org.embl.ebi.escience.scuflui.workbench.Workbench>
      a       ns1:Wikipage ;
      ns1:content "description content" ;
      ns1:firstVersion <https://localhost:8443/webdav/taverna/
        taverna/org/embl/ebi/escience/scuflui/workbench/Workbench/
        1/1/Workbench.java> ;
      dcterms:created "Tue Aug 01 22:57:55 BST 2006"^^
        <http://www.w3.org/2001/XMLSchema#dateTime> ;
      swp:inGraph <urn:uuid:CA2CAF30-21A8-11DB-8270-9859210973A2> .
}

<urn:uuid:CB28C270-21A8-11DB-8270-E4B573A1C2E3> {
   <urn:uuid:CB28C270-21A8-11DB-8270-E4B573A1C2E3>
      swp:assertedBy
        <urn:uuid:CB28C270-21A8-11DB-8270-E4B573A1C2E3> ;
      swp:authority <mailto:jjc@hpl.hp.com> ;
      swp:signature
        """t5Lzx8PY3oRWrwoXknv4iMdciR5oVFyVHUh9AQK4/EdK6a5i6z4T2mni
        zZT0/EAW2xQ9ME85ZyHK3mEf15QNFUnOXedgBUZIshK7oXflr3/ifOZFCy4
        mrfmITZAIY6HYwlrj1qTLXgB3//NfSCtkKNEsMI8yzvsdMwTWLgpI5Ns
        =""""^^<http://www.w3.org/2001/XMLSchema#base64Binary> ;
```

```
    swp:signatureMethod swp:JjcRdfC14N-rsa-sha1 ;
    swp:validFrom "Tue Aug 01 22:57:58 BST 2006"^^
      <http://www.w3.org/2001/XMLSchema#dateTime> ;
    swp:validUntil "Mon Aug 01 22:57:58 BST 2016"^^
      <http://www.w3.org/2001/XMLSchema#dateTime> .

<mailto:jjc@hpl.hp.com>
    rdfs:label "Jeremy J Carroll" ;
    swp:X509Certificate
      """MIID7zCCA1igAwIBAgIBCDANBgkqhkiG9w0BAQQFADC
      BoTELMAkGA1UEBhMCVUsxEjAQBgNVBAgTCUhhbXBzaGlyZTEUMBIGA1UEBx
      MLU291dGhhbXB0b24xIjAgBgNVBAoTGVVuaXZlcnNpdHkgb2YgU291dGhhb
      XB0b24xDTALBgNVBAsTBERTU0UxDjAMBgNVBAMTBURQIENBMSUwIwYJKoZI
      hvcNAQkBFhZlcncwMXJAZWNzLnNvdG9uLmFjLnVrMB4XDTA1MDEwMjEzNTA
      wNloXDTA2MDEwMjEzNTAwNlowgacxCzAJBgNVBAYTAlVLMRAwDgYDVQQIEw
      dCcmlzdG9sMRAwDgYDVQQHEwdCcmlzdG9sMRgwFgYDVQQKEw9IZXdsZXR0I
      FBhY2thcmQxHTAbBgNVBAsTFEhld2xldHQgUGFja2FyZCBMYWJzMRcwFQYD
      VQQDEw5KZXJlbXkgQ2Fycm9sbDEiMCAGCSqGSIb3DQEJARYTampjQGhwbGI
      uaHBsLmhwLmNvbTCBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEAuJZtp1
      LBfVRdVhWbSHxXcL12QKJohLruaz6mQgc2p457zoOhWBgKRRt619Loe/r9r
      kOww8FsGEzEdJ8jClDUMPikJEQgHHh77CDS/Yij/gANXwDr2JF7tm+ggpU4
      xfXD/BBON/V9QmldmmECNOnlJCqSew1navIkaeLewSYkXhUCAwEAAaOCAS0
      wggEpMAkGA1UdEwQCMAAwLAYJYIZIAYb4QgENBB8WHU9wZW5TU0wgR2VuZX
      JhdGVkIENlcnRpZmljYXRlMB0GA1UdDgQWBBQuFPsaZQrqZ/nU+/GHRsrGI
      s+lhjCBzgYDVR0jBIHGMIHDgBTYGUpxKuD0zkwpARfVPq9eSgxw9aGBp6SB
      pDCBoTELMAkGA1UEBhMCVUsxEjAQBgNVBAgTCUhhbXBzaGlyZTEUMBIGA1U
      EBxMLU291dGhhbXB0b24xIjAgBgNVBAoTGVVuaXZlcnNpdHkgb2YgU291dG
      hhbXB0b24xDTALBgNVBAsTBERTU0UxDjAMBgNVBAMTBURQIENBMSUwIwYJK
      oZIhvcNAQkBFhZlcncwMXJAZWNzLnNvdG9uLmFjLnVrggEAMA0GCSqGSIb3
      DQEBBAUAA4GBABeWT2cnm7ybPePb/6QToRV1XRYulN4x/0XqZyqfqGPz9zN
      GEy1KXncoIRU3Iw5h32N2HXnce2M/YlOP49r1ucdhiGlJG3M0XAEUY7c8gw
      afuK7BZZYD6cdxfAY9g8OwezR+MdwJYwFSUVKc67k6apiJEygy8MPLP3CEp
      d4Eging"""^^
        <http://www.w3.org/2001/XMLSchema#base64Binary> ;
    foaf:mbox <mailto:jjc@hpl.hp.com> .
```

```
   <urn:uuid:C9180AE0-21A8-11DB-8270-8FCC55490BC5>
      swp:assertedBy
        <urn:uuid:CB28C270-21A8-11DB-8270-E4B573A1C2E3> ;
      swp:digest
        "OTIwN2E3Nzg3YTYzYzA4ODNhMGRjZjQ1YzJmYjVmZTI5NmNkOWZmYQ=="^^
        <http://www.w3.org/2001/XMLSchema#base64Binary> ;
      swp:digestMethod swp:JjcRdfC14N-sha1 .

   <urn:uuid:CA2CAF30-21A8-11DB-8270-9859210973A2>
      swp:assertedBy
        <urn:uuid:CB28C270-21A8-11DB-8270-E4B573A1C2E3> ;
      swp:digest
        "MmY5MGMyMGI5Y2I3YjI0MTE3NDNhN2FjNmNkMmIxNjNkMmQ1NjNiMw=="^^
        <http://www.w3.org/2001/XMLSchema#base64Binary> ;
      swp:digestMethod swp:JjcRdfC14N-sha1 .
}
```

B.3 DOAP

```
<Project
 rdf:about="http://www.ecs.soton.ac.uk/~erw/AcmeGrid"
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:swp="http://www.w3.org/2004/03/trix/swp-2/"
 xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
 xmlns="http://usefulinc.com/ns/doap#"
 xmlns:doap="http://usefulinc.com/ns/doap#"
 xmlns:foaf="http://xmlns.com/foaf/0.1/"
 xmlns:admin="http://webns.net/mvcb/">
 <name>AcmeGrid</name>
 <shortname>AcmeGrid</shortname>
 <shortdesc>AcmeGrid</shortdesc>
 <description>AcmeGrid</description>
 <homepage
  rdf:resource="http://www.ecs.soton.ac.uk/~erw/AcmeGrid" />
 <programming-language>Java</programming-language>
```

```
 <workpackage>
    <Workpackage
     rdf:about="http://www.ecs.soton.ac.uk/~erw/AcmeGrid/WP2">
      <workpackageName>Integration</workpackageName>
    </Workpackage>
  </workpackage>
  <workpackage>
    <Workpackage
     rdf:about="http://www.ecs.soton.ac.uk/~erw/AcmeGrid/WP3">
      <workpackageName>Application Scenario</workpackageName>

    </Workpackage>
  </workpackage>
  <workpackage>
    <Workpackage
     rdf:about="http://www.ecs.soton.ac.uk/~erw/AcmeGrid/WP1">
      <workpackageName>Project Management</workpackageName>
    </Workpackage>
  </workpackage>
 <license
  rdf:resource="http://usefulinc.com/doap/licenses/lgpl" />

 <knownCA>
MIIFNTCCBB2gAwIBAgIBATANBgkqhkiG9w0BAQUFADCBsTEXMBUGA1UEAxMOR1JJ
QSBUZXN0IENBIDIxCzAJBgNVBAYTAlVLMRQwEgYDVQQHEwtTb3V0aGFtcHRvbjES
MBAGA1UECBMJSGFtcHNoaXJlMR0wGwYDVQQKExRJVCBJbm5vdmF0aW9uIENlbnRy
ZTESMBAGA1UECxMJVGVjaFN1aXRlMSwwKgYJKoZIhvcNAQkBFh1lcndAaXQtaW5u
b3ZhdGlvbi5zb3Rvbi5hYy51azAeFw0wNjA2MTYxNDQ2MTZaFw0xNjA2MTMxNDQ2
MTZaMIGxMRcwFQYDVQQDEw5HUklBIFRlc3QgQ0EgMjELMAkGA1UEBhMCVUsxFDAS
BgNVBAcTC1NvdXRoYW1wdG9uMRIwEAYDVQQIEwlIYW1wc2hpcmUxHTAbBgNVBAoT
FElUIElubm92YXRpb24gQ2VudHJlMRIwEAYDVQQLEwlUZWNoU3VpdGUxLDAqBgkq
hkiG9w0BCQEWHWVyd0BpdC1pbm5vdmF0aW9uLnNvdG9uLmFjLnVrMIIBIjANBgkq
hkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAv5XKg1X2EKKPFj1GSNHsPgHD31V9CsPN
rhVRtDwTqng1XKYxuvyu37cOyndMO++sTq3MRrt81nD8wKROJ0D0xqtPAo/4lu5s
gCErrQM8dNi1md3Cxa/ys3vuuqZUvgvj6Lm6EPZXbvK17C5EG1u0JKfcdPg3kscX
```

mv+8NO0d3u1HIUXR839RWKZ3e7VKQGldyW82pgC2vO8Uss8zJudr+LHduiqGFmwb
SLnYgDmH20QfvDnJO0LanqVzBdXXsFt204s/+T4Q1rU3a8Si6PokYvD8Wc4zQE0k
xhym/r11djRMM17+yrroSz45xoTfsFlDBCeLTGQ3y1s1Wev4ZbvOhwIDAQABo4IB
VDCCAVAwDwYDVR0TAQH/BAUwAwEB/zAdBgNVHQ4EFgQUsjkeD0g02XqXFlO4LWJ8
noDoQIEwgd4GA1UdIwSB1jCB04AUsjkeD0g02XqXFlO4LWJ8noDoQIGhgbekgbQw
gbExFzAVBgNVBAMTDkdSSUEgVGVzdCBDQSAyMQswCQYDVQQGEwJVSzEUMBIGA1UE
BxMLU291dGhhbXB0b24xEjAQBgNVBAgTCUhhbXBzaGlyZTEdMBsGA1UEChMUSVQg
SW5ub3ZhdGlvbiBDZW50cmUxEjAQBgNVBAsTCVRlY2hTdWl0ZTEsMCoGCSqGSIb3
DQEJARYdZXJ3QGl0LWlubm92YXRpb24uc290b24uYWMudWuCAQEwCwYDVR0PBAQD
AgEGMBEGCWCGSAGG+EIBAQQEAwIABzAdBglghkgBhvhCAQ0EEBYOR1JJQSBUZXN0
IENBIDIwDQYJKoZIhvcNAQEFBQADggEBAAwKOIbKwTX6IOxab0NdPp2X7EQ8K2CV
hmsNCBWt2vk3JMxN6ymde9zPGam3qTQ3DoCGAPa+R4F7kH+PM96XZ4OVR0D8OlAS
pbzYMiKEDPhJQZA+JczZ8rvfyOxJZX5w6HfbwHr9tAJLsc/2XTOHFhvx4S9Ly8QF
018Uk/FX9Fv4W8m4kXidSfZGrtFBL1AyZHUp6QaZ23ESzERBqEQLkNhdA3rmzPHV
V1oZ6eU+eIoHnGIwGWZssfztqQfjcF0jV+W9CKLdzbNv+sKPW7aE8KHxN/dLBsUZ
VAPKyQcwSU6uURMzOCC747ye/GpTVHDsozkgdDMdiqwTTbOhcThoevU=
</knownCA>
 <developer>
  <foaf:Person>
     <foaf:name>Rowland Watkins</foaf:name>
     <foaf:mbox>erw@it-innovation.soton.ac.uk</foaf:mbox>
     <swp:certificate>
MIIFNTCCBB2gAwIBAgIBEjANBgkqhkiG9w0BAQQFADCBsTEXMBUGA1UEAxMOR1JJ
QSBUZXN0IENBIDIxCzAJBgNVBAYTAlVLMRQwEgYDVQQHEwtTb3V0aGFtcHRvbjES
MBAGA1UECBMJSGFtcHNoaXJlMR0wGwYDVQQKExRJVCBJbm5vdmF0aW9uIENlbnRy
ZTESMBAGA1UECxMJVGVjaFN1aXRlMSwwKgYJKoZIhvcNAQkBFh1lcndAaXQtaW5u
b3ZhdGlvbi5zb3Rvbi5hYy51azAeFw0wNjEwMDUwODQ4MTNaFw0wNzEwMDUwODQ4
MTNaMIGyMRgwFgYDVQQDEw9Sb3dsYW5kIFdhdGtpbnMxCzAJBgNVBAYTAlVLMRQw
EgYDVQQHEwtTb3V0aGFtcHRvbjESMBAGA1UECBMJSGFtcHNoaXJlMR0wGwYDVQQK
ExRJVCBJbm5vdmF0aW9uIENlbnRyZTESMBAGA1UECxMJVGVjaFN1aXRlMSwwKgYJ
KoZIhvcNAQkBFh1lcndAaXQtaW5ub3ZhdGlvbi5zb3Rvbi5hYy51azCCASIwDQYJ
KoZIhvcNAQEBBQADggEPADCCAQoCggEBAOEbQJuYRyYXWV/EiMlT+Ku37aPvGAUi
BdSeI+a2zOsXH/AwUV19IabIzQqSY7H/KtPGN3UBjAyN3VgImxdK+ojhBpUV6New
J87ExAqYCbqZNFcu6Oe6cZTMQeAS1lGllXcdR/d74peJJqY+Yj6Ur+qNNpgxpAD4
oF8TxmZMpJCXdXexFc5D3gGprtPLkPMH3KvPzjrYhKYLKYJV+paQ8l9WbO90+Ino
MF9eWjkiGrqcvSD4zoTQg705T6p1r+UIilhrFKHSTOS4FLnkTIppm7AC+AY+txN9

```
M9hYD4Os+tsN7yxc/Tlt121pnGOvs7J6llduJ6eVGflxbN7LttRsg38CAwEAAaOC
AVMwggFPMAwGA1UdEwEB/wQCMAAwHQYDVR0OBBYEFA0/1D9J4sc5um6BxY5GeLVL
vf82MIHeBgNVHSMEgdYwgdOAFLI5Hg9INNl6lxZTuC1ifJ6A6ECBoYG3pIG0MIGx
MRcwFQYDVQQDEw5HUklBIFRlc3QgQ0EgMjELMAkGA1UEBhMCVUsxFDASBgNVBAcT
C1NvdXRoYW1wdG9uMRIwEAYDVQQIEwlIYW1wc2hpcmUxHTAbBgNVBAoTFElUIElu
bm92YXRpb24gQ2VudHJlMRIwEAYDVQQLEwlUZWNoU3VpdGUxLDAqBgkqhkiG9w0B
CQEWHWVyd0BpdC1pbm5vdmF0aW9uLnNvdG9uLmFjLnVrggEBMAsGA1UdDwQEAwIE
sDARBglghkgBhvhCAQEEBAMCBaAwHwYJYIZIAYb4QgENBBIWEEdEQ0QgQ2VydGlm
aWNhdGUwDQYJKoZIhvcNAQEEBQADggEBACrb7uI7ti4c8NV1x3486SKmVPutizrM
p423QLoDJuJVVJUqpniKyr0g4Bo9+yUzKhaGFMquf9pNubopwSf+3R6aIkcMqRKA
F37YyCkzaVgt7j0p3pRymTcHv9c9xLHfvWZv2SyzPgYZB0Yq2jqSttCF2Yoe2uZQ
/iCcj7s7ZuKubPOpoYDSh455yPn99SLQs5H3NYe+KYMqNntO6a9yDhifaryF6O0l
jbixyNp9kSioocu5pqlE69tNA9dEe+/qkTaLAK/GSSgbUscjxhL69Iqc3/m/AAXr
houJWMWr+ggJIzIGUAaZBOf6qHzZ3PFebPt/yiWTjoeFuOutXVnIilo=
</swp:certificate>
  <workpackage
   rdf:resource="http://www.ecs.soton.ac.uk/~erw/AcmeGrid/WP2" />

  </foaf:Person>
 </developer>
 <repository>
   <SVCRepository>
     <browse
      rdf:resource='http://www.ecs.soton.ac.uk/~erw/Wiki' />
     <location
      rdf:resource='http://www.ecs.soton.ac.uk/~erw/sparql' />
   </SVCRepository>
 </repository>
</Project>
```

B.4 Simple Java Ontology

```
<java:org.embl.ebi.escience.scuflui.workbench>
   a       ns0:Package .
```

```
<java:org.embl.ebi.escience.scuflui.workbench.Workbench>
   a       ns0:Class ;
   ns0:contained
     <java:org.embl.ebi.escience.scuflui.workbench> ;
   ns0:located
     <https://localhost:8443/webdav/taverna/taverna/
     org/embl/ebi/escience/scuflui/workbench/Workbench/
     1/1/Workbench.java> ;
   ns0:uses
     <java:java.awt.Dimension> ,
     <java:java.awt.event.WindowEvent> ,
     <java:org.embl.ebi.escience.scufl.ScuflModel> ,
     <java:java.awt.event.ActionEvent> ,
     <java:java.lang.String> ,
     <java:org.embl.ebi.escience.scuflui.workbench.XScuflFrame> ,
     <java:java.lang.Exception> ,
     <java:org.embl.ebi.escience.scuflui.workbench.ExplorerFrame> ,
     <java:org.embl.ebi.escience.scuflui.workbench.DiagramFrame> ,
     <java:org.embl.ebi.escience.scufl.parser.XScuflParser> ,
     <java:java.awt.Toolkit> ,
     <java:javax.swing.*> ,
     <java:java.awt.event.WindowAdapter> ,
     <java:java.io.File> ,
     <java:java.awt.event.ActionListener> ,
     <java:java.lang.System> ;
```

Appendix C

C.1 Canonicalisation Examples

This appendix includes examples where Carroll (2003)'s canonicalisation algorithm succeeds with a limited number of blank nodes and fails with a graph comprised of only blank nodes. We have taken an OWL class from the WordNet ontology to show where blank nodes can be canonicalised, and the Petersen graph to demonstrate how Carroll's algorithm fails. To prove that the Petersen graph can be canonicalised, we show the use of **nauty** as a valid alternative. We include two TriG-serialised RDF graphs representing the OWL WordNet ontology class and Petersen graph, along with a step-by-step example of how **nauty** can verify the equality of two graphs.

C.1.1 The WordNet Ontology

To show that Carroll's algorithm can be used with a limited number of blank nodes and that ***nauty*** canonicalisation is incompatible, we took a class from the WordNet ontology[1] and canonicalised it using both algorithms. Figure C.1 shows the original RDF graph together with URI labels. The structure for this graph was generated using the W3C RDF Validator Service[2].

The most complicated part of the NounWordSense class is the **owl:unionOf** OWL construct. This is commonly encoded as an RDF Collection in RDF/XML-ABBREV concrete syntax. Since the **owl:unionOf** construct is only applied to

[1] Available at http://www.w3.org/2006/03/wn/wn20/download/wn20full.zip, accessed on 07.02.2007.

[2] http://www.w3.org/RDF/Validator/.

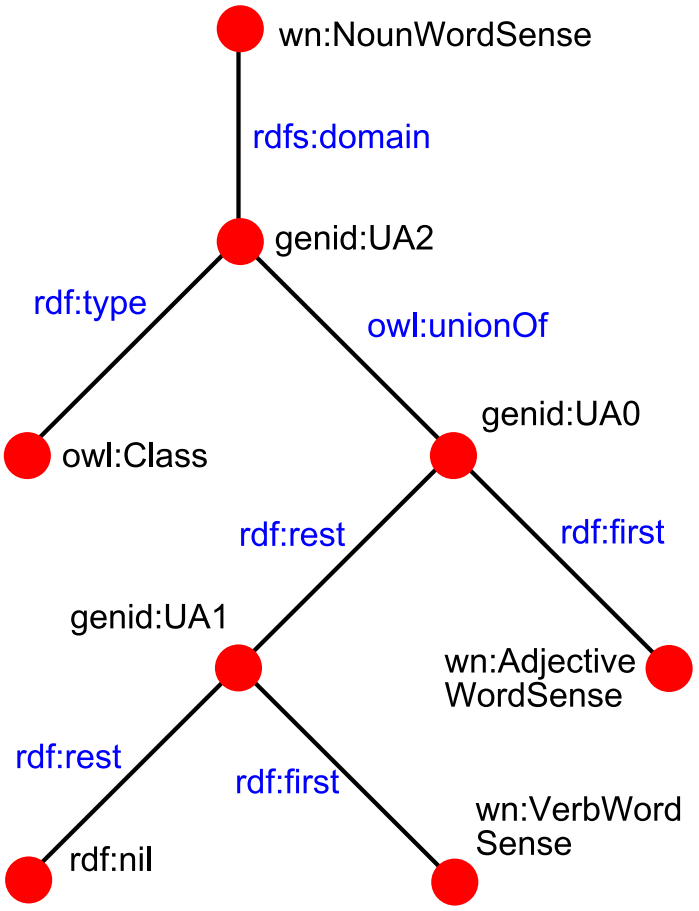

FIGURE C.1: WordNet NounWordSense Class Labelled Graph.

two objects we are left with three blank nodes; this means Carroll's algorithm can relabel reliably.

C.1.1.1 TriG-Serialised RDF Graphs

Here we present two *different* serialisations of the WordNet NounWordSense class in the TriG concrete syntax. We include canonical representations and SHA-512 digests that prove that Carroll's algorithm is capable of serialising simple RDF Collections.

TriG serialisation for NounWordSense Class:

```
@prefix rdfs:    <http://www.w3.org/2000/01/rdf-schema#> .
@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix owl:     <http://www.w3.org/2002/07/owl#> .
@prefix wn20schema: <https://example.org/wnfull.rdfs#> .
```

```
<urn:uuid:E192F360-226F-11DB-94B3-E05EDA46CF20> {
   wn20schema:NounWordSense
     rdfs:domain
       [ a       owl:Class ;
         owl:unionOf
           [ rdf:first wn20schema:AdjectiveWordSense ;
              rdf:rest
                 [ rdf:first wn20schema:VerbWordSense ;
                    rdf:rest () ;
                 ] ;
           ]
       ] ;
}
```

Canonical string based on Carroll's algorithm:

```
[urn:uuid:E192F360-226F-11DB-94B3-E05EDA46CF20,
_:g000001 http://www.w3.org/1999/02/22-rdf-syntax-ns#first
          https://example.org/wnfull.rdfs#AdjectiveWordSense,
_:g000001 http://www.w3.org/1999/02/22-rdf-syntax-ns#rest
          _:g000002,
_:g000002 http://www.w3.org/1999/02/22-rdf-syntax-ns#first
          https://example.org/wnfull.rdfs#VerbWordSense,
_:g000002 http://www.w3.org/1999/02/22-rdf-syntax-ns#rest
          http://www.w3.org/1999/02/22-rdf-syntax-ns#nil,
_:g000003 http://www.w3.org/1999/02/22-rdf-syntax-ns#type
          http://www.w3.org/2002/07/owl#Class,
_:g000003 http://www.w3.org/2002/07/owl#unionOf _:g000001,
https://example.org/wnfull.rdfs#NounWordSense
          http://www.w3.org/2000/01/rdf-schema#domain _:g000003]
```

SHA-512 digital digest:

```
YmFkNzI1ZGVlOTg4Njg1Y2NmODc3YWZkZjQzMTU0YzZjNWQ2NTY2Yzg1NzkOZGExYzN
jOTg2NTIxMWQzZTYxYTY1YjM2MTJhOTBlYzViYTk1MTQ3ZjEOODVlNTNhZmY4Zjk5Nj
Y3ZmJjMTQxMjgOMWQyZDJmODViZmQ3ZjU1MzU=
```

Another TriG serialisation for NounWordSense Class:

```
@prefix rdfs:    <http://www.w3.org/2000/01/rdf-schema#> .
@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix owl:     <http://www.w3.org/2002/07/owl#> .

<urn:uuid:E192F360-226F-11DB-94B3-E05EDA46CF20> {
<https://example.org/wnfull.rdfs#NounWordSense>
          rdfs:domain
                  [ a       owl:Class ;
                    owl:unionOf
(<https://example.org/wnfull.rdfs#AdjectiveWordSense>
<https://example.org/wnfull.rdfs#VerbWordSense>)
                  ] .
}
```

Canonical string based on Carroll's algorithm:

```
[urn:uuid:E192F360-226F-11DB-94B3-E05EDA46CF20,
_:g000001 http://www.w3.org/1999/02/22-rdf-syntax-ns#first
          https://example.org/wnfull.rdfs#AdjectiveWordSense,
_:g000001 http://www.w3.org/1999/02/22-rdf-syntax-ns#rest
          _:g000002,
_:g000002 http://www.w3.org/1999/02/22-rdf-syntax-ns#first
          https://example.org/wnfull.rdfs#VerbWordSense,
_:g000002 http://www.w3.org/1999/02/22-rdf-syntax-ns#rest
          http://www.w3.org/1999/02/22-rdf-syntax-ns#nil,
_:g000003 http://www.w3.org/1999/02/22-rdf-syntax-ns#type
          http://www.w3.org/2002/07/owl#Class,
_:g000003 http://www.w3.org/2002/07/owl#unionOf
          _:g000001,
https://example.org/wnfull.rdfs#NounWordSense
          http://www.w3.org/2000/01/rdf-schema#domain _:g000003]
```

SHA-512 digital digest:

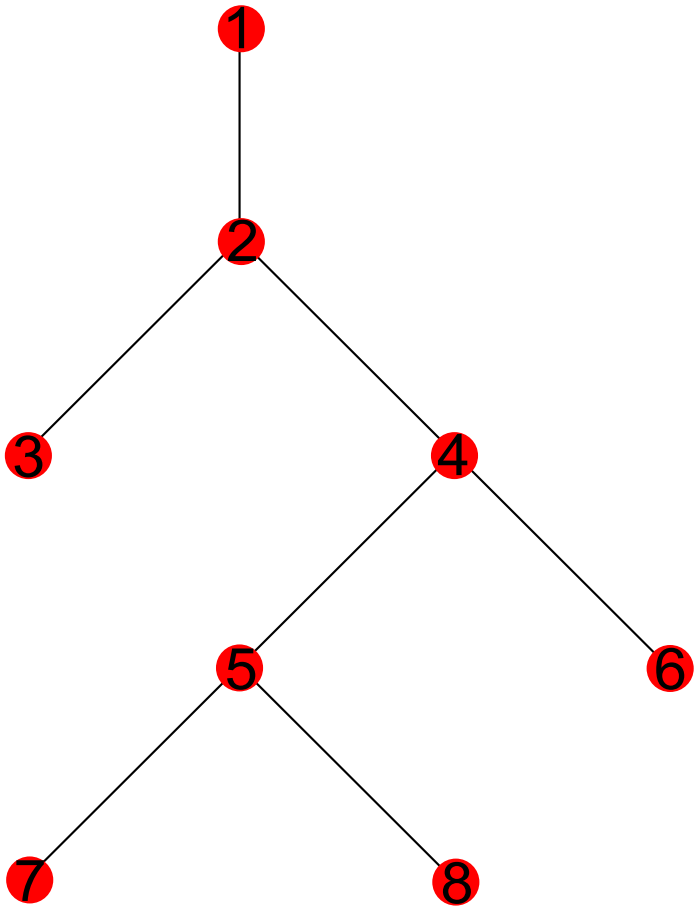

FIGURE C.2: WordNet NounWordSense Class Graph A

YmFkNzI1ZGVlOTg4Njg1Y2NmODc3YWZkZjQzMTU0YzZjNWQ2NTY2Yzg1NzkOZGExYzN
jOTg2NTIxMWQzZTYxYTY1YjM2MTJhOTBlYzViYTk1MTQ3ZjEOODVlNTNhZmY4Zjk5Nj
Y3ZmJjMTQxMjgOMWQyZDJmODViZmQ3ZjU1MzU=

C.1.1.2 Comparison to *nauty*

The graph represented in Figure C.1 is one of the few examples that can be canonicalised by both Carroll's algorithm and ***nauty***. Unfortunately, due to the distinct differences between each algorithm, they are incompatible. Carroll's algorithm acknowledges labelled arcs and re-arranges triples based on *lexicographic* ordering, whereas ***nauty*** takes a far more complicated approach based on isomorphism groups [McKay (2006)].

Figure C.1 includes a number labelled version of the NounWordSense OWL class. We have used this as the basis for input into ***nauty***.

While ***nauty*** uses its own graph encodings known as *graph6* and *sparse6* respectively [McKay (2006)], we have chosen to show the input of graphs into ***nauty*** using its interactive **dreadnaut** shell for the sake of repeatability. Below shows the canonical result of Figure C.2:

```
Dreadnaut version 2.4 (32 bits).
> c -a -m       turn getcanon on, group writing off
> n=8 g         enter graph
 0 : 1;
 1 : 2 3;
 2 : ;
 3 : 4 5;
 4 : 6 7;
 5 : ;
 6 : ;
 7 : ;
> x             execute
4 orbits; grpsize=8; 3 gens; 8 nodes; maxlev=3
tctotal=10; canupdates=1; cpu time = 0.01 seconds
> b
 5 0 6 7 2 1 4 3
  0 :  7;
  1 :  5;
  2 :  6;
  3 :  6;
  4 :  5;
  5 :  1 4 7;
  6 :  2 3 7;
  7 :  0 5 6;
>
```

The ***nauty*** below proves that Figure C.2 and Figure C.3 are identical:

```
Dreadnaut version 2.4 (32 bits).
> c -a -m       turn getcanon on, group writing off
> n=8 g         enter first graph
```

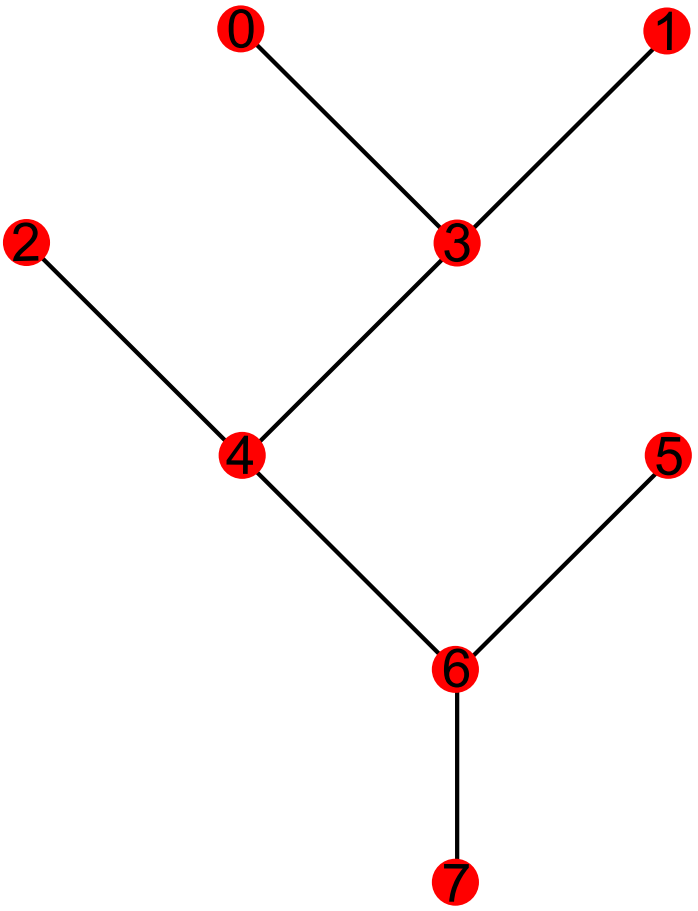

FIGURE C.3: WordNet NounWordSense Class Graph B

```
 0 : 1;
 1 : 2 3;
 2 : ;
 3 : 4 5;
 4 : 6 7;
 5 : ;
 6 : ;
 7 : ;
> x @            execute and save result
4 orbits; grpsize=8; 3 gens; 8 nodes; maxlev=3
tctotal=10; canupdates=1; cpu time = 0.00 seconds
> g              enter second graph
 0 : 3;
 1 : 3;
 2 : 4;
 3 : 4;
```

```
 4 : 6;
 5 : 6;
 6 : 7;
 7 : ;
> x             execute
4 orbits; grpsize=8; 3 gens; 8 nodes; maxlev=3
tctotal=10; canupdates=1; cpu time = 0.00 seconds
> ##            compare with first graph
h and h' are identical.
 0-0 1-3 2-1 3-4 4-6 5-2 6-5 7-7
```

C.1.1.3 Summary

While there exist simple graphs that both Carroll's algorithm and ***nauty*** can canonicalise, their outputs are very different. Carroll's algorithm is designed for RDF and takes into account labelled vertices and edges that are reordered lexicographically. ***nauty*** takes a more analytical approach to graph isomorphism that does not re-order based on labelled vertices and edges.

C.1.2 The Petersen Graph

In this second set of canonicalisation examples we show a graph that cannot be reliably canonicalised by Carroll's algorithm, namely, the Petersen Graph.

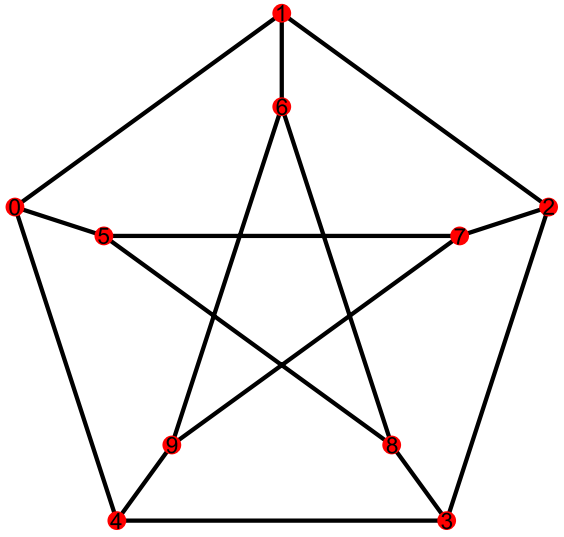

FIGURE C.4: Common Petersen Graph Representation

Petersen graphs can be presented in more than one arrangement. Figure C.4 shows a common representation as a pentagon with a star inside, with five spokes; Figure C.5 shows a slightly different arrangement with two crossings. Every vertex in each graph has been numbered, zero through nine.

We will first of all create RDF Named Graph representations of Figure C.4 and Figure C.5, then go on to their ***nauty*** representation.

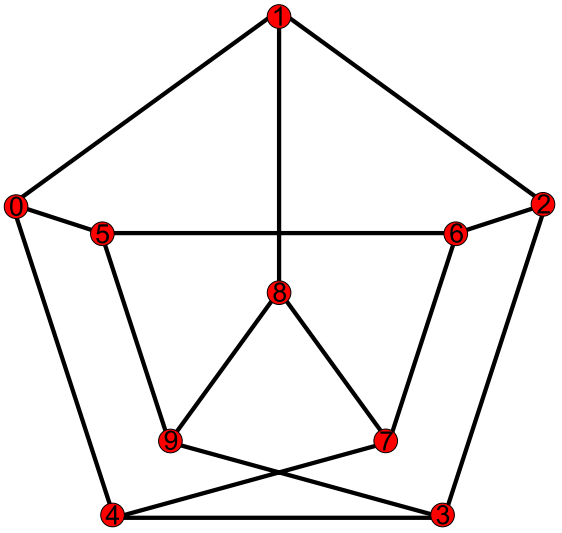

FIGURE C.5: Petersen Graph with Two Crossings

C.1.2.1 Carroll's Algorithm

For each representation, we will create a TriG RDF graph, then calculate its canonical form and SHA-512 hash.

TriG-Serialised RDF Graphs TriG serialisation for Figure C.4:

```
<urn:petersen:graph:a:11-01-2007>
{_:b1  <urn:predicate> _:b2 , _:b3 , _:b4 ;
          <urn:predicate>
                  [
                  ] .
    _:b5  <urn:predicate>
                  [
                  ] .
    _:b6  <urn:predicate> _:b5 , _:b7 , _:b8 ;
```

```
          <urn:predicate>
                  [
                  ] .
    _:b7  <urn:predicate> _:b6 , _:b9 , _:b10 ;
          <urn:predicate>
                  [
                  ] .
    _:b8  <urn:predicate> _:b6 , _:b2 , _:b4 ;
          <urn:predicate>
                  [
                  ] .
    _:b9  <urn:predicate>
                  [
                  ] .
    _:b10
          <urn:predicate> _:b7 , _:b3 , _:b4 ;
          <urn:predicate>
                  [
                  ] .
    _:b3  <urn:predicate> _:b1 , _:b9 , _:b10 ;
          <urn:predicate>
                  [
                  ] .
    _:b2  <urn:predicate> _:b1 , _:b5 , _:b8 ;
          <urn:predicate>
                  [
                  ] .
    _:b4  <urn:predicate> _:b1 , _:b5 , _:b6 , _:b7 , _:b8 , _:b9 ,
    _:b10 , _:b2 , _:b3 ;
          <urn:predicate>
                  [
                  ] .
}
```

Canonical string based on Carroll's algorithm:

```
[urn:petersen:graph:a:11-01-2007,
```

```
_:g000001 http://www-uk.hpl.hp.com/people/jjc/rdf/c14n#true "1",
_:g000002 http://www-uk.hpl.hp.com/people/jjc/rdf/c14n#true "10",
_:g000003 http://www-uk.hpl.hp.com/people/jjc/rdf/c14n#true "11",
_:g000003 urn:predicate _:g000004,
_:g000003 urn:predicate _:g000011,
_:g000003 urn:predicate _:g000019,
_:g000003 urn:predicate _:g000020,
_:g000004 http://www-uk.hpl.hp.com/people/jjc/rdf/c14n#true "12",
_:g000004 urn:predicate _:g000003,
_:g000004 urn:predicate _:g000009,
_:g000004 urn:predicate _:g000014,
_:g000004 urn:predicate _:g000016,
_:g000005 http://www-uk.hpl.hp.com/people/jjc/rdf/c14n#true "13",
_:g000005 urn:predicate _:g000013,
_:g000006 http://www-uk.hpl.hp.com/people/jjc/rdf/c14n#true "14",
_:g000007 http://www-uk.hpl.hp.com/people/jjc/rdf/c14n#true "15",
_:g000008 http://www-uk.hpl.hp.com/people/jjc/rdf/c14n#true "16",
_:g000009 http://www-uk.hpl.hp.com/people/jjc/rdf/c14n#true "17",
_:g000010 http://www-uk.hpl.hp.com/people/jjc/rdf/c14n#true "18",
_:g000011 http://www-uk.hpl.hp.com/people/jjc/rdf/c14n#true "19",
_:g000012 http://www-uk.hpl.hp.com/people/jjc/rdf/c14n#true "2",
_:g000012 urn:predicate _:g000001,
_:g000012 urn:predicate _:g000014,
_:g000012 urn:predicate _:g000015,
_:g000012 urn:predicate _:g000016,
_:g000013 http://www-uk.hpl.hp.com/people/jjc/rdf/c14n#true "20",
_:g000014 http://www-uk.hpl.hp.com/people/jjc/rdf/c14n#true "3",
_:g000014 urn:predicate _:g000003,
_:g000014 urn:predicate _:g000004,
_:g000014 urn:predicate _:g000005,
_:g000014 urn:predicate _:g000010,
_:g000014 urn:predicate _:g000012,
_:g000014 urn:predicate _:g000015,
_:g000014 urn:predicate _:g000016,
_:g000014 urn:predicate _:g000018,
_:g000014 urn:predicate _:g000019,
```

```
_:g000014 urn:predicate _:g000020,
_:g000015 http://www-uk.hpl.hp.com/people/jjc/rdf/c14n#true "4",
_:g000015 urn:predicate _:g000005,
_:g000015 urn:predicate _:g000008,
_:g000015 urn:predicate _:g000012,
_:g000015 urn:predicate _:g000018,
_:g000016 http://www-uk.hpl.hp.com/people/jjc/rdf/c14n#true "5",
_:g000016 urn:predicate _:g000004,
_:g000016 urn:predicate _:g000005,
_:g000016 urn:predicate _:g000006,
_:g000016 urn:predicate _:g000012,
_:g000017 http://www-uk.hpl.hp.com/people/jjc/rdf/c14n#true "6",
_:g000018 http://www-uk.hpl.hp.com/people/jjc/rdf/c14n#true "7",
_:g000018 urn:predicate _:g000015,
_:g000018 urn:predicate _:g000017,
_:g000018 urn:predicate _:g000019,
_:g000018 urn:predicate _:g000020,
_:g000019 http://www-uk.hpl.hp.com/people/jjc/rdf/c14n#true "8",
_:g000019 urn:predicate _:g000002,
_:g000019 urn:predicate _:g000003,
_:g000019 urn:predicate _:g000014,
_:g000019 urn:predicate _:g000018,
_:g000020 http://www-uk.hpl.hp.com/people/jjc/rdf/c14n#true "9",
_:g000020 urn:predicate _:g000007]
```

SHA-512 digital digest:

```
NDFmNzhjNTFlYzcwZTAxMjI5Y2QyMTM3OGFjMWViYzU1YjZiMTQxYjA0MjIxM2NhZDB
mMGZlMDU0NzA1Y2UxZTVlM2EzOGU5YmI3YWU1YjU0MzBlZmQ0NjFmNGEyYWU4ZTk2Yz
IzYzQ0ODc1Y2NjNmM2MmQ3ZDk0OTA0ZWEyYTA=
```

TriG serialisation for Figure C.5:

```
<urn:petersen:graph:b:11-01-2007>
{_:b1  <urn:predicate> _:b2 , _:b3 , _:b4 ;
          <urn:predicate>
```

```
                [
                ] .
_:b5  <urn:predicate> _:b6 , _:b7 , _:b8 ;
      <urn:predicate>
                [
                ] .
_:b6  <urn:predicate>
                [
                ] .
_:b2  <urn:predicate>
                [
                ] .
_:b3  <urn:predicate> _:b6 , _:b1 , _:b5 , _:b2 , _:b3 , _:b7 ,
_:b9 ;
      <urn:predicate>
                [
                ] .
_:b7  <urn:predicate> _:b5 , _:b3 , _:b10 ;
      <urn:predicate>
                [
                ] .
_:b8  <urn:predicate> _:b5 , _:b9 , _:b10 ;
      <urn:predicate>
                [
                ] .
_:b9  <urn:predicate> _:b2 , _:b8 , _:b4 ;
      <urn:predicate>
                [
                ] .
_:b10
      <urn:predicate> _:b7 , _:b8 , _:b4 ;
      <urn:predicate>
                [
                ] .
_:b4  <urn:predicate> _:b1 , _:b9 , _:b10 ;
      <urn:predicate>
```

```
                [
                ] .
}
```

Canonical string based on Carroll's algorithm:

```
[urn:petersen:graph:b:11-01-2007,
_:g000001 http://www-uk.hpl.hp.com/people/jjc/rdf/c14n#true "1",
_:g000002 http://www-uk.hpl.hp.com/people/jjc/rdf/c14n#true "10",
_:g000002 urn:predicate _:g000002,
_:g000002 urn:predicate _:g000005,
_:g000002 urn:predicate _:g000011,
_:g000002 urn:predicate _:g000012,
_:g000002 urn:predicate _:g000014,
_:g000002 urn:predicate _:g000016,
_:g000002 urn:predicate _:g000019,
_:g000002 urn:predicate _:g000020,
_:g000003 http://www-uk.hpl.hp.com/people/jjc/rdf/c14n#true "11",
_:g000004 http://www-uk.hpl.hp.com/people/jjc/rdf/c14n#true "12",
_:g000004 urn:predicate _:g000013,
_:g000004 urn:predicate _:g000016,
_:g000004 urn:predicate _:g000017,
_:g000004 urn:predicate _:g000019,
_:g000005 http://www-uk.hpl.hp.com/people/jjc/rdf/c14n#true "13",
_:g000005 urn:predicate _:g000009,
_:g000006 http://www-uk.hpl.hp.com/people/jjc/rdf/c14n#true "14",
_:g000007 http://www-uk.hpl.hp.com/people/jjc/rdf/c14n#true "15",
_:g000008 http://www-uk.hpl.hp.com/people/jjc/rdf/c14n#true "16",
_:g000009 http://www-uk.hpl.hp.com/people/jjc/rdf/c14n#true "17",
_:g000010 http://www-uk.hpl.hp.com/people/jjc/rdf/c14n#true "18",
_:g000011 http://www-uk.hpl.hp.com/people/jjc/rdf/c14n#true "19",
_:g000012 http://www-uk.hpl.hp.com/people/jjc/rdf/c14n#true "2",
_:g000012 urn:predicate _:g000001,
_:g000013 http://www-uk.hpl.hp.com/people/jjc/rdf/c14n#true "20",
_:g000014 http://www-uk.hpl.hp.com/people/jjc/rdf/c14n#true "3",
_:g000014 urn:predicate _:g000002,
```

```
_:g000014 urn:predicate _:g000005,
_:g000014 urn:predicate _:g000008,
_:g000014 urn:predicate _:g000015,
_:g000015 http://www-uk.hpl.hp.com/people/jjc/rdf/c14n#true "4",
_:g000015 urn:predicate _:g000014,
_:g000015 urn:predicate _:g000016,
_:g000015 urn:predicate _:g000017,
_:g000015 urn:predicate _:g000018,
_:g000016 http://www-uk.hpl.hp.com/people/jjc/rdf/c14n#true "5",
_:g000016 urn:predicate _:g000004,
_:g000016 urn:predicate _:g000005,
_:g000016 urn:predicate _:g000006,
_:g000016 urn:predicate _:g000015,
_:g000017 http://www-uk.hpl.hp.com/people/jjc/rdf/c14n#true "6",
_:g000017 urn:predicate _:g000004,
_:g000017 urn:predicate _:g000010,
_:g000017 urn:predicate _:g000015,
_:g000017 urn:predicate _:g000020,
_:g000018 http://www-uk.hpl.hp.com/people/jjc/rdf/c14n#true "7",
_:g000019 http://www-uk.hpl.hp.com/people/jjc/rdf/c14n#true "8",
_:g000019 urn:predicate _:g000004,
_:g000019 urn:predicate _:g000007,
_:g000019 urn:predicate _:g000012,
_:g000019 urn:predicate _:g000020,
_:g000020 http://www-uk.hpl.hp.com/people/jjc/rdf/c14n#true "9",
_:g000020 urn:predicate _:g000002,
_:g000020 urn:predicate _:g000003,
_:g000020 urn:predicate _:g000017,
_:g000020 urn:predicate _:g000019]
```

SHA-512 digital digest:

```
NDFiNGYzY2IzZmIzMjA4MmZkMGZjZDhlZjQxMTljMWE2OTVjY2VjOGUyZDRhMWNjNDk
0ZTJkZjJhY2NiYjk2MTM0YzJiNTIzYzEwNjdkZDliNDZlZTIyYWFlMzdlNDRmOTgyOW
E1MTNkY2I2NmZjM2RkNDI0Y2EzYzkyZmQ5YWY=
```

As can be seen in with the above results, Carroll's algorithm is incapable of reliably relabelling both Petersen graphs in such a way that their canonical strings match. This is further proved by the SHA-512 digital digest.

C.1.3 Nauty

nauty is an application suite that can compare and canonically relabel arbitrary graphs. In this section we will demonstrate how ***nauty*** is able to provide a canonical representation for Figure C.4 and Figure C.5, that could not have otherwise been reliably solved using Carroll's algorithm.

C.1.3.1 Setup

In this example we have again used **dreadnaut** to demonstrate how the ***nauty*** algorithm is capable of reliably canonicalising the Petersen Graph. ***nauty*** input for Figure C.4:

```
> c -a -m       turn getcanon on, group writing off
> n=10 g        enter graph
 0 : 4 9 1;
 1 : 0 8 2;
 2 : 1 7 3;
 3 : 2 6 4;
 4 : 3 5 0;
 5 : 4 6 9;
 6 : 5 3 7;
 7 : 6 2 8;
 8 : 7 1 9;
 9 : 8 0 5;
> x             execute
1 orbit; grpsize=20; 3 gens; 8 nodes; maxlev=3
tctotal=16; canupdates=1; cpu time = 0.00 seconds
```

Canonically relabelled graph:

```
> b             display the canonical labelling
 0 9 1 4 6 7 2 3 8 5
  0 :  1 2 3;
  1 :  0 8 9;
  2 :  0 6 8;
  3 :  0 7 9;
  4 :  5 7 9;
  5 :  4 6 8;
  6 :  2 5 7;
  7 :  3 4 6;
  8 :  1 2 5;
  9 :  1 3 4;
```

nauty input for Figure C.4:

```
> n=10 g        enter graph
 0 : 1 5 4;
 1 : 0 6 2;
 2 : 1 7 3;
 3 : 2 8 4;
 4 : 3 9 0;
 5 : 0 9 6;
 6 : 5 1 7;
 7 : 6 2 8;
 8 : 7 3 9;
 9 : 8 4 5;
> x             execute
1 orbit; grpsize=20; 3 gens; 8 nodes; maxlev=3
tctotal=16; canupdates=1; cpu time = 0.00 seconds
```

Canonically relabelled graph:

```
>b              display the canonical labelling
0 5 1 4 8 7 2 3 6 9
  0 :  1 2 3;
  1 :  0 8 9;
```

```
 2 :  0 6 8;
 3 :  0 7 9;
 4 :  5 7 9;
 5 :  4 6 8;
 6 :  2 5 7;
 7 :  3 4 6;
 8 :  1 2 5;
 9 :  1 3 4;
```

As can be seen, ***nauty*** is capable of reliable canonical relabelling of two representations of the Petersen graph. We can perform further analysis by using **dreadnaut** to compare Figure C.4 and Figure C.5 directly for us an state whether two input graphs are identical:

```
Dreadnaut version 2.4 (32 bits).
> c -a -m       turn getcanon on, group writing off
> n=10 g        enter first graph
 0 : 4 9 1;
 1 : 0 8 2;
 2 : 1 7 3;
 3 : 2 6 4;
 4 : 3 5 0;
 5 : 4 6 9;
 6 : 5 3 7;
 7 : 6 2 8;
 8 : 7 1 9;
 9 : 8 0 5;
> x @           execute, save the result
1 orbit; grpsize=20; 3 gens; 8 nodes; maxlev=3
tctotal=16; canupdates=1; cpu time = 0.00 seconds
> g             enter second graph
 0 : 1 5 4;
 1 : 0 6 2;
 2 : 1 7 3;
 3 : 2 8 4;
 4 : 3 9 0;
```

```
 5 : 0 9 6;
 6 : 5 1 7;
 7 : 6 2 8;
 8 : 7 3 9;
 9 : 8 4 5;
> x             execute
1 orbit; grpsize=20; 3 gens; 8 nodes; maxlev=3
tctotal=16; canupdates=1; cpu time = 0.00 seconds
> ##            compare to saved graph
h and h' are identical.
 0-0 1-1 2-2 3-3 4-4 5-9 6-8 7-7 8-6 9-5
```

nauty not only reveals correctly that Figure C.4 and Figure C.5 are identical, but it can also show the mapping between differing vertices in each graph.

Appendix D

D.1 Federation Scenario Inference Rules

Here we include the inference rules used in our federation scenarios described in Section 4.7. Rules are specified using the Jena 2 rule language as Horn clauses. Since Horn clauses in Jena 2 rules cannot represent logical *ORs*, additional rules have been written to cover for alternative cases.

D.1.1 FLOSS Signature Recovery

If a digital signature fails most systems will typically report the error without chance for recovery. Here we try to recover from a signature failure by federating data from external sources. For FLOSS, we will check that the signer is a committer on the project (check project DOAP) and user's personal FOAF; we also confirm whether they have successfully committed before. Once these criteria have been satisfied, a report is produced advising the administrator about what to do next.

We defined the FLOSS recovery process (see Section 4.7) as follows:

1. Determine the author of commit with the broken signature;
2. Check if author is listed a committer to project (DOAP);
3. Check if author has made other commits for same document;
4. Search for other commits with same author and check status;
5. Generate report of author for repository admin.

While the above steps have a particular order, it is not possible with the Jena rule engine to prioritise which rule is fired first. More advanced RETE implementations such as Jess, include a feature where the *salience* can be specified. This means it is possible to state that some rules will have priorities over others if the necessary data to fire them exists.

```
[checkDOAP:
        (?doc rdf:type dp:Document)
        (?doc dp:revision "+version+")
        (?doc swp:inGraph ?graph)
        (?graph swp:assertedBy ?warrant)
        (?warrant swp:authority ?authority)
        (?authority swp:X509Certificate ?cert)
        doapAuthorKnown(false)
        -> (dp:Report"+date+" rdf:type dp:ValidityReport)
           (dp:Report"+date+" dp:target ?doc)
           (dp:Report"+date+" foaf:maker "+admin+")
           (?doc swp:authority ?authority)
           (dp:Report"+date+" swp:authority dp:known)
]

[checkDOAPFail:
        (?doc rdf:type dp:Document)
        (?doc dp:revision "+version+")
        (?doc swp:inGraph ?graph)
        (?graph swp:assertedBy ?warrant)
        (?warrant swp:authority ?authority)
        (?authority swp:X509Certificate ?cert)
        doapAuthorKnown(false)
        -> (dp:Report"+date+" rdf:type dp:ValidityReport)
           (dp:Report"+date+" dp:target ?doc)
           (dp:Report"+date+" foaf:maker "+admin+")
           (dp:Report"+date+" swp:authority dp:unknown)
]
```

FIGURE D.1: FLOSS DOAP inference

Figure D.1 describes the FLOSS DOAP inference rules. Two sets of rules are defined so that two possibilities are catered for: the author is either known or not known in the remote DOAP file[1]. We firstly check the certificate of the document that failed and compare it with the certificate stored in the DOAP description (Step 1) using the `doapAuthorKnown` builtin[2]. Once this check has been

[1] http://www.ecs.soton.ac.uk/ erw/MyProject/doap.rdf

[2] `doapAuthorKnown` take a boolean as input, depending on the scenario, e.g., false for FLOSS, true for IST.

made, the the inference engine begins building up a report for the administrator, stating the affected document, the creator of the report and whether the author of the broken signature is known or not.

It is important to note that the order in which a rule is declared is indepedent of the order of execution. In the case of the RETE algorithm, each time new information is added to the knowledge-base, the rule engine will fire the appropriate rule.

```
[checkLocalRepositoryAuthor:
        (?report dp:target ?doc)
        (?doc dp:hasClass ?class)
        (?doc dp:maker ?maker)
        (?doc swp:authority ?authority)
        -> (?report dp:localAuthorStatus dp:valid)
           (?authority dp:localCommitStatus dp:valid)
]

[checkLocalRepositoryAuthorFail:
        (?report dp:target ?doc)
        (?doc dp:hasClass ?class)
        (?doc dp:maker ?maker)
        (?doc swp:authority ?authority)
        (?report swp:authority dp:unknown)
        -> (?report dp:localAuthorStatus dp:invalid)
           (?authority dp:localCommitStatus dp:invalid)
]
```

FIGURE D.2: FLOSS check author inference

Once the DOAP description of a FLOSS project has been checked, we verify whether they have committed the same document before (Step 2) as shown in Figure D.2. This is a simple matching problem against the local repository.

```
[listCommits:
        (?report rdf:type dp:ValidityReport)
        (?authority dp:localCommitStatus dp:valid)
        (?doc dp:maker ?authority)
        (?report dp:target ?target)
        -> (?report dp:knownCommit ?doc)
]
```

FIGURE D.3: FLOSS list local author commits

Finally, we list other documents that the author has committed and check their status. For this rule to fire, the author must have already been identified as an author. The absence of commits in Figure D.3 simply means that the committer has not made any commits.

D.1.1.1 FLOSS Recovery Report

Figure D.4 shows an example validity report, serialised in RDF/XML-ABBREV.

```
<rdf:RDF
    xmlns:j.0="http://xmlns.com/foaf/0.1/"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:j.1="http://www.w3.org/2004/03/trix/swp-2/"
    xmlns:j.2="http://grid.cx/dp/1.0/">
  <rdf:Description rdf:about="https://localhost:8443/webdav/
  taverna/taverna-workbench/org/embl/ebi/escience/scuflui/
  workbench/Workbench/1/2/Workbench.java">
    <j.1:authority>
      <rdf:Description
      rdf:about="mailto:erw@it-innovation.soton.ac.uk">
        <j.2:localCommitStatus
        rdf:resource="http://grid.cx/dp/1.0/valid"/>
      </rdf:Description>
    </j.1:authority>
  </rdf:Description>
  <rdf:Description>
    <j.0:maker
    rdf:resource="mailto:erw@it-innovation.soton.ac.uk"/>
  </rdf:Description>
  <j.2:ValidityReport
  rdf:about="http://grid.cx/dp/1.0/Report2007-04-01T12:34:34.734">
    <j.2:recommendation
    rdf:resource="http://grid.cx/dp/1.0/signatureOverride"/>
    <j.2:knownCommit rdf:resource="https://localhost:8443/webdav/
    taverna/taverna-workbench/org/embl/ebi/escience/scuflui/
    workbench/Workbench/1/2/Workbench.java"/>
    <j.2:knownCommit rdf:resource="https://localhost:8443/webdav/
    taverna/taverna-workbench/org/embl/ebi/escience/scuflui/
    workbench/Workbench/1/1/Workbench.java"/>
    <j.2:localAuthorStatus
    rdf:resource="http://grid.cx/dp/1.0/valid"/>
    <j.2:target rdf:resource="https://localhost:8443/webdav/
    taverna/taverna-workbench/org/embl/ebi/escience/scuflui/
    workbench/Workbench/1/2/Workbench.java"/>
  </j.2:ValidityReport>
  <rdf:Description
  rdf:about="http://www.ecs.soton.ac.uk/~erw/MyProject/doap.rdf">
</rdf:RDF>
```

FIGURE D.4: Example FLOSS Recovery Report

D.1.2 EC IST Signature Recovery

In the IST federation scenario we can use additional information that was not available in the FLOSS scenario, notably project workpackage information, certification authorities and contributing partner remote repositories. Like in the FLOSS recovery scenario, DOAP and FOAF information will be used from the project website to verify the committer's membership to the project which is cross-checked with the certificate in the broken signature.

We defined the IST recovery process (see Section 4.7) as follows:

1. Determine the author of commit with the broken signature;
2. Check if author's certificate is signed by a CA known to the project (Certificate);
3. Check if author is listed as working in the workpackage the document is part of (FOAF, DOAP);
4. Check if author has committed in local repository;
5. Request metadata about any commits in Contributing Partners' repository;
6. Generate report of author for repository admin;
7. Provide override option (new Digital Signature).

In a similar fashion to Figure D.1, Figure D.5 accesses the DOAP description of our AcmeGrid project and verifies the identification of the author. However, `doapAuthorKnown` does several additional steps. Firstly, it verifies the corresponding certificate of the broken signature against the known CAs of the project (Step 2). Secondly, it also checks that the workpackage of the author matches that listed in the DOAP description (Step 3). Once these have been checked, a new report is generated.

Figure D.6 performs a local and remote check that determines three things: firstly, whether the author of the broken signature has committed the same document before locally (Step 4). Secondly, the builtin `remoteAuthorUnknown` performs the same check at a contributing partner's repository. Thirdly, for all documents committed in a remote repository, we return the document descriptions (Step 5). This means that the administrator can determine whether the broken signature is an error or a malicious modification.

```
[checkDOAP:
        (?doc rdf:type dp:Document)
        (?doc dp:revision "+version+")
        (?doc swp:inGraph ?graph)
        (?graph swp:assertedBy ?warrant)
        (?warrant swp:authority ?authority)
        (?authority swp:X509Certificate ?cert)
        doapAuthorKnown(true "+wp+")
        -> (dp:Report"+date+" rdf:type dp:ValidityReport)
           (dp:Report"+date+" dp:target ?doc)
           (dp:Report"+date+" foaf:maker ?authority)
           (?doc swp:authority ?authority)
           (?authority dp:wpCheck dp:valid)
           (?authority dp:caCheck dp:valid)
]

[checkDOAPfail:
        (?doc rdf:type dp:Document)
        (?doc dp:revision "+version+")
        (?doc swp:inGraph ?graph)
        (?graph swp:assertedBy ?warrant)
        (?warrant swp:authority ?authority)
        (?authority swp:X509Certificate ?cert)
        doapAuthorUnknown(true "+wp+")
        -> (dp:Report"+date+" rdf:type dp:ValidityReport)
           (dp:Report"+date+" dp:target ?doc)
           (?report foaf:maker ?authority)
           (?doc swp:authority ?authority)
           (?authority dp:wpCheck dp:invalid)
           (?authority dp:caCheck dp:invalid)
]
```

FIGURE D.5: IST DOAP inferences

As Figure D.7 shows, once these checks have been made, the rule engine will fire another rule that provides a simple recommendation as to what they should do next (Step 6).

D.1.2.1 IST Recovery Report

Figure D.8 shows an example validity report for the IST recovery scenario.

```
[checkRemoteRepositoryAuthor:
        (?report dp:target ?doc)
        (?doc dp:revision "+version")
        (?doc dp:hasClass ?class)
        (?doc dp:maker ?maker)
        (?doc swp:authority ?authority)
        (?report dp:target ?target)
        remoteAuthorKnown(?maker ?class)
        listCommits(?report ?authority "+version+")
        -> (?report dp:remoteAuthorStatus dp:valid)
           (?authority dp:remoteCommitStatus dp:valid)
]

[checkRemoteRepositoryAuthorFail:
        (?report dp:target ?doc)
        (?doc dp:revision "+version")
        (?doc dp:hasClass ?class)
        (?doc dp:maker ?maker)
        (?doc swp:authority ?authority)
        (?report dp:target ?target)
        remoteAuthorUnknown(?maker ?class)
        -> (?report dp:remoteAuthorStatus dp:invalid)
           (?authority dp:remoteCommitStatus dp:invalid)
]
```

FIGURE D.6: IST check author inferences

```
[makeRecommendation:
        (?report rdf:type dp:ValidityReport)
        (?report dp:remoteAuthorStatus dp:valid)
        -> (?report dp:recommendation dp:signatureOverride)
]

[makeRecommendationFail:
        (?report rdf:type dp:ValidityReport)
        (?report dp:remoteAuthorStatus dp:invalid)
        -> (?report dp:recommendation dp:manualFix)
]
```

FIGURE D.7: IST recommendation inferences

```
<rdf:RDF
    xmlns:j.0="http://xmlns.com/foaf/0.1/"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:j.1="http://www.w3.org/2004/03/trix/swp-2/"
    xmlns:j.2="http://grid.cx/dp/1.0/">
  <rdf:Description>
    <j.0:maker>
      <rdf:Description
      rdf:about="mailto:erw@it-innovation.soton.ac.uk">
        <j.2:remoteCommitStatus
        rdf:resource="http://grid.cx/dp/1.0/valid"/>
        <j.2:caCheck
        rdf:resource="http://grid.cx/dp/1.0/valid"/>
        <j.2:wpCheck
        rdf:resource="http://grid.cx/dp/1.0/valid"/>
      </rdf:Description>
    </j.0:maker>
  </rdf:Description>
  <rdf:Description rdf:about="https://localhost:8443/webdav/
  taverna/taverna-workbench/org/embl/ebi/escience/scuflui/
  workbench/Workbench/1/7/Workbench.java">
    <j.1:authority
    rdf:resource="mailto:erw@it-innovation.soton.ac.uk"/>
  </rdf:Description>
  <j.2:ValidityReport
  rdf:about="http://grid.cx/dp/1.0/Report2007-04-01T12:25:09.125">
    <j.2:knownCommit rdf:resource="https://localhost:8443/webdav/
    taverna/taverna-workbench/org/embl/ebi/escience/scuflui/
    workbench/Workbench/1/3/Workbench.java"/>
    <j.2:knownCommit rdf:resource="https://localhost:8443/webdav/
    taverna/taverna-workbench/org/embl/ebi/escience/scuflui/
    workbench/Workbench/1/1/Workbench.java"/>
    <j.2:knownCommit rdf:resource="https://localhost:8443/webdav/
    taverna/taverna-workbench/org/embl/ebi/escience/scuflui/
    workbench/Workbench/1/6/Workbench.java"/>
    <j.2:recommendation
    rdf:resource="http://grid.cx/dp/1.0/signatureOverride"/>
    <j.2:knownCommit rdf:resource="https://localhost:8443/webdav/
    taverna/taverna-workbench/org/embl/ebi/escience/scuflui/
    workbench/Workbench/1/5/Workbench.java"/>
    <j.2:target rdf:resource="https://localhost:8443/webdav/
    taverna/taverna-workbench/org/embl/ebi/escience/scuflui/
    workbench/Workbench/1/7/Workbench.java"/>
    <j.2:remoteAuthorStatus
    rdf:resource="http://grid.cx/dp/1.0/valid"/>
    </j.2:ValidityReport>
  <rdf:Description
  rdf:about="http://www.ecs.soton.ac.uk/~erw/AcmeGrid/doap.rdf">
  </rdf:Description>
</rdf:RDF>
```

FIGURE D.8: Example IST Recovery Report

Bibliography

Philippe Aigrain, Roberto Andradas, Raphaël Badin, Renaud Bernard, Luis Ca nas Daz, Paul David, Santiago Due nas, Theo Dunnewijk, Rishab Aiyer Ghosh, Ruediger Glott, Jesus Gonzalez-Barahona, Kirsten Haaland, Bronwyn Hall, Wendy Hansen, Juan Jose Amor, Huub Meijers, Alvaro Navarro, Francesco Rentocchini, Gregorio Robles, Barbara Russo, Giancarlo Succi, and Adriaan van Zon. Study on the: Economic impact of open source software on innovation and the competitiveness of the Information and Communication Technologies (ICT) sector in the EU. Technical report, UNU-MERIT, URJC, SOPINSPACE, BICST, November 2006. Available at http://ec.europa.eu/enterprise/ict/policy/doc/2006-11-20-flossimpact.pdf, accessed on 20.01.2007.

Steve Anderson, Jeff Bohren, Toufic Boubez, Marc Chanliau, Giovanni Della-Libera, Brendan Dixon, Praerit Garg, Phillip Hallam-Baker, Maryann Hondo, Chris Kaler, Hal Lockhart, Robin Martherus, Hiroshi Maruyama, Nataraj Nagaratnam, Andrew Nash, Rob Philpott, Darren Platt, Hemma Prafullchandra, Maneesh Sahu, John Shewchuk, Dan Simon, Davanum Srinivas, Elliot Waingold, David Waite, Doug Walter, and Riaz Zolfonoon. Web Services Trust Language (WS-Trust). IBM, February 2005. Available at ftp://www6.software.ibm.com/software/developer/library/ws-trust.pdf, accessed on 13.01.2007.

Mario Antonioletti, Malcolm Atkinson, Rob Baxter, Andrew Borley, Neil P. Chue Hong, Brian Collins, Neil Hardman, Alastair C. Hume, Alan Knox, Mike Jackson, Amy Krause, Simon Laws, James Magowan, Norman W. Paton, Dave Pearson, Tom Sugden, Paul Watson, and Martin Westhead. The design and implementation of grid database services in OGSA-DAI. Concurrency and Computation: Practice and Experience, 17(2-4):357–376, February 2005.

Mario Antonioletti, Neil Chue Hong, Ally Hume, Mike Jackson, Amy Krause, Jeremy Nowell, Charaka Palansuriya, Tom Sugden, and Martin Westhead. Experiences of Designing and Implementing Grid Database Services in the OGSA-DAI Project. In Global Grid Forum Workshop on Designing and

Building Grid Services, Chicago, Illinois, USA, October 2003. Available at, http://www-unix.mcs.anl.gov/~keahey/DBGS/DBGS_files/dbgs_papers/hong.pdf, accessed on 13.02.2007.

Alessandro Artale and Enrico Franconi. Introducing Temporal Description Logics. In TIME '99: Proceedings of the 6th International Workshop on Temporal Representation and Reasoning, page 2, Washington, DC, USA, 1999. IEEE Computer Society. ISBN 0-7695-0173-7.

Alessandro Artale and Enrico Franconi. Handbook of Time and Temporal Reasoning in Artificial Iintelligence. MIT Press, Cambridge, MA, USA, 2000.

Malcolm Atkinson, David DeRoure, Alistair Dunlop, Geoffrey Fox, Peter Henderson, Tony Hey, Norman Paton, Steven Newhouse, Savas Parastatidis, Anne Trefethen, Paul Watson, and Jim Webber. Web Service Grids: An Evolutionary Approach. Concurrency and Computation: Practice and Experience, 17(2-4): 377–389, February 2005.

F. Baader and W. Nutt. In F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, and P.F. Patel-Schneider, editors, Description Logic Handbook, chapter Basic Description Logics, pages 47–100. Cambridge University Press, Cambridge, UK, 2002.

F. Baader and U. Sattler. An Overview of Tableau Algorithms for Description Logics. Studia Logica, 69:5–40, 2001.

Franz Baader. Restricted Role-value-maps in a Description Logic with Existential Restrictions and Terminological Cycles. In Description Logics, 2003. Available at http://SunSITE.Informatik.RWTH-Aachen.de/Publications/CEUR-WS/Vol-81/baader.pdf, accessed on 12.12.2006.

Franz Baader, Diego Calvanese, Deborah Mcguinness, Daniele Nardi, and Peter Patel-Schneider. The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press, 2003. ISBN 0521781760.

Siddharth Bajaj, Giovanni Della-Libera, Brendan Dixon, Mike Dusche, Maryann Hondo, Matt Hur, Hal Lockhart, Hiroshi Maruyama, Nataraj Nagaratnam, Andrew Nash, Hemma Prafullchandra, and John Shewchuk. Web Services Federation Language (WS-Federation). IBM, July 2003. Available at ftp://www6.software.ibm.com/software/developer/library/ws-fed.pdf, accessed on 12.01.2007.

Mark A. Baker and Richard Boakes. A Framework for Unified Information Browsing. In Semantic Web Applications and Perspectives (SWAP), 1st Italian Semantic Web Workshop, Ancona, Italy, December 2004. Available at http://rdfx.org/docs/presentation/boakes.pdf, accessed on 13.11.2006.

Dibyendu Baksi. J2EE Transaction Frameworks: Distributed Transaction Primer. Website, May 2001. Available at http://www.onjava.com/lpt/a/852, accessed on 23.02.3007.

Ed Barkmeyer. Rules on the Web: Why? In Rules on the Web Workshop, WWW2006, May 2006. Available at http://www.aifb.uni-karlsruhe.de/WBS/phi/RoW06/talks/barkmeyer.ppt, accessed on 05.03.2007.

Jesús Barrasa, Óscar Corcho, and Asunción Gómez-Pérez. R2O, an Extensible and Semantically Based Database-to-Ontology Mapping Language. In Second Workshop on Semantic Web and Databases (SWDB2004), Toronto, Canada, August 2004. Available at http://www.cs.man.ac.uk/~ocorcho/documents/SWDB2004_BarrasaEtAl.pdf, accessed on 28.11.2006.

Mark Bartel, John Boyer, Barb Fox, Brian LaMacchia, and Ed Simon. XML-Encryption – W3C Recommendation. The Internet Society & W3C® (MIT, INRIA, Keio), February 2002a. Available at http://www.w3.org/TR/xmlenc-core/, accessed on 04.12.2006.

Mark Bartel, John Boyer, Barb Fox, Brian LaMacchia, and Ed Simon. XML-Signature Syntax and Processing – W3C Recommendation. The Internet Society & W3C® (MIT, INRIA, Keio), February 2002b. Available at http://www.w3.org/TR/xmldsig-core/, accessed on 13.12.2006.

Sean Bechhofer, Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein. OWL Web Ontology Language Reference – W3C Recommendation. W3C® (MIT, ERCIM, Keio), February 2004. Available at http://www.w3.org/TR/owl-ref/, accessed on 24.11.2006.

Dave Beckett, editor. RDF/XML Syntax Specification (Revised). W3C® (MIT, ERCIM, Keio), February 2004. Available at http://www.w3.org/TR/rdf-syntax-grammar/, accessed on 12.10.2006.

T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. Scientific American, pages 28–37, May 2001.

Tim Berners-Lee. Information Management: A Proposal. Technical report, Organisation Européenne pour la Recherche Nucléaire (CERN), March 1989. Available at http://www.w3.org/History/1989/proposal.html, accessed on 13.12.2006.

Tim Berners-Lee. Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web. Collins, November 2000. ISBN 006251587X.

Tim Berners-Lee. Semanic web design issues: Rules, 2001. Available at http://www.w3.org/DesignIssues/Logic.html, accessed on 13.12.2006.

Tim Berners-Lee. Web for real people. Website, 2005. Available at http://www.w3.org/2005/Talks/0511-keynote-tbl/, accessed on 12.11.2006.

Paul V. Biron and Ashok Malhotra. XML Schema Part 2: Datatypes Second Edition – W3C Recommendation. W3C® (MIT, ERCIM, Keio), October 2004. Available at http://www.w3.org/TR/xmlschema-2/, accessed on 07.11.2006.

Chris Bizer. TriQL - A Query Language Named Graphs, April 2004a. Available at http://www.wiwiss.fu-berlin.de/suhl/bizer/TriQL/, accessed on 15.01.2007.

Chris Bizer. TriQL.P - A Query Language for Querying Named Graphs Published by Untrustworthy Sources, September 2004b. Available at http://www.wiwiss.fu-berlin.de/suhl/bizer/TriQLP/, accessed on 15.01.2007.

Chris Bizer. The TriG Syntax. Website, June 2005. Available at http://www.wiwiss.fu-berlin.de/suhl/bizer/TriG/Spec/, accessed on 15.01.2007.

Chris Bizer, Richard Cyganiak, and Rowland Watkins. Named Graphs for Jena (NG4J) API. The 2nd European Semantic Web Conference, 2005a.

Christian Bizer and Richard Cyganiak. D2R Server - Publishing Relational Databases on the Semantic Web. In Poster at the 5th International Semantic Web Conference, Athens, USA, November 2006. Available at, http://sites.wiwiss.fu-berlin.de/suhl/bizer/d2r-server/resources/d2r-Server-poster-iswc2006.pdf, accessed on 15.01.2007.

Christian Bizer, Richard Cyganiak, Tobias Gauss, and Oliver Maresch. The TriQL.P Browser: Filtering Information using Context-, Content- and Rating-Based Trust Policies. In Semantic Web and Policy Workshop at the 4th International Semantic Web Conference. Springer-Verlag New York, Inc.,

November 2005b. Available at http://sites.wiwiss.fu-berlin.de/suhl/bizer/pub/Bizer-TriQLP-Browser-SWPW.pdf, accessed on 03.03.2007.

Christian Bizer and Radoslaw Oldakowski. Using Context- and Content-Based Trust Policies on the Semantic Web. In 13th World Wide Web Conference, WWW 2004 (poster), May 2004. Available at http://www.wiwiss.fu-berlin.de/suhl/bizer/SWTSGuide/p747-bizer.pdf, accessed on 15.01.2007.

Christian Bizer and Andy Seaborne. D2RQ - Treating Non-RDF Databases as Virtual RDF Graphs, November 2004. Available at http://www.wiwiss.fu-berlin.de/suhl/bizer/pub/Bizer-D2RQ-ISWC2004.pdf, accessed on 15.01.2007.

Don Box, Erik Christensen, Francisco Curbera, Donald Ferguson, Jeffrey Frey, Marc Hadley, Chris Kaler, David Langworthy, Frank Leymann, Brad Lovering, Steve Lucco, Steve Millet, Nirmal Mukhi, Mark Nottingham, David Orchard, John Shewchuk, Eugène Sindambiwe, Tony Storey, Sanjiva Weerawarana, and Steve Winkler. Web Services Addressing (WS-Addressing) – W3C Member Submission. August 2004. Available at http://www.w3.org/Submission/ws-addressing/, accessed on 07.02.2007.

Uri Braun and Avi Shinnar. A Security Model for Provenance. Technical report, Harvard University Computer Science Technical Report TR-04-06, 2006.

Tim Bray, Dave Hollander, and Andrew Layman, editors. Namespaces in XML. W3C® (MIT, INRIA, Keio), January 1999. Available at http://www.w3.org/TR/REC-xml-names/, accessed on 05.11.2006.

Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler (Second Edition), and François Yergeau (Third Edition), editors. Extensible Markup Language (XML) 1.0 – W3C Recommendation. W3C® (MIT, ERCIM, Keio), third edition, February 2004. Available at http://www.w3.org/TR/REC-xml/, accessed on 05.11.2006.

Paulo Tibério Bulhoes, Chansup Byun, Rick Castrapel, and Omar Hassaine. N1™ Grid Engine 6 Features and Capabilities. In Sun Users Performace Group (SUPerG), Phoenix, Arizona, USA, May 2004. Available at http://www.sun.com/products-n-solutions/edu/whitepapers/pdf/N1GridEngine6.pdf, accessed on 05.12.2006.

Peter Buneman, Sanjeev Khanna, and Wang-Chiew Tan. Data provenance: Some basic issues. Foundations of Software Technology and Theoretical Computer Science, 2000.

Peter Buneman, Sanjeev Khanna, and Wang-Chiew Tan. Data Provenance. May 2001a. Available at http://db.cis.upenn.edu/Research/provenance.html, accessed on 09.09.2006.

Peter Buneman, Sanjeev Khanna, and Wang-Chiew Tan. Why and Where: A Characterisation of Data Provenance. International Conference on Database Theory (ICDT), 2001b.

Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, and Michael Stal. Pattern-Oriented Software Architecture. John Wiley and Sons Ltd, July 1996. ISBN 0-471-95869-7.

Dr Mark H. Butler. Is the Semantic Web Just Hype? March 2005. Available at http://www.hpl.hp.com/personal/marbut/isTheSemanticWebHype.pdf, accessed on 05.11.2006.

D. Byrne, M. Chue Hong, A. Hume, and M. Jackson. e-Science, the Grid and Microsoft .NET. In GlobusWorld 2004, San Francisco, USA, January 2004.

D. Byrne, A. Hume, and M. Jackson. Grid Services and Microsoft .NET. In Proceedings of UK e-Science All Hands Meeting (ed. Cox, S), pages 129–136, Nottingham, UK, September 2003. ISBN 1-904425-11-9.

Diego Calvanese and Giuseppe De Giacomo. Expressive Description Logics. pages 178–218. Cambridge University Press, 2003.

S. Cantor, F. Hirsch, J Kemp, R. Philpott, and E. Maler. Bindings and Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0. OASIS, 2005.

Nicholas Carr. Nature's Flawed Study of Wikipedia's Quality. Website, February 2006. Available at http://www.roughtype.com/archives/2006/02/community_and_h.php, accessed on 04.04.007.

J. J. Carroll, I. Dickinson, C. Dollin, D. Reynolds, A. Seaborne, and K. Wilkinson. Jena: Implementing the Semantic Web Recommendations. In 13th World Wide Web Conference, WWW2004, 2004.

Jeremy Carroll. Signing RDF Graphs. In 2nd International Semantic Web Conference (ISWC2003), volume 2870. Springer-Verlag LNCS, July 2003. Available at http://www.hpl.hp.com/techreports/2003/HPL-2003-142.pdf, accessed on 05.11.2006.

Jeremy J. Carroll, Christian Bizer, Pat Hayes, and Patrick Stickler. Named Graphs, provenance and trust. In 14th International World Wide Web Conference, Chiba, Japan, May 2005.

Donald D. Chamberlin and Raymond F. Boyce. SEQUEL: A structured English query language. In FIDET '74: Proceedings of the 1974 ACM SIGFIDET (now SIGMOD) workshop on Data description, access and control, pages 249–264, New York, NY, USA, 1974. ACM Press.

Merrill R. Chapman. In Search of Stupidity : Over 20 Years of High-Tech Marketing Disasters. Springer-Verlag, New York, Inc., 2003. ISBN 1-59059-104-6.

L. Chen, N. Shadbolt, C. Goble, and F. Tao. Managing Semantic Metadata for Web Grid Services. International Journal of Web Services Research, 2006a.

L. Chen, F. Tao, and N. Shadbolt. A Semantic Web Service Based Approach for Augmented Provenance. In Proceedings of IEEE/WIC/ACM Web Intelligent 2006, pages 594–600. IEEE Computer Society, December 2006b.

Liming Chen, Zhuoan Jiao, and Simon J. Cox. On the use of semantic annotations for supporting provenance in grids. In Proceedings of Euro-Par 2006 Parallel Processing, pages 371–380, Dresden, Germany, August-September 2006c. Lecture Notes in Computer Science.

Liming Chen, Victor Tan, Fenglian Xu, Alexis Biller, Paul Groth, Simon Miles, John Ibbotson, Michael Luck, and Luc Moreau. A proof of concept: Provenance in a Service Oriented Architecture. In Proceedings of the 4th All Hands Meeting (AHM), Nottingham, UK, September 2005. EPSRC. Available at http://eprints.ecs.soton.ac.uk/10796/01/strawman.pdf, accessed on 04.12.2006.

Kevin Cline, Josh Cohen, Doug Davis, Donald F. Ferguson, Heather Kreger, Raymond McCollum, Bryan Murray, Ian Robinson, Jeffrey Schlimmer, John Shewchuk, Vijay Tewari, and William Vambenepe. Toward Converging Web Service Standards for Resources, Events, and Management: A Joint White

Paper from Hewlett Packard Corporation, IBM Corporation, Intel Corporation and Microsoft Corporation. IBM, 2006.

Russell Cloran and Barry Irwin. XML Digital Signature and RDF. In Information Society South Africa (ISSA 2005), July 2005. Available at http://russell.rucus.net/masters/writings/conferences/issa2005cloran-poster.pdf, accessed on 04.04.2007.

C. C. Cocks. A Note on Non-Secret Encryption. Technical report, CESG Report, 1973.

E. F. Codd. A Relational Model of Data for Large Shared Data Banks. Communications of the ACM, 13(6):377387, June 1970. Available at http://www.acm.org/classics/nov95/toc.html, accessed on 15.03.2007.

E. F. Codd. The Relational Model for Database Management, Version 2. Addison Wesley Publishing Company, 1990. ISBN 0201141922.

Ben Collins-Sussman. Dispelling Subversion FUD, July 2004. Available at http://www.red-bean.com/sussman/svn-anti-fud.html, accessed on 13.12.2006.

Ben Collins-Sussman, Brian W. Fitzpatrick, and C. Michael Pilato. Version Control with Subversion. O'Reilly Media, first edition, June 2004. ISBN 0-596-00448-6.

Alain Colmerauer and Philippe Roussel. The birth of Prolog. In The second ACM SIGPLAN conference on History of programming languages, pages 37–52, November 1992. Available at http://www.lim.univ-mrs.fr/~colmer/ArchivesPublications/HistoireProlog/19november92.pdf, accessed on 25.11.2006.

Michael A. Covington, Donald Nute, and Andre Vellino. Prolog Programming in Depth. Prentice Hall, 1996. ISBN 0-13-138645-X.

John Cowan and Richard Tobin, editors. XML Information Set. W3C® (MIT, INRIA, Keio), 2nd edition, February 2004. Available at http://www.w3.org/TR/xml-infoset/, accessed on 03.12.2006.

Richard Cyganiak. Guus Schreiber on Semantic Web best practices. October 2004. Available at http://dowhatimean.net/2004/10/guus-schreiber-on-semantic-web-best-practices, accessed on 04.02.2007.

K. Czajkowski, D. Ferguson, I. Foster, J. Frey, S. Graham, T. Maguire, D. Snelling, and S. Tuecke. From Open Grid Services Infrastructure to WS-Resource Framework: Refactoring & Evolution. March 2004a.

K. Czajkowski, D. F. Ferguson, I. Foster, J. Frey, S. Graham, I. Sedukhin, D. Snelling, S. Tuecke, and W. Vambenepe. The WS-Resource Framework. March 2004b. Available at http://www.globus.org/wsrf/specs/ws-wsrf.pdf, accessed on 15.02.2007.

Paulo Pinheiro da Silva, Deborah L. McGuinness, and Rob McCool. Knowledge provenance infrastructure. 26(4):26–32, December 2003.

C. J. Date. The Database Relational Model: A Retrospective Review and Analysis: A Historical Account and Assessment of E. F. Codd's Contribution to the Field of Database Technology. Addison Wesley Longman, 2000. ISBN 0201612941.

C. J. Date. And now for something completely computational. Website, July 2006. Available at http://www.dcs.warwick.ac.uk/~hugh/TTM/comput.pdf, accessed 20.03.2007.

Matthew Davis. Congress 'made Wikipedia changes'. February 2006. Available at http://news.bbc.co.uk/2/hi/technology/4695376.stm, accessed on 20.01.2007.

Chris DiBona, Sam Ockman, and Mark Stone, editors. Open Sources: Voices from the Open Source Revolution. O'Reilly & Associates, 1st edition edition, January 1999. ISBN 1-56592-582-3.

Chris DiBona, Mark Stone, and Danese Cooper. Open Sources 2.0: The Continuing Evolution. O'Reilly & Associates, first edition edition, October 2005. ISBN 0-596-00802-3.

T. Dierks and C. Allen. The TLS Protocol Version 1.0, January 1999. Available at http://www.faqs.org/rfcs/rfc2246.html, accessed on 08.01.2007.

W. Diffie and M. E. Hellman. New directions in cryptography. IEEE Transactions on Information Theory, 1976.

W. Diffie and M. E. Hellman. The First Ten Years of Public-Key Cryptography. In Proceedings of the IEEE, volume 76, pages 560–577. IEEE, 1988.

Theo Dimitrakos, Brian Matthews, and Juan Bicarregui. Towards security and trust management policies on the Web. In ERCIM Workshop

'The Role of Trust in e-Business' in conjunction with IFIP I3E Conference, CLRC Rutherford Appleton Laboratory, Oxfordshire, October 2001. Available at http://www.elec.qmul.ac.uk/staffinfo/stefan/fipa-security/rfi-responses/ral-SecurityPoliciesWeb.pdf, accessed on 13.02.2007.

Li Ding, Tim Finin, Yun Peng, Anupam Joshi, Paulo Pinheiro da Silva, and Deborah L. McGuinness. Tracking RDF Graph Provenance using RDF Molecules. In Proceedings of the 4th International Semantic Web Conference, Galway, Ireland, November 2005.

Cory Doctorow. Metacrap: Putting the torch to seven straw-men of the meta-utopia. Website, August 2001. Available at http://www.well.com/~doctorow/metacrap.htm, accessed on 27.03.2007.

Edd Dumbill. PGP Signing FOAF Files, 2002. Available at http://usefulinc.com/foaf/signingFoafFiles, accessed on 13.03.2007.

Edd Dumbill. Decentralizing Software Project Registries with DOAP. In XML 2004, Marriott Wardman Park Hotel, Washington, D.C., U.S.A., November 2004. SchemaSoft. Available at http://www.idealliance.org/proceedings/xml04/papers/273/273.pdf, accessed on 04.04.2007.

Holger Lausen Jos de Bruijn Rubén Lara Michael Stollberg Axel Polleres Cristina Feier Christoph Bussler Dumitru Roman, Uwe Keller and Dieter Fensel. Web Service Modeling Ontology. Applied Ontology, 2005.

EC-IST. ICT FP7 Work Programme 2007-08, December 2006a. Available at ftp://ftp.cordis.lu/pub/fp7/ict/docs/ict-wp-2007-08_en.pdf, accessed on 05.01.2007.

EC-IST. Information society technologies 2005-2006: Strategies for Leadership, 2006b. Available at http://bookshop.europa.eu/uri?target=EUB:NOTICE:KKAB05001:EN:HTML, accessed on 05.01.2007.

ECMA. Standard ECMA-334 C# Language Specification. June 2005.

J. H. Ellis. The Possibility of Non-Secret Digital Encryption. Technical report, CESG Report, 1970.

J. H. Ellis. The Story of Non-Secret Encryption. Technical report, CESG Report, 1987.

Thomas Erl. Service-Oriented Architecture: Concepts, Technology, and Design. Prentice Hall, July 2005.

Carlos Escalante. A simple model of prolog's performance: extensional predicates. In CASCON '93: Proceedings of the 1993 conference of the Centre for Advanced Studies on Collaborative research, pages 1119–1132. IBM Press, 1993.

M. P. Evett. PARKA: A System for Massively Parallel Knowledge Representation. PhD thesis, 1994.

Al Fasoldt. Librarian: Don't use Wikipedia as source. August 2004. Available at http://www.syracuse.com/news/poststandard/index.ssf?/base/news-0/1093338972139211.xml, accessed on 16.03.2007.

Dieter Fensel, Holger Lausen, Axel Polleres, Jos de Bruijn, Michael Stollberg, Dumitru Roman, and John Domingue. Enabling Semantic Web Services. Springer, 2006.

R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter amd P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1, June 1999. Available at http://www.w3.org/Protocols/rfc2616/rfc2616.html, accessed on 04.02.2007.

Roy Thomas Fielding. Architectural Styles and the Design of Network-based Software Architectures. PhD thesis, 2000. Available at http://www.ebuilt.com/fielding/pubs/dissertation/fielding_dissertation.pdf, accessed on 04.12.2006.

C. L Forgy. RETE: A fast algorithm for the many pattern/many object pattern match problem. In Artificial Intelligence, volume 19, pages 17–37, 1982.

I. Foster. The Grid: A New Infrastructure for 21st Century Science. Physics Today, 55:42–47, 2002.

I. Foster and C. Kesselman. Globus: A Metacomputing Infrastructure Toolkit. International Journal Supercomputer Applications, 11:115–128, 1997.

I. Foster, H. Kishimoto, A. Savva, D. Berry, A. Djaoui, A. Grimshaw, B. Horn, F. Maciel, F. Siebenlist, R. Subramaniam, J. Treadwell, and J. Von Reich. The open grid services architecture, version 1.0. January 2005.

Ian Foster, Carl Kesselman, Jeffrey M. Nick, and Steven Tuecke. The Physiology of the Grid. Website, 2002. Available at http://www.gridforum.org/ogsi-wg/drafts/ogsa_draft2.9_2002-06-22.pdf, accessed on 04.04.2007.

M. S. Fox and J. Huang. Knowledge Provenance: An Approach to Modeling and Maintaining the Evolution and Validity of Knowledge. Technical report, EIL Technical Report, University of Toronto, 2003. Available at http://www.eil.utoronto.ca/km/papers/fox-kp1.pdf, accessed on 04.12.2006.

Ernest Friedman-Hill. Jess in Action. Manning Publications Co., July 2003. ISBN 1930110898.

Frederic J. Frommer. Coleman staff makes changes to Web encyclopedia bio. January 2006. Available at http://www.kansascity.com/mid/kansascity/news/politics/13750512.html, accessed on 04.04.2007.

Toshiaki Fujiki. Differences between Blogs and Web Diaries. In WWW 2005 2nd Annual Workshop on the Weblogging Ecosystem: Aggregation, Analysis and Dynamics, Chiba, Japan, May 2005.

Joseph C. Giarratano and G. Riley. CLIPS, Expert Systems: Principles and Programming. PWS, 2nd edition, 1993. ISBN 0-534-95053-1.

Dan Gillmoor. Wikipedia Shows Power of Cooperation, January 2004. Available at http://weblog.siliconvalley.com/column/dangillmor/archives/001709.shtml, accessed on 04.11.2006.

Carole Goble. Position Statement: Musings on Provenance, Workflow and (Semantic Web) Annotations for Bioinformatics. Website, September 2002. Published at Zhao (2002).

Y. Goland, E. Whitehead, A. Faizi, S. Carter, and D. Jensen. Http extensions for distributed authoring – WEBDAV, 1999. Available at http://citeseer.nj.nec.com/goland99http.html, accessed on 04.04.2007.

Jennifer Golbeck and James Hendler. Accuracy of Metrics for Inferring Trust and Reputation in Semantic Web-based Social Network. In Proceedings of EKAW 04, 2004a. Available at http://www.mindswap.org/papers/GolbeckEKAW04.pdf, accessed on 04.03.2007.

Jennifer Golbeck and James Hendler. Reputation network analysis for email filtering. In Proceedings of the 1st Conference on Email and Anti-Spam, 2004b. Available at http://www.mindswap.org/papers/Email04.pdf, accessed on 04.03.2007.

Jennifer Golbeck and James Hendler. Reputation network analysis for email filtering. In Proceedings of the 1st Conference on Email and Anti-Spam, 2004c. Available at http://www.mindswap.org/papers/Email04.pdf.

Roberto García González. A Semantic Web Approach to Digital Rights Management. PhD thesis, Department of Technologies, Universitat Pompeu Fabra, Barcelona, Spain, 2005.

Andrew Gowers. Gowers Review of Intellectual Property. Her Majesty's Treasury, December 2006. ISBN 9-780118-4083-9. Available at http://hm-treasury.gov.uk/media/583/91/pbr06_gowers_report_755.pdf, accessed on 03.02.2007.

B. Cuenca Grau, E.Sirin B. Parsia, and A.Kalyanpur. Automatic Partitioning of OWL Ontologies using $\mathcal{E}$-Connections. Technical report, 2005a. Available at http://www.mindswap.org/2004/multipleOnt/papers/Partition.pdf, accessed on 04.02.2007.

Bernardo Cuenca Grau, Bijan Parsia, and Evren Sirin. Combining OWL Ontologies Using $\mathcal{E}$-Connections. In Journal of Web Services. Elsevier Science, 2005b.

Stephan Grimm and Boris Motik. Closed World Reasoning in the Semantic Web through Epistemic Operators. In OWL: Experiences and Directions, OWL Workshop, Galway, Ireland, November 2005.

Benjamin N. Grosof, Ian Horrocks, Raphael Volz, and Stefan Decker. Description logic programs: Combining logic programs with description logic. In 12th International World Wide Web Conference (WWW 2003), pages 48–57. Communications of the ACM, 2003.

William Grosso. Java RMI. O'Reilly, November 2001. ISBN 1565924525.

Paul Groth. On the Record: Provenance in Large Scale, Open Distributed Systems. University of Southampton, Faculty of Engineering, Science and Mathematics, School of Electronics and Computer Science mini-thesis, July 2005.

Paul Groth, Sheng Jiang, Simon Miles, Steve Munroe, Victor Tan, Sofia Tsasakou, and Luc Moreau. An Architecture for Provenance Systems. Technical report, February 2006. PROVENANCE Project, IST Framework 6, D3.1.1 (Final Architecture) Deliverable.

Paul Groth, Michael Luck, and Luc Moreau. A protocol for recording provenance in service-oriented grids. In 8th International Conference on Principles of Distributed Systems (OPODIS'04), Grenoble, France, December 2004. Available at http://www.ecs.soton.ac.uk/~lavm/papers/opodis04.pdf, accessed on 16.03.2007.

Paul Groth, Simon Miles, Weijian Fang, Sylvia C. Wong, Klaus-Peter Zauner, and Luc Moreau. Recording and Using Provenance in a Protein Compressibility Experiment. In 14th IEEE International Symposium on High Performance Distributed Computing (HPDC'05), July 2005. Available at http://twiki.pasoa.ecs.soton.ac.uk/pub/PASOA/PublicationStore/hpdc05.pdf, accessed on 16.03.2007.

NGG Group. Future for European Grids: Grids and Service-Oriented Knowledge Utilities, Vision and research directions 2010 and beyond. Information Society and Media, European Commission, January 2006. ISBN 92-79-01521-4. Available at ftp://ftp.cordis.europa.eu/pub/ist/docs/grids/ngg3-report_en.pdf, accessed on 05.01.2007.

T. R. Gruber. A Translation Approach to Portable Ontologies. Knowledge Acquisition, 5:199–220, 1993. Available at http://ksl-web.stanford.edu/KSL_Abstracts/KSL-92-71.html, accessed on 16.12.2006.

Thomas R. Gruber. Toward principles for the design of ontologies used for knowledge sharing. International Journal of Human-Computer Studies, 43(5–6):907–928, 1995.

M. Grüninger and M. Fox. Methodology for the Design and Evaluation of Ontologies. In Workshop on Basic Ontological Issues in Knowledge Sharing (IJCAI'95), April 1995. Available at http://citeseer.nj.nec.com/grninger95methodology.html, accessed on 16.03.2007.

Ramanathan V. Guha, Rob McCool, and Eric Miller. Semantic search. In Proceedings of the 12th International World Wide Web Conference (WWW2003), pages 700–709, Budapest, Hungary, May 2003. ACM Press.

Peter Gutmann. Why XML Security is Broken. Website, October 2004. Available at http://www.cs.auckland.ac.nz/~pgut001/pubs/xmlsec.txt, accessed on 12.11.2006.

V. Haarslev and R. Mller. Racer: A Core Inference Engine for the Semantic Web. In 2nd International Workshop on Evaluation of Ontology-based

Tools (EON2003), located at the 2nd International Semantic Web Conference (ISWC2003), page 2736, Sanibel Island, Florida, USA, October 2003.

Stephen Harris and Dr Nicholas Gibbins. 3store: Efficient Bulk RDF Storage. In Proceedings 1st International Workshop on Practical and Scalable Semantic Web Systems, Sanibel Island, Florida, USA, 2003. Available at http://eprints.aktors.org/archive/00000273/01/psss03-swh.pdf, accessed on 16.03.2007.

Sandro Hawke. RDF Database Federations And Logic, August 2002. Available at http://www.w3.org/2002/01/rdf-databases/federation, accessed on 13.01.2007.

P. Hayes, editor. RDF Semantics – W3C Recommendation. W3C® (MIT, ERCIM, Keio), February 2004. Available at http://www.w3.org/TR/rdf-mt/, accessed on 12.10.2006.

P. J. Hayes. The Logic for Frames. In D. Metzing, editor, Frame Concept and Text Understanding, pages 46–61, 1979.

Jim Hendler. The Dark Side of the Semantic Web. Website, December 2006. Available at http://www.cs.umd.edu/~hendler/presentations/DarkSide.pdf, accessed on 27.03.2007.

Carl Hewitt. The challenge of open systems: current logic programming methods may be insufficient for developing the intelligent systems of the future. BYTE, 10(4):223–242, 1985. ISSN 0360-5280.

Pascal Hitzler, Peter Haase, Markus Krötzsch, York Sure, and Rudi Studer. DLP isnt so bad after all. In OWL: Experiences and Directions, OWL Workshop, Galway, Ireland, November 2005.

D. A. Holton and J. Sheehan. The Petersen Graph. Cambridge University Press, 1993. ISBN 0521435943.

Alfred Horn. On sentences which are true of direct unions of algebras. Journal of Symbolic Logic, 16:14–21, 1951.

I. Horrocks. Using an expressive description logic: FaCT or fiction? In A. G. Cohn, L. Schubert, and S. C. Shapiro, editors, Principles of Knowledge Representation and Reasoning: Proceedings of the 6th International Conference (KR'98), pages 636–647. Morgan Kaufmann Publishers, San Francisco, California, June 1998.

Ian Horrocks, Bijan Parsia, Peter Patel-Schneider, and James Hendler. Semantic Web Architecture: Stack or Two Towers? In Francois Fages and Sylvain Soliman, editors, Principles and Practice of Semantic Web Reasoning (PPSWR 2005), number 3703 in LNCS, pages 37–41. Springer-Verlag, 2005.

Ian Horrocks, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, and Mike Dean. SWRL: A Semantic Web Rule Language Combining OWL and RuleML – W3C Member Submission. National Research Council of Canada, Network Inference, and Stanford University, May 2004. Available at http://www.w3.org/Submission/SWRL/, accessed on 30.2.2007.

Ian Horrocks and Ulrike Sattler. Ontology Reasoning in the $\mathcal{SHOQ}$(D) Description Logic. In 17th International Joint Conference on Artificial Intelligence, August 2001.

Ian Horrocks and Ulrike Sattler. Decidability of SHIQ with Complex Role Inclusion Axioms. In Eightheenth International Joint Conference on Artificial Intelligence (IJCAI2003), pages 343–348, August 2003. Available at http://dli.iiit.ac.in/ijcai/IJCAI-2003/PDF/051.pdf, accessed on 05.03.2007.

J. Huang and M. S. Fox. Dynamic Knowledge Provenance. Technical report, EIL Technical Report, University of Toronto, June 2003. Available at http://www.eil.utoronto.ca/km/papers/kp2-TR03.pdf, accessed on 04.12.2006.

Jingwei Huang and Mark S. Fox. Uncertainty in Knowledge Provenance. In Proceedings of the European Semantic Web Symposium, Springer Lecture Notes in Computer Science, May 2004. Available at http://www.eil.utoronto.ca/km/papers/EuroSemWeb04-online.pdf, accessed on 04.12.2006.

Greg Hudson. Notes on keeping version histories of files. Website, October 2002. Available at http://web.mit.edu/ghudson/thoughts/file-versioning, accessed on 13.11.2006.

M. Humphrey, G. Wasson, K. Jackson, J. Boverhof, M. Rodriguez, Joe Bester, J. Gawor, S. Lang, I. Foster, S. Meder, S. Pickles, and M. McKeown. State and Events for Web Services: A Comparison of Five WS-Resource Framework and WS-Notification Implementations. In Proceedings of The 14th IEEE International Symposium on High Performance Distributed Computing (HPDC-14), Research Triangle Park, NC, July 2005. IEEE Computer Society Press.

James J. Hunt and Jürgen Reuter. Using the Web for Document Versioning: An Implementation Report for DeltaV. In Proceedings of the 23rd International Conference on Software Engineering, Toronto, pages 507–513, May 2001. Available at http://wwwipd.ira.uka.de/~reuter/publications/deltav.pdf, accessed on 14.02.2007.

Joichi Ito. Wikipedia attacked by ignorant reporter. Website, August 2004. Available at http://joi.ito.com/archives/2004/08/29/wikipedia_attacked_by_ignorant_reporter.html#c014592, accessed on 04.03.2007.

Ivar Jacobson, Grady Booch, and James Rumbaugh. The Unified Software Development Process. Addison Wesley Longman, 1998. ISBN 0-201-57169-2.

Javier Jaén, Artur Boronat, and José H. Canós. Federated RDF Repositories for Integrated Hybrid Museums. In ICHIM 05 - Digital Culture & Heritage / Patrimoine & Culture Numérique, September 2005. Available at http://www.archimuse.com/publishing/ichim05/Jaen.pdf, accessed on 01.04.2007.

Dean M. Jones and Ray C. Paton. Some Problems in the Formal Representation of Hierarchical Knowledge. In International Conference on Formal Ontology in Information Systems FOIS'98, In conjunction with the 6th International Conference on Principles of Knowledge Representation and Reasoning KR'98, volume 1, Trento, Italy, June 1998.

Diane Jordan and John Evdemon. Web Services Business Process Execution Language Version 2.0. OASIS, January 2007. Available at http://docs.oasis-open.org/wsbpel/2.0/CS01/wsbpel-v2.0-CS01.html, accessed on 01.04.2007.

Gilbert Kalb. Bilateral Research and Industrial Development Enhancing and Integrating Grid Enabled (bridge) technologies. In The 3rd Grid@Asia & GFK 2006 International Joint Workshop, December 2006. Available at http://www.gridforumkorea.org/workshop/2006/w_data/11/Bridge.pdf, accessed on 12.12.2006.

Yarden Katz and Bijan Parsia. Towards a Nonmonotonic Extension to OWL. In OWL: Experiences and Directions, OWL Workshop, Galway, Ireland, November 2005.

Steve Kemp. Debian server compromise. Website, July 2006. Available at, http://www.debian-administration.org/articles/417.

W. Kent. A Simple Guide to Five Normal Forms in Relational Database Theory. Communications of the ACM, 26(2):120–125, February 1983. Available at http://www.bkent.net/Doc/simple5.htm, accessed on 15.03.2007.

Michael Kifer, Georg Lausen, and James Wu. Logical Foundations of Object-Oriented and Frame-Based Languages. Journal of the Association for Computing Machinery, May 1995.

G. Klyne and J. J. Carroll. Resource Description Framework (RDF): Concepts and Abstract Syntax – W3C Recommendation. W3C® (MIT, ERCIM, Keio), February 2004. Available at http://www.w3.org/TR/rdf-concepts/, accessed on 24.11.2006.

J. Köbler, U. Schöning, and J. Torán. The Graph Isomorphism Problem: Its Structural Complexity. Birkhauser, 1993.

Vladimir Kolovski, Bijan Parsia, Yarden Katz, and James Hendler. Representing Web Service Policies in OWL-DL. In Proceedings of 4th Internation Semantic Web Conference (ISWC'05), Galway, Ireland, November 2005.

Micki Krause and Harold F. Tipton. Handbook of Information Security Management, chapter Application Program Security. CRC Press LLC, January 1998. ISBN 0849399475. Available at http://www.ccert.edu.cn/education/cissp/hism/ewtoc.html, accessed on 25.11.2006.

Bo Leuf and Ward Cunningham. The Wiki Way. Addison-Wesley Longman, March 2001. ISBN 0-201-71499-X.

Alon Y. Levy and Marie-Christine Rousset. CARIN: A Representation Language Combining Horn Rules and Description Logics. In Proceedings of 12th European Conference on Artificial Intelligence (ECAI-96), pages 323–327, Budapest, Hungary, August 1996.

Y. Liang. Enabling active ontology change management within semantic web-based applications. Technical report, Mini-thesis: PhD upgrade report. School of Electronics and Computer Science, University of Southampton, October 2006. Available at http://eprints.ecs.soton.ac.uk/13068/01/minithesis.pdf, accessed on 04.04.2007.

Jon Loeliger. Collaborating with Git. Linux Magazine, June 2006a. Available at http://www.jdl.com/papers/Collaborating_Using_Git.pdf, accessed on 03.01.2007.

Jon Loeliger. How to Git It. Linux Magazine, March 2006b. Available at http://www.jdl.com/papers/How_To_Git_It.pdf, accessed on 03.01.2007.

D. Martin, M. Paolucci, S. McIlraith, M. Burstein, D. McDermott, D. McGuinness, B. Parsia, T. Payne, M. Sabou, M. Solanki, N. Srinivasan, and K. Sycara. Bringing Semantics to Web Services: The OWL-S Approach. In Semantic Web Services and Web Process Composition: 1st International Workshop, SWSWPC, July, 2004, volume 3387, pages 26–42, San Diego, CA, USA, 2005. Springer-Verlag Berlin Heidelberg.

Vincent Massol, Jason van Zyl, Brett Porter, John Casey, and Carlos Sanchez. Better Builds with Maven. Mergere Library Press, 2006.

Ross Mayfield. Ross Mayfield's Weblog: Collaborative Proposal Development, September 2003. Available at http://ross.typepad.com/blog/2003/09/collaborative_p.html, accessed on 23.11.2006.

Ross Mayfield. Wikidmedia, July 2004a. Available at http://www.thestandard.com/movabletype/rossmayfield/archives/000387.php, accessed on 23.11.2006.

Ross Mayfield. Wikipedia Reputation and the Wemedia Project, August 2004b. Available at http://www.corante.com/many/archives/2004/08/29/wikipedia_reputation_and_the_wemedia_project.php, accessed on 23.11.2006.

Adrian McCullagh and William Caelli. Non-Repudiation in the Digital Environment. First Monday: Peer-reviewed Journal on the Internet, July 2000. Available at http://www.firstmonday.dk/issues/issue5_8/mccullagh/, accessed on 14.10.2006.

Deborah L. McGuinness and Paulo Pinheiro da Silva. Registry-Based Support for Information Integration. In Proceedings of IJCAI-2003 Workshop on Information Integration on the Web (IIWeb-03), pages 117–122, Acapulco, Mexico, USA, August 2003.

Michael McIntosh, Martin Gudgin, K. Scott Morrison, and Abbie Barbir, editors. Basic Security Profile Version 1.1. Web Services Interoperability Organization, working group draft edition, 2006.

Brendan D. McKay. Practical Graph Isomorphism. In Congressus Numerantium, volume 30, pages 45–87, 1981. Available at http://cs.anu.edu.au/~bdm/papers/pgi.pdf, accessed on 23.11.2006.

Brendan D. McKay. *nauty* User's Guide (Version 2.4). Website, December 2006. Available at http://cs.anu.edu.au/~bdm/nauty/nug-2.4b3.pdf, accessed on 4.11.2006.

Jim Melton. SQL, XQuery, and SPARQL: What's Wrong With This Picture? In XTech 2006: Building Web 2.0. IDEAlliance Inc., May 2006. Available at http://www.w3.org/2006/Talks/0301-melton-query-langs.pdf, accessed on 04.02.2007.

S. Miles, S. C. Wong, W. Fend, P. Groth, K. P. Zauner, and L. Moreau. Provenance-based validation of e-science experiments. Journal of Web Semantics, 5(1):28–38, March 2007.

Arthur Raphael Miller and Michael H. Davis. Intellectual Property: Patents, Trademarks, and Copyright. West/Wadsworth, New York, 3rd edition, 2000. ISBN 0-314-23519-1.

Libby Miller and Dan Brickley. SWAD-Europe Deliverable 3.16: Final Workshop Report. 2004. IST Project IST-2001-34732, (2004). A report of the 1st Workshop on Friend of a Friend, Social Networking and the Semantic Web, 1-2 September 2004, Galway, Ireland.

Libby Miller, Andy Seaborne, and Alberto Reggiori. Three Implementations of SquishQL, a Simple RDF Query Language. In 1st International Semantic Web Conference (ISWC2002), Sardinia, Italia, June 2002.

M. Minsky. A framework for representing knowledge. In J. Haugeland, editor, Mind Design, pages 95–128, 1981.

T. Miyazaki. The complexity of McKay's canonical labelling algorithm. In L. Finkelstein and W. M. Kantor, editors, Groups and Computation II, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, volume 28, pages 239–256, Providence, R.I., 1997. American Mathematical Society. Available at http://www.cs.trincoll.edu/~miyazaki/rutgers.ps, accessed on 23.11.2006.

Nick Moffitt. Revision Control with Arch: Introduction to Arch. Linux Journal, Nov 2004. Available at http://www.linuxjournal.com/article/7671, accessed on 09.09.2006.

Boris Motik. Reasoning in Description Logics using Resolution and Deductive Databases. PhD thesis, January 2006.

Boris Motik, Ian Horrocks, and Ulrike Sattler. Bridging the Gap Between OWL and Relational Databases. In 16th International World Wide Web Conference (WWW2007), May 2007. Available at .

Boris Motik, Ulrike Sattler, and Rudi Studer. Query Answering for OWL-DL with Rules. Journal of Web Semantics, July 2005.

William Nagel. Subversion 101: The new open source version control system promises to obsolete CVS, May 2004. Available at http://www.linux-mag.com/2004-05/subversion_01.html, accessed on 12.12.2006.

D. Nardi and R. J. Brachman. In F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, and P.F. Patel-Schneider, editors, Description Logic Handbook, chapter An Introduction to Description Logics, pages 5–44. Cambridge University Press, 2002.

B. Clifford Neuman and Theodore Ts'o. Kerberos: An Authentication Service for Computer Networks. IEEE Communications, 32(9):33–38, September 1994. Available at http://www.isi.edu/gost/publications/kerberos-neuman-tso.html, accessed on 09.09.2006.

Gary Ng. Open vs. Closed world, Rules vs. Queries: Use cases from Industry. In OWL: Experiences and Directions, OWL Workshop, Galway, Ireland, November 2005.

Ulf Nilsson and Jan Mabarluszyǹski. Logic, Programming and Prolog. 2nd edition, 2000.

NIST. NIST FIPS PUB 180, Secure Hash Standard. U.S. Department of Commerce, May 1993.

Natalya F. Noy and Michel Klein. Ontology Evolution: Not the Same as Schema Evolution. Knowledge Information Systems, 6(4):428–440, 2004. ISSN 0219-1377.

Natalya F. Noy and Deborah L. McGuinness. Ontology Development 101: A Guide to Creating Your First Ontology, March 2001. Available at http://www.ksl.stanford.edu/people/dlm/papers/ontology-tutorial-noy-mcguinness.pdf, accessed on 09.10.2006.

Christopher M. E. Painter. Tracing in Internet Fraud Cases: Pairgain and NEI Webworld. Website, May 2001. Available at http://www.usdoj.gov/criminal/cybercrime/usamay2001_3.htm, accessed on 04.12.2006.

Jack Park. Promiscuous Semantic Federation: Semantic Desktops meet Web 2.0. In Semantic Desktop and Social Semantic Collaboration Workshop (SemDesk 2006) located at the 5th International Semantic Web Conference ISWC 2006, volume 202. SRI International, CEUR-WS, November 2006. Available at http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-202/SEMDESK2006_0035.pdf, accessed on 01.04.2007.

Bijan Parsia and Evren Sirin. Pellet: An OWL DL Reasoner. In 3rd International Semantic Web Conference (ISWC2004), Hiroshima, Japan, Nov 2004.

Bijan Parsia, Evren Sirin, and Aditya Kalyanpur. Debugging OWL Ontologies. In 14th International World Wide Web Conference (WWW 2005), Chiba, Japan. Communications of the ACM, May 2005.

PASOA. Provenance Aware Service Oriented Architecture. Website, 2005. Available at http://twiki.pasoa.ecs.soton.ac.uk/bin/view/PASOA/WebHome, accessed on 09.10.2006.

Eric Prud'hommeaux. Adding SPARQL Support to MySQL. Website, May 2006. Available at http://www.w3.org/2006/Talks/0518-SPASQL/, accessed on 05.03.2007.

Eric Prud'hommeaux and Andy Seaborne, editors. SPARQL Query Language for RDF – W3C Working Draft. W3C® (MIT, ERCIM, Keio), March 2007. Available at http://www.w3.org/TR/rdf-sparql-query/, accessed on 30.03.2007.

Alan Rector, Nick Drummond, Matthew Horridge, Jeremy Rogers, Holger Knublauch, Robert Stevens, Hai Wang, and Chris Wroe. OWL Pizzas: Practical Experience of Teaching OWL-DL: Common Errors & Common Patterns. In 14th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2004), Whittlebury Hall, Northamptonshire, UK, October 2004.

Alberto Reggiori, Dirk-Willem van Gulik, and Zavisa Bjelogrlic. Indexing and Retrieving Semantic Web resources: the RDFStore Model. In SWAD-Europe Workshop on Semantic Web Storage and Retrieval, Vrije Universiteit, Amsterdam, Netherlands, November 2003. Available at http://www.w3.org/2001/sw/Europe/events/20031113-storage/positions/asemantics.pdf, accessed on 09.01.2007.

Brian Reistad, Bryan Murray, Doug Davis, Alexander Nosov, Steve Graham, Vijay Tewari, and William Vambenepe. Web Services Resource Transfer (WS-RT) 1.0. IBM, 2006.

R. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Communications of the ACM, 1978.

Ronald G. Ross. Principles Of Business Rule Approach. Aw Professional, 2003. ISBN 0-201-78893-4.

David Roundy. Implementing the darcs patch formalism . . . and verifying it. In Free and Open Source Developer's European Meeting (FOSDEM06), February 2006. Available at http://darcs.net/fosdem_talk/talk.pdf, accessed on 03.01.2007.

D De Roure, N R Jennings, and N R Shadbolt. The Semantic Grid: Past, Present, and Future. Proc. IEEE, 93:669–681, March 2005. ISSN 0018-9219.

David De Roure, Nicholas R. Jennings, and Nigel Shadbolt. The Semantic Grid: A Future eSience Infrastructure. International Journal of Concurrency and Computation: Practice and Experience, 2003.

Tom Rowland. Can you trust what you read on-line? Website, February 2007. Available http://business.timesonline.co.uk/tol/business/related_reports/identity_management/article1413273.ece, accessed on 04.04.2007.

Seth Russell. Why Should the Semantic Web be a Monotonic Logic? January 2003. Available at http://robustai.net/papers/Monotonic_Reasoning_on_the_Semantic_Web.html, accessed on 09.03.2007.

M. Sabou, C. Wroe, C. Goble, and H. Stuckenschmidt. Learning Domain Ontologies for Semantic Web Service Descriptions. In Journal of Web Semantics. Elsevier Science, 2005.

Craig Sayers and Alan H. Karp. Computing the Digest of an RDF Graph. Technical report, Hewlett Packard Laboratories, Bristol, November 2003. Available at http://www.hpl.hp.com/techreports/2003/HPL-2003-235.pdf, accessed on 04.04.2007.

Robert W. Scheifler and James Gettys. X Window System: Core and extension protocols: X version 11, releases 6 and 6.1. Digital Press, 1996. ISBN 1-55558-148-X.

M. Schmidt-Schauss. Subsumption in KL-ONE is undecidable. In R. J. Brachman, H. J. Levesque, R. Reiter, eds.: Proceedings of the 1st International Conference on the Principles of Knowledge Representation and Reasoning (KR89), Morgan Kaufmann, pages 421–431, 1989.

B. Schneier and J. Kelsey. Tamperproof Audit Logs as a Forensics Tool for Intrusion Detection Systems. Computer Networks and ISDN Systems, 1999a.

Bruce Schneier and John Kelsey. Secure Audit Logs to Support Computer Forensics. ACM Transactions on Information and System Security, 1(3), 1999b. Available at http://www.schneier.com/paper-auditlogs.html, accessed on 03.04.2007.

M. C. Schraefel, M. Karam, and S. Zhao. mspace: Interaction Design for User-Determined, Adaptable Domain Exploration in Hypermedia. In AH 2003: Workshop on Adaptive Hypermedia and Adaptive Web Based Systems, pages 217–235, 2003.

Guus Schreiber. OWL Restrictions, May 2005. Available at http://www.cs.vu.nl/~guus/public/owl-restrictions/, accessed on 15.02.2007.

Kenn Scribner. Microsoft Windows Workflow Foundation Step by Step. Microsoft Press, February 2007. ISBN 0-7356-2335-X.

Andy Seaborne. Joseki - The Jena RDF Server. Website, 2003. Available at http://www.joseki.org/, accessed on 16.03.2007.

Andy Seaborne. RDQL - A Query Language for RDF. 2004. Available at http://www.w3.org/Submission/2004/SUBM-RDQL-20040109, accessed on 15.02.2007.

Nigel R. Shadbolt, Nicholas Gibbins, Hugh Glaser, Stephen Harris, and Monica M. C. Schraefel. CS AKTive Space or How We Stopped Worrying and Learned to Love the Semantic Web. In IEEE Intelligent Systems, 2003. Available at http://eprints.ecs.soton.ac.uk/archive/00007440/01/CSaktiveSpace-ISWC.pdf, accessed on 15.03.2007.

F. Shipman and C. Marshall. Formality Considered Harmful: Experiences, Emerging Themes, and Directions on the Use of Formal Representations in Interactive Systems. In Computer Supported Cooperative Work (CSCW), volume 8, pages 333–352, 1999.

Michael Sintek and Stefan Decker. TRIPLE - A Query, Inference, and Transformation Language for the Semantic Web. In First International Semantic Web Conference on The Semantic Web, pages 364–378, June 2002.

Michael K. Smith, Chris Welty, and Deborah L. McGuinness, editors. OWL Web Ontology Language Guide – W3C Recommendation. W3C® (MIT, ERCIM, Keio), February 2004. Available at http://www.w3.org/TR/owl-guide/, accessed on 15.02.2007.

R. Housley W. Polk W. Ford D. Solo. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. IETF, April 2002. Available at http://www.ietf.org/rfc/rfc3280.txt, accessed on 13.02.2007.

R. Stevens, A. Robinson, and C.A. Goble. myGrid: Personalised Bioinformatics on the Information Grid. In 11th International Conference on Intelligent Systems in Molecular Biology, volume 19, pages 302–304, 2003.

Kilian Stoffel, Merwyn Taylor, and James Hendler. PARKA-DB: Integrating Knowledge and Data-Based technologies, 1996.

Martin Szomszor and Luc Moreau. Recording and Reasoning over Data Provenance in Web and Grid Services. In International Conference on Ontologies, Databases and Applications of SEmantics (ODBASE'03), volume 2888 of Lecture Notes in Computer Science, Catania, Sicily, Italy, nov 2003. ISBN 3-540-20498-9. Available at http://www.ecs.soton.ac.uk/~lavm/papers/odbase03.ps.gz, accessed on 15.02.2007.

V. Tan, P. Groth, S. Miles, S. Jiang, S. Munroe, and L. Moreau. Security Issues in a SOA-based Provenance System. In Third International Provenance and Annotation Workshop, Chicago, USA, May 2006a. LNCS. Available at http://eprints.ecs.soton.ac.uk/12569/01/tan06security.pdf, accessed on 04.04.2007.

V. Tan, S. Munroe, P. Groth, S. Jiang, S. Miles, and L. Moreau. A Profile for Non-Repudiable Process Documentation. Technical report, School of Electronics and Computer Science, University of Southampton, 2006b.

Bruce A. Tate and Justin Gehtland. Better, Faster, Lighter Java. O'Reilly, June 2004. ISBN 0596006764.

Patrick Taylor. Etymology of Wiki, November 2003. Available at http://c2.com/wiki/WikiEtymology/3.pdf, accessed on 04.12.2006.

Alex Toussaint, editor. Java Rule Engine API: JSR-94. Java Community Process, September 2003.

S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, C. Kesselman, T. Maguire, T. Sandholm, P. Vanderbilt, and D. Snelling. Open Grid Services Infrastructure (OGSI) Version 1.0. Global Grid Forum Draft Recommendation, June 2003.

Giovanni Tummarello, Christian Morbidoni, Paulo Puliti, and Francesco Piazza. Signing individual fragments of an RDF graph. May 2005.

United Kingdom Patent Office UKPO. Confidentiality and Confidential Disclosure Agreements (CDA) booklet. Website. Available at http://www.patent.gov.uk/patent/info/cda.pdf, accessed on 20.01.2007.

Adrian Walker. Knowledge Systems and Prolog: Developing Expert, Database and Natural Language Systems. Addison-Wesley, 2nd edition, July 1990. ISBN 0-201-52424-4.

Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Collision Search Attacks on SHA1. Technical report, 2005a.

Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding Collisions in the Full SHA-1. In The 25th Annual International Cryptology Conference (Crypto'05), Santa Barbara, California, USA, August 2005b.

Xiaoyun Wang, Hongbo Yu, and Yiqun Lisa Yin. Efficient Collision Search Attacks on SHA-0. In The 25th Annual International Cryptology Conference (Crypto'05), Santa Barbara, California, USA, August 2005c.

Glenn Wasson, Norm Beekwilder, and Marty Humphrey. OGSI.NET: An OGSI-compliant Hosting Container for the .NET Framework. Website, May 2003. Available at http://www.cs.virginia.edu/~gsw2c/grid/OGSIdotNet_framework.pdf, accessed on 04.12.2006.

K. Watanabe. Introduction of Dublin Core metadata. Journal of Information Processing and Management, 43, 2001.

E. Rowland Watkins and Denis A. Nicole. Version control in online software repositories. In Proceedings of The 2005 International MultiConference in Computer Science & Computer Engineering (SERP'05), Las Vegas, Nevada, USA, June 2005a.

E. Rowland Watkins and Denis A. Nicole. Version control in online software repositories. ACM TechNews 7(872), 2005b.

E. Rowland Watkins and Denis A. Nicole. Named Graphs as a Mechanism for Reasoning about Provenance. In Lecture Notes in Computer Science, Frontiers of WWW Research and Development – APWeb 2006: 8th Asia-Pacific Web Conference, volume 3841, pages 943–948, Harbin, China, January 2006.

Jim Whitehead. WebDAV and DeltaV: Collaborative Authoring, Versioning, and Configuration Management for the Web. Website, 2001. Available at www.webdav.org/deltav/WWW10/deltav-www10.pdf, accessed on 16.12.2006.

E. James Whitehead, Jr. and Yaron Y. Goland. WebDAV: A network protocol for remote collaborative authoring on the Web. In Proceedings of 6th European Conf. on Computer Supported Cooperative Work (ECSCW'99), pages 291–310, Copenhagen, Denmark, September 1999. Available at http://citeseer.nj.nec.com/whitehead99webdav.html, accessed on 15.12.2006.

Dennis M. Wilkinson and Bernardo A. Huberman. Assessing the value of cooperation in Wikipedia. Technical report, Information Dynamics Laboratory, HP Labs, Palo Alto, CA, February 2007. Available at http://www.hpl.hp.com/research/idl/papers/wikipedia/wikipedia.pdf, accessed on 04.04.2007.

M. J. Williamson. Non-Secret Encryption Using a Finite Field. Technical report, CESG Report, January 1974.

M. J. Williamson. Thoughts on Cheaper Non-Secret Encryption. Technical report, CESG Report, August 1976.

Michael Wilson and Daniel Dardailler. Using RDF to query multiple SQL Databases, March 2003. Available at http://www.w3.org/2002/08/qh-d11-p15.html, accessed on 21.02.2007.

Sylvia C. Wong, Simon Miles, Weijian Fang, Paul Groth, and Luc Moreau. Provenance-based Validation of E-Science Experiments. In Proceedings of 4th Internation Semantic Web Conference (ISWC'05), Galway, Ireland, November 2005.

X/Open-Group. Distributed Transaction Processing: The XA Specification – X/Open CAE Specification. X/Open Company Ltd., UK, 1992. ISBN 1-87263-024-3.

Charles Young. WF: Comparing WF rules and the Microsoft Business Rule Engine. Website, October 2005. Available at http://geekswithblogs.net/cyoung/articles/56488.aspx, accessed on 24.04.2007.

J. Zhao, C. Wroe, C. Goble, R. Stevens, D. Quan, and M. Greenwood. Using Semantic Web Technologies for Representing e-Science Provenance. In Proceedings of 3rd International Semantic Web Conference (ISWC2004), Hiroshima, Japan, November 2004a. Springer LNCS.

Jun Zhao, Carole Goble, Mark Greenwood, Chris Wroe, and Robert Stevens. Annotating, linking and browsing provenance logs for e-Science. In Workshop on Semantic Web Technologies for Searching and Retrieving Scientific Data, Florida, USA, October 2003.

Jun Zhao, Carole Goble, Robert Stevens, and Sean Bechhofer. Semantically Linking and Browsing Provenance Logs for e-Science. In International Conference on Semantics of a Networked World, pages 158–176, Paris, France, 2004b. Springer-Verlag.

Yong Zhao. Data Provenance/Derivation Workshop Position Papers and Talks, October 2002. Available at http://people.cs.uchicago.edu/~yongzh/position_papers.html, accessed on 14.12.2006.

Jianying Zhou. Non-Repudiation in e-Commerce and e-Government, 2003. Available at http://acns2003.i2r.a-star.edu.sg/ACNS03-tutorial2.pdf, accessed on 04.12.2006.