Version Control in Online Software Repositories

E. Rowland Watkins
School of Electronics & Computer Science
University of Southampton
UK, SO17 1BJ

Denis A. Nicole
School of Electronics & Computer Science
University of Southampton
UK, SO17 1BJ

***Abstract*— Software version control repositories provide a uniform and stable interface to manage documents and their version histories. Unfortunately, Open Source systems, for example, CVS, Subversion, and GNU Arch are not well suited to highly collaborative environments and fail to track semantic changes in repositories. We introduce document provenance as our Description Logic framework to track the semantic changes in software repositories and draw interesting results about their historic behaviour using a rule-based inference engine. To support the use of this framework, we have developed our own online collaborative tool, leveraging the fluency of the modern WikiWikiWeb.**

Keywords: Cryptography, Online Collaboration, Semantic Web, Version Control, WikiWikiWeb

I. Introduction

Current Open Source version control repositories, such as CVS [1], Subversion [2] and GNU Arch [3], provide a framework for tracking evolving documents and grouping them into projects and releases. Unfortunately, they lack the reasoning capability necessary to make intelligent inferences over the content and to audit its evolution.

The modern Wiki provides a very natural text-based environment for shared discussion and for the evolution of partially formed ideas. This is in sharp contrast to the strict version orientation required in source code control tools such as CVS. Within our work, we are trying to explore the value of mixing the Wiki's informality for discussions about structure and implementation with CVS's rigour in ensuring that consistent builds can be achieved. We further inform this process by providing inference tools which make it easy to deduce patterns of activity in the codebase and by requiring authentication which ensure continued individual accountability.

We thus bring together three ideas; the Semantic Web [4], the new freedoms of the WikiWikiWeb [5], and XML Digital Signature-based document signing with public-key validation [6], to deliver an online collaborative tool that can provide intelligent software management to address the limitations of earlier version control systems and help software developers build an organic grasp of their software engineering process. We have leveraged existing popular Resource Description Framework (RDF)-based ontologies, to introduce a minimal set of extensions to track the provenance of documents. Using our ontology as a description logic (DL), we have enhanced the JSPWiki [7] implementation to track online resources kept in a WebDAV [8] repository and more importantly, added an inference mechanism to allow useful deductions to be made about the evolution of project files.

Reusing existing ontologies as far as possible is important because defining a new ontology does not help in shared understanding across domains; by leveraging existing work, the semantic content of our system is immediately accessible to tools built for pre-existing ontologies. It also becomes straightforward to federate information from our system with other knowledge scattered over the Semantic Grid [9]. We have therefore taken advantage of Dublin Core [10], Friend of a Friend (FOAF) [11], and Description of a Project (DOAP) [12] to promote ontology extension.

We use Named Graphs [13] to label RDF trees that allow us to create relations between graphs, and hence make provenance statements. We have built a small Java applet to facilitate secure signing of the metadata in a browser environment, including hashes of source files. This can digitally sign RDF [14] and store the result in a Named Graph, and thus validate inputs and detect corruption in the knowledge-base. We can also use digital signatures to promote developer accountability in the software project.

Our work on RDF digital signatures has now become part of the Semantic Web Publishing framework (SWP) [15], an extension to the Named Graphs for Jena project.

The WikiWikiWeb interface makes for simple annotation of developer ideas. The system automatically generates a single Wiki page per file (Class); these can easily link to and from pages devoted to more generic ideas. The automatic page provides hyperlinks between packages and classes to support routine navigation. This and other information is automatically parsed from Java source codes.

The rest of this paper provides a overview of our online collaborative tool and its advantages over simple version controlled systems. Section II describes our description logic framework, realized as an OWL ontology. Section III goes on to outline our tool's architecture and implementation. Section IV gives an account of our system in action with some sample inferences. We discuss related work in V.

II. Ontology Design Overview

Software version control repositories like CVS manage the changes made to documents over time. CVS uses a bespoke metadata format to record the author, description and version of a document which cannot readily be shared externally. CVS uses a form of delta versioning [16] which holds the information for all file versions and their metadata contained within the same file. A well-known consequent restriction of CVS is its assumption of long-lived file names and particularly directory structures. The assumption can interact badly with modern practices such as Extreme Programming [17], where semantically simple refactorings can have complex syntactic implications on files and directory hierarchies.

Another immediate problem with older tools such as CVS is that they keep the history metadata and delta versioning [16] information together in the same logical structure. The Delta-V [18] Working Group addressed this problem by separating the history and version metadata. Subversion [2] 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) [19] framework based on open standards which can be used for semantic version control and validation.

A. Document Provenance

Document provenance is an abstract and somewhat ill-defined concept that associates authenticity with a document, based on work done by Buneman [20], Goble [21], and Szomszor et al. [22]. 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. Hence, the code unit we map onto a document is intended to be the smallest natural source object that should be updated as an entity.

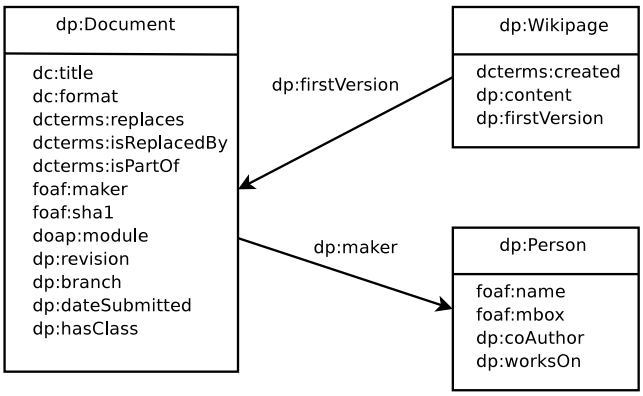

Fig. 1. Document Provenance Ontology

We have defined document provenance as a DL framework using RDF and OWL to develop an ontology to describe the evolving documents. While we can leverage existing ontologies, we believe that we need to introduce small extensions for semantic version control. Figure 1 shows the three new classes we have created which themselves inherit from FOAF and DOAP as well as importing properties from Dublin Core.

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 [23], 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" [24].

III. Implementation

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 [7] as the base our system. As its name suggests, JSPWiki uses Java Server Pages and Java Servlets found in the J2EE framework. We have retained much of the general functionality of JSPWiki, but have, however, changed various underlying components to integrate the semantic and authentication features.

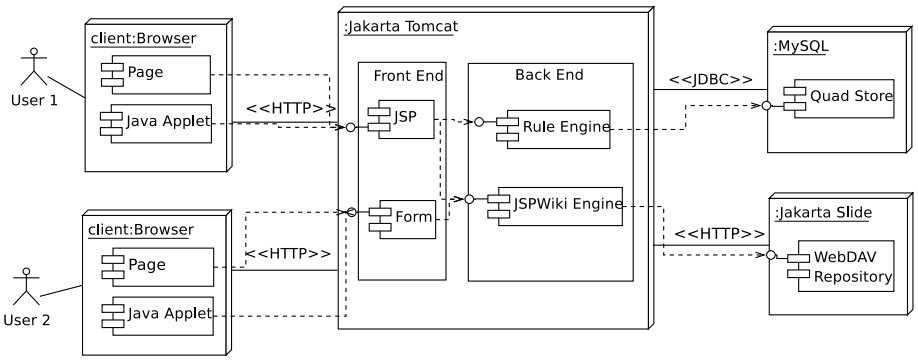

Fig. 2. Online Collaboration Tool Architecture.

Figure 2 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 Semantic Web framework and its Named Graph extension library, NG4J. We use NG4J extensively for manipulating RDF, Named Graphs and RDF digital signatures. Cryptographic support has come from the Bouncy Castle [25] JCE provider and the XML Signing of Named Graphs allows us to restrict inferences to chosen trust domains.

A. Client Side

The client side uses standard web browser capable of executing a Java™applet. Developers select their source code to upload; the applets job is to generate metadata based on those files and cryptographically sign. Note that, in this architecture, the integrity of the repository is vested in the individually signed graphs of metadata; 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.

B. Server Side

The server side is a generic J2EE web application hosting an enhanced instance of JSPWiki. The server processes requests from the client, accepting verified and signed commits as they arrive from the client, and providing the Wiki interface and inference query support. All semantic content is stored within a Named Graph quad store[1] held by a MySQL RDBMS, while all source code is in a WebDAV repository.

The new server-generated metadata created as a result of Description Logic inference on the server is, of course, only as trustworthy as the set of author metadata from which it is inferred and also the server itself. The inference interface allows this metadata provenance to be tracked through the Named Graphs.

Wikipages are stored as plaintext files, which give developers the opportunity to discuss design issues, post news, link diagrams, and make announcements.

As an additional benefit besides the adherence to the MVC 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 [26]. These can be used to support automatic build and installation of named releases onto Grid hosts.

C. Semantic Inferences

Not only does the Jena provide a comprehensive API to manipulate RDF and ontologies, it also features a stable rule-based reasoner. Convenience reasoners support RDFS and OWL entailments, while a generic reasoner lets developers define their own rules. The generic reasoner has both forward (RETE algorithm [27]) and backward (Logic Programming) engines.

Our online tool utilizes Jena's forward RETE rule engine for inference support. We have written various rules that match triple patterns to create new relations which we can then query with an RDF query language like RDQL [28]. While our description logic implementation is based on Named Graphs, it is compatible with Jena, so we can take full advantage without any problems. Indeed, we have also used Jena's OWL reasoner to periodically check the consistency of the quad store based on our ontology.

[1]The Named Graph's URI is the first element of the quad.

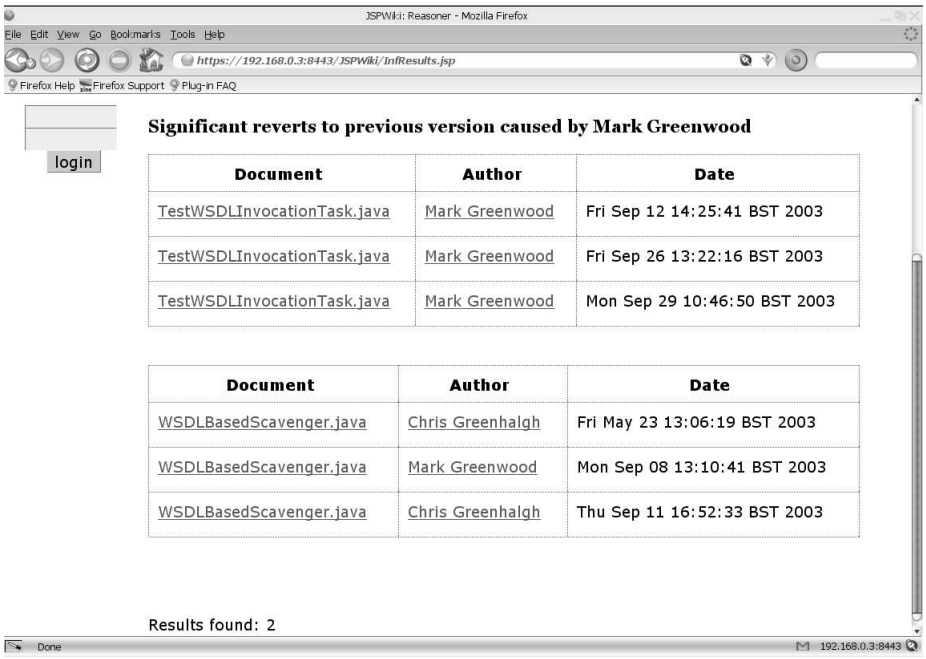

Fig. 3. Significant Reverts to previous versions.

IV. Evaluation

Our tools are primarily designed to support the management and execution of ongoing software development projects. We can, however, evaluate the effectiveness of the semantic components by bulk loading the code repositories of existing projects and drawing interesting results about their historic behaviour. We took several well known UK e-Science and international grid projects and deposited their CVS repositories into our quad store. These projects included MyGrid's [29] Taverna, and the UNICORE [30] grid project.

Rather than upload each version individually, we wrote a small application similar to the applet used in the client web browser to bulk import all project version histories. Each project was given a DOAP description and all developers found at the project's SourceForge webpage were given a synthetic FOAF description and a local PKCS #12 digital certificate. This has given us a quad store holding over 27000 Named Graphs or just over half a million quads to query. In terms of raw storage, the ratio of semantic content to source code is just over 2:1.

Each of the projects loaded is written in Java; we mined important class, package and import information and added it to the metadata about each class as we would with a normal author commit. This forms the basis of navigation in the Wiki. Lists of imports link to other classes in the repository, leveraging the intuitive interface of the Wiki.

A. Example Inferences

For our online collaborative tool to be useful to the developer community, inferences should be able to solve complex queries that are difficult or beyond the scope of simple relational database queries.

- Find all occurrences where a developer reverts the changes made by another developer.
- Find all cases where a developer modifies a file which is not part of their usual responsibility.
- A defect has been found in a class which may affect various projects, packages and classes: find all projects, packages and classes related to this class by import, by author group, or by time of modification.
- As a routine practice, we can also use metadata to revalidate the authenticity and validity of a set of project files.

The first inference provides new insight in how often developers reverse each others' modifications. Such reversals may just be as a result of a defect in a class, however, if one developer constantly reverts changes made by another, there may be a social problem that should be addressed. The same inference can also be used to see how many times a developer has reverted their own changes to a previous version; this gives a

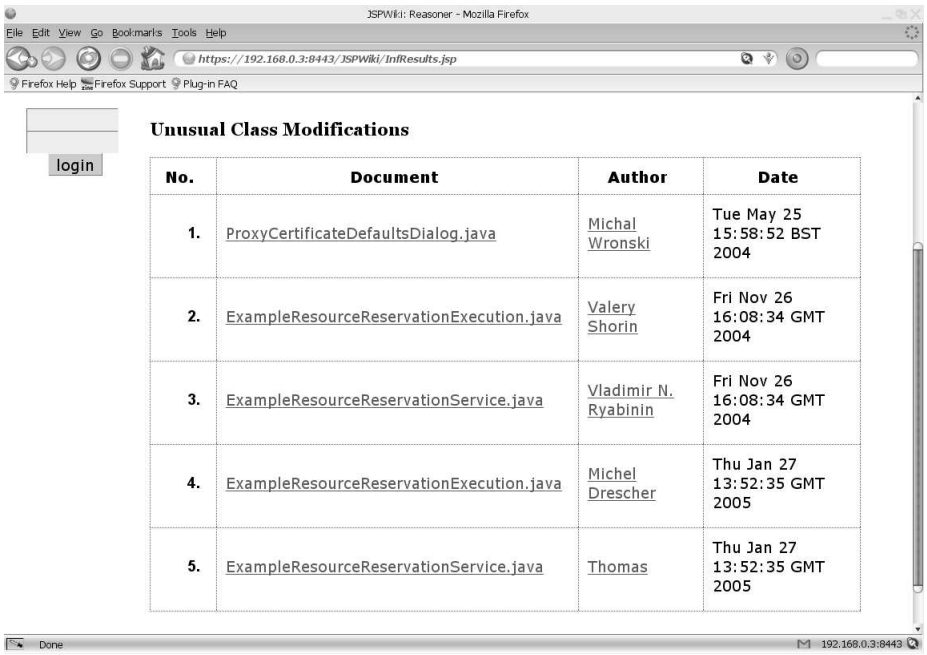

Fig. 4. Unusual Modifications to Classes.

feel for the degree to which the developer is using the repository as a scratchpad.

Figure 3 shows results when searching for significant reverts to UNICORE by a particular user. The top result is a self-revert and the second case, which might be more problematic, is a revert performed by the author of the previous version. In the codebase we examined, such reverts are infrequent.

The second inference investigates the possibility that a developer is roving outside their area of expertise. This might mean they have inadvertently introduced defects. This may also be a signal to the development community that communication needs to be improved. Figure 4 shows the results for unusual modification of classes of the entire UNICORE repository.

The third inference is useful in giving a detailed view of the impact of a defect. By pin-pointing which products are affected, warnings can be sent to users and developers. Figure 5 shows a small part of a large list of dependents for the class AbstractJob.java, a core component of UNICORE's AbstractJobObject framework.

B. Performance

The inferences outlined above have been executed using Jena's forward RETE rule engine. The backward logic rule engine is more responsive for simple interactive queries, but the forward engine performs better across broad inferences on the code base. As an example, the three queries used above took 13s, 14s and 61s respectively on a AMD K7 workstation with 1GB of memory, using the Sun™Java™1.5 JVM.

V. Related Work

Companies are starting to use Wikis as a means to encourage knowledge sharing and general collaboration [31], [32], [33] and we are enthusiastic about the mix of the freedoms of the Wiki for brainstorming coupled with the accountability associated with a validated web repository. While several Wiki implementations exist that support version control (JSPWiki can track wikipage versions), few if any have facilities similar to CVS, Subversion, or GNU Arch. Other Wikis go so far as to include Semantic content [34], but without the use of Named Graphs, digital signatures, or inference rules.

Similarly, little work has been done with regard to digital signatures in RDF. While at least two C14N algorithms exist for RDF [14], [35], only one has been implemented, which we use in our system.

VI. Conclusion

In this study, we have brought together three ideas: the Semantic Web, the WikiWikiWeb, and XML Signature-based document signing and introduced a novel online collaborative tool that provides rule-based semantic inferences as well as basic version control. Our work has led to the development of a minimal set of extensions

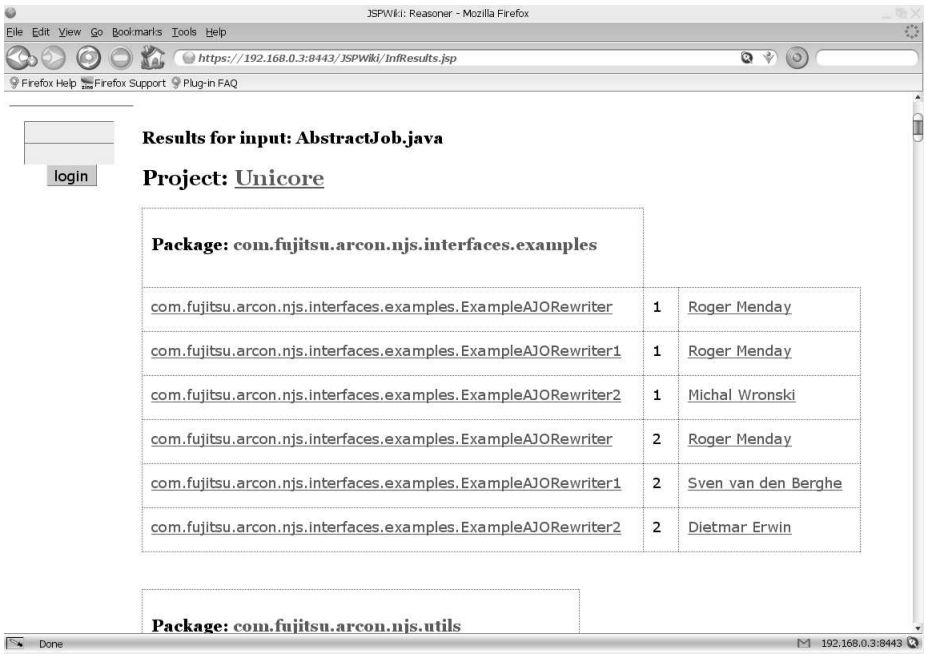

Fig. 5. List of Packages and Classes Affected by the Defective Class, AbstractJob.java.

based on popular ontologies. Using our extension set as a Description Logic, we then enhanced a basic Wiki with semantic content that described documents and their relations to different versions. We then created a cryptographic validation mechanism based on digital signatures for RDF.

Developers have the ability not only to actively collaborate in a centralized location, but also gain a greater understanding of the software engineering process through querying of the underlying knowledge-base.

Future work in this area will include exposing the NG4J quad store with web and grid services [36], which will allow external entities to make their own inferences and gain new understanding.

We will also be using the repository for live development of the WSeSS managed programme component of the UK *Open Middleware Infrastructure Institute* [37].

REFERENCES

[1] Karl Fogel and Moshe Bar. *Open Source Development with CVS*. Paraglyph Press, third edition, July 2003. http://cvsbook.red-bean.com/OSDevWithCVS_3E.pdf.

[2] Ben Collins-Sussman, Brian W. Fitzpatrick, and C. Michael Pilato. *Version Control with Subversion*. O'Reilly Media, first edition, June 2004.

[3] Nick Moffitt. Revision Control with Arch: Introduction to Arch. *Linux Journal*, Nov 2004. http://www.linuxjournal.com/article/7671.

[4] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, page 2837, May 2001.

[5] Bo Leuf and Ward Cunningham. *The Wiki Way*. Addison-Wesley Longman, March 2001.

[6] Mark Bartel, John Boyer, Barb Fox, Brian LaMacchia, and Ed Simon. Xml-signature syntax and processing. http://www.w3.org/TR/xmldsig-core/.

[7] Janne Jalkanen. JSPWiki - a JSP-based WikiWiki clone. http://www.jspwiki.org/.

[8] E. James Whitehead, Jr. and Yaron Y. Goland. WebDAV: A network protocol for remote collaborative authoring on the Web. In *Proc. of the Sixth European Conf. on Computer Supported Cooperative Work (ECSCW'99), Copenhagen, Denmark, September 12-16, 1999*, pages 291–310. http://citeseer.nj.nec.com/whitehead99webdav.html.

[9] D De Roure, N R Jennings, and N R Shadbolt. The semantic grid: Past, present, and future. *Proc. IEEE*, 93:669–681, March 2005.

[10] K. Watanabe. Introduction of dublin core metadata. *Journal of Information Processing and Management*, 43, 2001.

[11] Libby Miller and Dan Brickley. SWAD-Europe Deliverable 3.16: Final Workshop Report. 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.

[12] Edd Dumbill. Decentralizing Software Project Registries with DOAP. In *XML 2004*, 2004.

[13] Jeremy J. Carroll, Christian Bizer, Patrick Hayes, and Patrick Stickler. Named Graphs, Provenance and Trust. In *WWW2005*, 2005.

[14] Jeremy Carroll. Signing RDF Graphs. In *2nd ISWC, volume 2870 of LNCS*, July 2003. http://www.hpl.hp.com/techreports/2003/HPL-2003-142.pdf.

[15] Chris Bizer, Richard Cyganiak, and Rowland Watkins. NG4J-Named Graphs API for Jena. 2005.

[16] Greg Hudson. Notes on keeping version histories of files, October 2002.

http://web.mit.edu/ghudson/thoughts/file-versioning.

[17] Kent Black. *Extreme Programming Explained : Embrace Change*. Addison-Wesley Professional, 2004.

[18] Sunghun Kim, Kai Pan, Elias Sinderson, and E. James Whitehead Jr. Architecture and Data Model of a WebDAV-based Collaborative System. In *International Symposium on Collaborative Technologies and Systems (CTS '04)*, 2004. http://www.cse.ucsc.edu/ hunkim/papers/catacomb-CTS04.pdf.

[19] Franz Baader, Diego Calvanese, Deborah Mcguinness, Daniele Nardi, and Peter Patel-Schneider. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2003.

[20] Peter Buneman, Sanjeev Khanna, and Wang-Chiew Tan. Data Provenance. May 2001. http://db.cis.upenn.edu/Research/provenance.html.

[21] Carole Goble. Position Statement: Musings on Provenance, Workflow and (Semantic Web) Annotations for Bioinformatics, September 2002. Published at [38].

[22] 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. http://www.ecs.soton.ac.uk/~lavm/papers/odbase03.ps.gz.

[23] Berliner XML Tage 2004. http://www.berliner-xmltage.de/.

[24] Richard Cyganiak. Guus Schreiber on Semantic Web best practices, October 2004. http://dowhatimean.net/2004/10/guus-schreiber-on-semantic-web-best-practices.

[25] The Legion of the Bouncy Castle. Bouncy Castle Crypto APIs, 2004. http://www.bouncycastle.org/.

[26] Malcolm Atkinson, David DeRoure, Alistair Dunlop, Geoffrey Fox, Peter Henderson, Tony Hey, Norman Paton, Steven Newhouse, Savas Parastatidis, Anne Trefethen, and Paul Watson. Web service grids: An evolutionary approach. *Concurrency and Computation: Practice and Experience*, 17:377389, Feb 2005.

[27] C.L Forgy. RETE: A fast algorithm for the many pattern/many object pattern match problem. 1982.

[28] Libby Miller, Andy Seaborne, and Alberto Reggiori. Three Implementations of SquishQL, a Simple RDF Query Language. In *ISWC*, 2002.

[29] 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 i302 – i304, 2003.

[30] Dirk Breuer, Dietmar Erwin, Daniel Mallmann, Roger Menday, Mathilde Romberg, Volker Sander, Bernd Schuller, and Philipp Wieder. Scientific Computing with UNICORE. In *NIC Symposium 2004, Proceedings, Dietrich Wolf, Gernot Münster, Manfred Kremer (Editors), John von Neumann Institute for Computing, Jülich, NIC Series*, volume 20, pages 429–440, 2003.

[31] Ross Mayfield. Ross Mayfield's Weblog: Collaborative Proposal Development, September 2003. http://ross.typepad.com/blog/2003/09/collaborative_p.html.

[32] Jon Udell. Year of the enterprise Wiki: Lightweight Web collaboration gets down to business, Dec 2004. http://www.infoworld.com/article/04/12/30/01FEtoycollab_1.html.

[33] Business Week Online. Championing a Wiki World, Oct 2004. http://www.businessweek.com/technology/content/oct2004/tc$20041019_0375_$tc182.htm.

[34] Roberto Tazzoli, Paolo Castagna, and Stefano Emilio Campanini. Towards a Semantic Wiki Wiki Web. In *Poster Track, 3rd International Semantic Web Conference (ISWC2004)*, Nov 2004. http://iswc2004.semanticweb.org/posters/PID-LPSVVIIZ-1090243438.pdf.

[35] Craig Sayers and Alan H. Karp. Computing the digest of an rdf graph. Technical report, Nov 2003. http://www.hpl.hp.com/techreports/2003/HPL-2003-235.pdf.

[36] 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. 2004. http://www.omii.ac.uk/paper_web_service_grids.pdf.

[37] The omii product roadmap–version 2. 2004. http://www.omii.ac.uk/OMIIRoadmapV2.pdf.

[38] Data Provenance/Derivation Workshop Position Papers and Talks, October 2002. http://people.cs.uchicago.edu/~yongzh/position_papers.html.