UNIVERSITY OF SOUTHAMPTON

The Building and Application of a Semantic Platform for an e-Research Society

by

David R. Newman

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

in the
Faculty of Physical and Applied Sciences
Electronics and Computer Science

October 2011

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF PHYSICAL AND APPLIED SCIENCES
ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

by David R. Newman

This thesis reviews the area of e-Research (the use of electronic infrastructure to support research) and considers how the insight gained from the development of social networking sites in the early 21st century might assist researchers in using this infrastructure. In particular it examines the myExperiment project, a website for e-Research that allows users to upload, share and annotate workflows and associated files, using a social networking framework. This Virtual Organisation (VO) supports many of the attributes required to allow a community of users to come together to build an e-Research society.

The main focus of the thesis is how the emerging society that is developing out of myExperiment could use Semantic Web technologies to provide users with a significantly richer representation of their research and research processes to better support reproducible research. One of the initial major contributions was building an ontology for myExperiment. Through this it became possible to build an API for generating and delivering this richer representation and an interface for querying it.

Having this richer representation it has been possible to follow Linked Data principles to link up with other projects that have this type of representation. Doing this has allowed additional data to be provided to the user and has begun to set in context the data produced by myExperiment. The way that the myExperiment project has gone about this task and consideration of how changes may affect existing users, is another major contribution of this thesis.

Adding a semantic representation to an emergent e-Research society like myExperiment, has given it the potential to provide additional applications. In particular the capability to support Research Objects, an encapsulation of a scientist's research or research process to support reproducibility. The insight gained by adding a semantic representation to myExperiment, has allowed this thesis to contribute towards the design of the architecture for these Research Objects that use similar Semantic Web technologies.

The myExperiment ontology has been designed such that it can be aligned with other ontologies. Scientific Discourse, the collaborative argumentation of different claims and hypotheses, with the support of evidence from experiments, to construct, confirm or disprove theories requires the capability to represent experiments carried out *in silico*. This thesis discusses how, as part of the HCLS Scientific Discourse subtask group, the myExperiment ontology has begun to be aligned with other scientific discourse ontologies to provide this capability. It also compares this alignment of ontologies with the architecture for Research Objects.

This thesis has also examines how myExperiment's Linked Data and that of other projects can be used in the design of novel interfaces. As a theoretical exercise, it considers how this Linked Data might be used to support a Question-Answering system, that would allow users to query myExperiment's data in a more efficient and user-friendly way.

It concludes by reviewing all the steps undertaken to provide a semantic platform for an emergent e-Research society to facilitate the sharing of research and its processes to support reproducible research. It assesses their contribution to enhancing the features provided by myExperiment, as well as e-Research as a whole. It considers how the contributions provided by this thesis could be extended to produce additional tools that will allow researchers to make greater use of the rich data that is now available, in a way that enhances their research process rather than significantly changing it or adding extra workload.

Contents

List of Figures

List of Tables

Listings

DECLARATION OF AUTHORSHIP

I, **David Russell Newman**

declare that the thesis entitled

The Building and Application of a Semantic Platform for an e-Research Society

and the work presented in are my own. I confirm that:

- this work was done wholly or mainly while in candidature for a research degree at this University;
- where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
- where I have consulted the published work of others, this is always clearly attributed;
- where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
- I have acknowledged all main sources of help;
- where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
- parts of this work have been published as:

Newman, D., Bechhofer, S. and De Roure, D. (2009) myExperiment: An ontology for e-Research. *In: Semantic Web Applications in Scientific Discourse, 25-10-2009, Washington DC, US* (In Press)

Bechhofer, S., Ainsworth, J., Bhagat, J., Buchan, I., Couch, P., Cruickshank, D., Delderfield, M., Dunlop, I., Gamble, M., Goble, C., Michaelides, D., Missier, P., Owen, S., Newman, D., De Roure, D. and Sufi, S. (2010) Why Linked Data is Not Enough for Scientists. *In: Sixth IEEE eScience conference (e-Science 2010), December 2010, Brisbane, Australia.* (In Press)

De Roure, D., Goble, C., Aleksejevs, S., Bechhofer, S., Bhagat, J., Cruickshank, D., Fisher, P., Kollara, N., Michaelides, D., Missier, P., Newman, D., Ramsden, M., Roos, M., Wolstencroft, K., Zaluska, E. and Zhao, J. (2010) The Evolution of myExperiment. *In: Sixth IEEE eScience conference (e-Science 2010), December 2010, Brisbane, Australia.* (In Press)

Zhao, J., Goble, C., De Roure, D., Corcho, O., Missier, P., Belhajjame, K., Newman, D. (2011) eScience. *Book chapter of Handbook of Semantic Web Technologies* (In Press)

De Roure, D., Bechhofer, S., Goble, C. and Newman, D. (2011) Scientific Social Objects: The Social Objects and Multidimensional Network of the myExperiment Website. *In: 1st International Workshop on Social Object Networks (SocialObjects 2011), October 2011, Boston, MA, US.* (In Press)

Signed: .....

Date: .....

Acknowledgements

I would like to thank my supervisor, David De Roure, myExperiment's project team and community, the members of the e-Laboratories Technical Architecture Group both at the University of Manchester and University of Southampton and the Scientific Discourse subtask group of the W3C's HealthCare and Life Sciences (HCLS) Interest Group.

To my parents Bernard and Sheila Newman, whose continual encouragement was of great help during the course of my PhD.

Chapter 1

Introduction

"Society does not consist of individuals, but expresses the sum of interrelations, the relations within which these individuals stand."

Karl Marx, The Grundrisse (1857)

This statement sets out how a society should be something that allows people together to work collaboratively towards a goal rather than each individual attempting to reach it alone. This may be an unusual concept to some researchers, who often work with only very small groups of people.

e-Research is the development of electronic infrastructure to support scholars in conducting their research. It has evolved out of e-Science, which was more focussed on supporting physical and biological sciences, allowing researchers from fields such as arts and humanities, to benefit from the advantages that an electronic infrastructure provides. Section 2.1 explores the role of e-Research by examining two of the UK's first e-Science platform projects, Comb*e*Chem and ^{my}Grid. Informed by the review of these two projects and the e-Research needs they uncovered, section 2.2 goes on to consider Social Networking and the rise of Social Networking websites. It reviews how the features of these websites, can be incorporated into e-Research projects. Section 2.3 examines the myExperiment project[1], an e-Research website that allows users to upload, share and annotate workflows and associated files, through a social networking framework. It looks at the myExperiment model and compares it with similar projects to determine its capabilities and understand how it reconciles the concepts of e-Research and Social Networking to build a Virtual Organisation (VO) that can support an emergent "e-Research society".

The myExperiment project demonstrates how adding a social aspect to e-Research produces another layer of data beyond that of a basic e-Research project. Therefore section

[1] `http://www.myexperiment.org`

3.1 first describes the various different features of Semantic Web (SW) technologies and then considers how these technologies can be used to provide a semantic platform for all the data and metadata that is captured within this emergent e-Research society. It concludes by considering how semantic representations for various projects, both e-Research and otherwise, can be linked together by following the principles proposed by the Linked Data project[2].

Section 3.2 describes how SW technologies have been used to produce a specification, i.e. an ontology. for representing the data captured in myExperiment. It then discusses how the ontology has been designed to both reuse existing ontologies and make itself suitable for reuse within other projects. This chapter then describes how this ontology has been applied to myExperiment's data to provide an RDF API, so users can access this data through a standard means. Providing access to RDF data is just the first step; therefore myExperiment also allows users to query over this data using a SPARQL endpoint. This chapter discusses how this SPARQL endpoint was developed and how it has been adapted and supported to allow novice users to employ it. It then reviews some use cases of this SPARQL endpoint and considers how myExperiment's RDF data may need to evolve to support these. Finally, it describes how myExperiment's infrastructure has been modified to support Linked Data and the insights gained through this process.

With this semantic representation of myExperiment, it has been possible to consider how similar representations could be applied to various other areas of e-Research. Chapter 4 considers three such applications. The first, Research Objects (ROs), is the concept of capturing all the elements of a piece of research or a research process, in such a way that it is possible for the user to use the RO to reproduce this. myExperiment already provides a means for aggregating items that are interrelated and representing research processes in the form of workflows and their enactment. This makes it ideally suited to providing an interface for building, sharing and evaluating ROs, as well as informing the design of the Research Objects architecture, drawing on the insights gained by building the myExperiment ontology.

Another key area of e-Research is Scientific Discourse. Scientific or Scholarly Discourse is the collaborative argumentation of different claims and hypotheses, with the support of evidence from experiments, to construct, confirm or disprove theories. Currently, much of this discourse is encapsulated within research papers. Section 4.2 considers various approaches for providing a standard machine-readable format for describing Scientific Discourse, providing a more accurate and succinct representation. In particular it examines how the SWAN project[3] represents scientific discourse and how it is aligning with related ontologies to provide a richer specification for this representation. It goes on to describe how the myExperiment ontology fits in this ontology alignment to give it the capability of supporting *in silico* experiments. Finally, it describes some of

[2] `http://linkeddata.org/`
[3] `http://swan.mindinformatics.org/`

the similarities between this alignment of ontologies and the architecture for Research Objects. Considering how data captured by each could be exchanged to take advantage of the tools provided by the other.

Standard Semantic Web query interfaces, such as SPARQL endpoints, are difficult for novices to use. Websites rarely provide sufficiently sophisticated interfaces to allow users to make full use of the data the website provides access to. Section 4.3 discusses a potential further application of myExperiment's Linked Data as a means of supporting a Question-Answering system for an e-Research society. In particular it considers how the execution of a question in a QA system is analogous to a research process that can be captured in a Research Object.

Chapter 5 considers what are the added benefits for users by incorporating social networking into e-research projects, particularly those of the myExperiment project. It appraises how taking the data contained within this new e-Research society and producing a Semantic Web representation has facilitated the exposing and querying of both e-Research data itself and the social metadata that surrounds it. In particular, analysing how this representation may be used to align myExperiment with similar projects using a Linked Data approach. Finally it reviews the three proposed applications for this semantic platform for an e-Research society and what is required to implement these applications to make them truly useful to members of such a society.

1.1 Research Contributions

The main contribution of this thesis has been to provide extensive insight into the design of Research Objects as a novel approach for supporting reproducible research within an e-Research society. This contribution has been as part of the e-Laboratories Technical Architecture group which also includes: John Ainsworth, Sean Bechhofer, Jiten Bhagat, Iain Buchan, Philip Couch, Don Cruickshank, Mark Delderfield, Ian Dunlop, Matthew Gamble, Carole Goble, Danuis Michaelides, Paolo Missier, Stuart Owen, David De Roure and Shoaib Sufi.

To be able to provide this insight and aid the design of Research Objects, several prerequisite research contributions were undertaken. The first of these was the review of the myExperiment data model to ascertain the abstract concepts it encapsulates to elucidate the critical entities and features that make up a generic e-Research society. A subsequent contribution has been to take the knowledge gained from this abstraction process to design an extensible and reusable OWL ontology for the myExperiment data model and then provide an RDF API to this data. A further novel contribution has been the augmentation of the main myExperiment website to integrate this RDF API to meet the principles of the Linked Data project. These three contributions provided an infrastructure to investigate the design of Research Objects, in particular exploring

how myExperiment packs could be extended to support the evolving uses cases set out by members of the e-Laboratories Technical Architecture group and how myExperiment itself may be evolved to be an application that supports full Research Objects in the future.

As a result of the way the myExperiment ontology was designed, it made it eminently possible to align with other ontologies. Another novel contribution of this thesis has been the initial alignment of the myExperiment ontology with other e-Research oriented ontologies, in the field of scientific discourse, as part of a collaboration within of the W3C's HealthCare and Life Sciences Scientific discourse sub task group[4]. This contribution also includes insight to the form that data produced through use of the alignment of these ontologies might take and how it compares to Research Objects and whether synergy exists between the two.

The final research contribution of this thesis has been to consider how the Linked Data produced for myExperiment could be used to build new e-Research applications. In particular, as a theoretical exercise, it has considered how Linked Data along with other existing tools in Natural Language Processing (NLP) could be used to provide a novel interface to myExperiment's data through a Question-Answering system. As part of this exercise it has also considered how the Research Object model may assist in both answering and representing these questions.

[4]Members include Sophia Ananiadou, Uldis Bojars, John Breslin, Gully Burns, Kei Cheung, Annamaria Carusi, Paolo Ciccarese, Tim Clark, Ron Daniel, Sudeshna Das, Anita deWaard, Alf Eaton, Ronan Fox, Matthew Gamble, Carole Goble, Tudor Groza, Christoph Lange, Joanne Luciano, Scott Marshall, Marco Ocana, Jack Park, Alexandre Passant, Satya Sahoo, Matthias Samwald, Tony Scerri, Jodi Schneider, David Shotton, Susie Stephens, Holger Stenzhorn, Karin Verspoor, Elizabeth Wu and Jun Zhao

Chapter 2

An e-Research Society

e-Research, the use of IT infrastructure and software to support research in the sciences, arts and humanities, is becoming more and more important in all areas of research. Many e-Research projects aim to build communities around the tools and applications they construct. Supporting this community is essential for such projects because keeping the community happy is a key requirement for producing the widest possible uptake of the tools and applications provided by a project and hence new research.

Social networking websites are a way of allowing communities to interact. By allowing a community of users to first self-organise into a social network and then interact with each other to discuss their shared interests, can provide invaluable information to and about the community. In the case of e-Research this benefits both those responsible for developing the tools and applications for the community and those wishing to reuse items contributed to it.

The myExperiment project is one of the first attempts to bring the users and outputs of an e-Research community together under a social networking framework. Bringing two different but symbiotic concepts together, is not a straightforward task and requires a lot to be learnt about in what areas users in the community need the support of a social network and how best to provide this in an efficient and user-friendly manner.

2.1 e-Research

In virtually all aspects of life, Information Technology (IT) is assuming a more and more significant role, academic research is no exception. Up until recently a lot of effort has been towards providing IT and electronic infrastructure for scientific subjects such as Chemistry, Physics, Biology, etc. categorised as e-Science. However, it is becoming ever more apparent that research areas in the Arts, Humanities and Social Sciences[1] would benefit from similar infrastructures and therefore e-Research is used as a more encompassing term.

Electronic infrastructure can take a number of different forms, e.g.

- High Performance Computing (HPC) systems can process huge amounts of data or run very complex simulations.
- Applications allowing processes to be automated and run by a machine saving the time and effort required to do them by hand.
- Tools for electronically logging data for experiments as they are carried out in the lab.

The Grid is often used as a term to refer to the electronic infrastructure required to facilitate this increasing reliance on collaborative, multidisciplinary research (Hey and Trefethen, 2002). The purpose of the Grid is to manage and provide access to computing resources, compute cycles and storage through the use of Middleware.

Many countries have invested in the concept of e-Science including the Netherlands[2], Denmark[3], Norway[4] and Germany[5] and the United States[6] (where it is described as Cyberinfrastructure) and the United Kingdom[7]. To the general public, probably the most well-known e-Science project is the Large Hadron Collider[8] at The European Organisation for Nuclear Research[9] (CERN).

The UK e-Science programme was created in 2000 and spanned all the scientific research councils in the UK. At this point each research council funded a number of pilot projects. These pilots covered research in the areas of:

- Environmental Science and Oceanography

[1] http://www.ahessc.ac.uk/
[2] http://www.dutchgrid.nl/
[3] http://www.escience.ku.dk/
[4] http://www.forskningsraadet.no/en/Funding/eVITA/1253954581253
[5] http://www.einfrastructures.org/content/calendar/4.7.Kunze.pdf
[6] http://www.nsf.gov/dir/index.jsp?org=OCI
[7] http://www.rcuk.ac.uk/escience/default.htm
[8] http://public.web.cern.ch/public/en/LHC/LHC-en.html
[9] http://public.web.cern.ch/public/

- Particle Physics and Astronomy
- Aerospace Engineering
- Medicine and Biological Sciences (including Biotechnology and Bioinformatics)
- Chemistry (in particular Combinatorial Chemistry and Crystallography)

Comb*e*Chem and *myGrid* were two of these pilots. Despite being focussed on different research areas, Chemistry and Bioinformatics respectively, these projects shared similar goals, namely providing users with a customised interface to manage their scientific process. This is highlighted by the use of a similar technique to aid the design of the tools for each of these projects.

2.1.1 Comb*e*Chem - Structure-Property Mapping: Combinatorial Chemistry and the Grid

The Comb*e*Chem project was a consortium made up of the Universities of Southampton and Bristol and industrial partners: Roche Discovery, Welwyn, Pfizer and IBM. It involved collaborators from both Chemistry (particularly crystallographers and combinatorial chemists) and Computer Science, as well as statisticians. Its main focus was on how e-Science infrastructure could help manage the process of synthesising new compounds using combinatorial methods, something that produces significant amounts of new data that needs to be captured and analysed. The structure of Comb*e*Chem has been as an umbrella with smaller more specific projects feeding in to meet the projects overall goal or to allow collaboration with other projects, acting as a testbed.

2.1.1.1 eCrystals Archive

One such project is the eBank-UK project, for which the eCrystals archive was one of the main deliverables. (Duke, 2009) It closely collaborated with Comb*e*Chem and the National Crystallography Service (NCS). The eCrystals Archive is a heavily adapted instance of EPrints 3[10]. The purpose of the archive is to allow crystallographers to capture from conception to publication the entire crystallographic experiment, in particular the files produced at each stage in this process (Coles et al., 2005).

The NCS collects large amounts of data from the chemical samples it analyses through processes such as X-ray diffraction. They allow crystallographers to run experiments remotely to analyse physical samples they have sent to the NCS. Once the NCS has run the first sample the data is uploaded to a folder. After this the Automated Experiment Driver Software (AEDS) can be invoked. This software allows the crystallographer access

[10] `http://www.eprints.org`

to this data, view the progress of the experiment and intervene to set parameters and decide which services to use to analyse the data. This process can vary in complexity depending on the sample and the types of analysis that the crystallographer wants to perform. Once the experiment is complete all the data, analysis and report files can be deposited into the eCrystals archive and the crystallographer who submitted the sample or any other permitted user can access these files. At some point after this, if the sample is deemed to be both interesting and of sufficient quality, these files can be published.

Comb*e*Chem's aim in this project was to develop an "e-Chemistry activity for the assessment and utilisation of chemical structure information". The NCS had existed for a significant amount of time prior to the start of the project but its work practices had remained very much off-line. Taking the NCS online as a Grid service would help reduce the amount of instrument time wasted with low quality or uninteresting samples. If the submitting crystallographer can see after initial data collection and analysis the sample is not worth proceeding with they can avoid using additional instrument time pointlessly. Being able to set their own parameters may also reduce the instrument time needed and would save the crystallographer having to resubmit the same sample with a different set of parameters.

Obviously by turning such a service into a Grid service raises a number of other issues including, authentication of clients, security of client data, the capture of provenance and inter-operability with other services. However, Comb*e*Chem as an inter-disciplinary project was able to provide computer scientists with the expertise to help tackle these issues.

2.1.1.2 Electronic Lab Notebook

One of the main elements of this project has been producing a digital replacement for the chemistry lab book, an Electronic Lab Notebook (ELN) (m. c. schraefel et al., 2004b). First the project observed the inherent flaws of existing paper-based lab books, demonstrated by the following difficulties they exhibit:

- Sharing information, particularly in a widely distributed community.
- Backing up information recorded.
- Proving Intellectual Property (IP) rights with sufficient rigour.
- Capturing the structure of data, to ensure all potentially crucial information is recorded.

The project determined that these difficulties could be alleviated by the careful development of an ELN as a digital lab book replacement. This ELN would connect to a

database to store all the data it captured. In such an implementation the following functionality would be provided:

- Immediate access to the data for any permitted chemist.
- A server that can easily be backed up each night to ensure no data is lost.
- A means of timestamping data as it is recorded to support IP rights. That can be verified by other chemists permitted to access the data.
- Structured but flexible forms for experiment planning and execution ensuring users submit data they may have previously forgotten.

Although paper-based lab books have their drawbacks, they also have many features that chemists make great use of:

- Ease of access, i.e. flipping back and forth through pages.
- Simplicity of data entry, i.e. no complex computer applications to learn.
- Ability to draw free-form sketches.
- Portability.
- Resilience to damage, e.g. chemical spills, dropping on the floor, etc.
- Security of storage.
- Minimal effort required to capture data.

Therefore a key requirement in the design of the ELN was to maintain most if not all of these features. Simplicity of data entry and the minimal effort required to capture data are the two most important of these features and particular effort was given to maintaining them in the ELN.

2.1.1.3 The SmartTea Project

The SmartTea project[11] was conducted to determine the best means of eliciting user requirements from the chemists for the ELN (m. c. schraefel et al., 2004a). The main problem for the designers of the ELN was understanding what and why chemists record things in their lab books. This was particularly difficult when a chemist was carrying out a complex chemistry experiment, as the designers at the time neither had an understanding of the purpose of the experiment nor the terminology being used to describe it.

[11] `http://www.smarttea.org/`

The project therefore developed a technique inspired by Dix's "Deconstructing the Cracker" process to better elicit information about how a chemist plans and carries out an experiment. (Dix et al., 2003) This technique produced the analogy of making a cup of tea being like conducting a chemistry experiment; first by determining the key features of an chemistry experiment and then recombining them in an everyday scenario that could be understood by the ELN designers. By better understanding the planning, process and expected outcome of the "experiment", the designers could observe and understand how and what the chemist recorded in their lab book during its course. This process was then iterated with more complex and Chemistry-oriented experiments to allow the designers to gain a more detailed and specific understanding.

The iterative process may have been useful for gathering the requirements needed for the ELN. However, this process cannot capture understanding of the whole chemistry domain and so the database that backs the ELN needed to be as flexible as possible to be able to represent a wide range of chemical experimentation.

2.1.1.4 Comb*e*Chem and the AKT Project

The Advanced Knowledge Technologies (AKT) project was a large multi-institution project that collaborated with the Comb*e*Chem project to assist it with modelling the chemistry domain so that data captured by the ELN could be stored in a way that chemists could make the greatest use of it. The CoAKTinG project was an example of an AKT project that used Comb*e*Chem as a testbed for integration of knowledge technology tools (Bachler et al., 2004a).

One of the most significant knowledge technology tools that the Comb*e*Chem project decided to take up was that of a triplestore. This was used to act as a database back-end for the ELN, instead of a relational database system such as MySQL or PostgreSQL. Triplestores are used to store Resource Description Framework (RDF) data in the form of subject-predicate-object triples. This process is explained in detail in section 3.1. At the time Comb*e*Chem decided to use a triplestore, there were a limited number of applications available. Being seen as one of the more established applications, Jena was chosen. Section 3.1.5.1 compares and contrasts current triplestore applications including Jena.

To model the data stored by Comb*e*Chem's triplestore a schema was written using RDF Schema (RDFS) (Hughes et al., 2004), (see section 3.1.1). The schema defined processes as well as substances, using RDFS's *subClassOf* property to define the hierarchy of each, e.g. *FiltrationWithBuchnerFunnel* is a *subClassOf Filtration*. Building on these concepts it was possible to define an experiment plan, which could then be instantiated when the experiment was carried out to make sure the plan was followed correctly. When an experiment plan was instantiated it also allowed observations to be made in the form

of annotations to the experiment instance. All of this was represented as RDF triples and stored in the Jena triplestore. One of the most significant aspects of the schema was the ability to create multiple instances of the experiment, allowing the same or a different chemist to repeat the experiment.

An experiment plan consisted of process-product pairings. This model required each experiment plan to start with a process, where one or more of the experiment ingredients go through this process, to produce a product. This product was then used in the next process, possibly with one or more other experiment ingredients. This was continued until the final experiment product was produced (Hughes et al., 2004). This was a very simplistic model with a single central spine. However the model could be extended, where it was necessary to have multiple paths to produce a set of products before a combining step. Concepts for OWL-S (Martin et al., 2004) and the Business Process Execution Language, WS-BPEL) could have helped in the design of such an extension (Alves et al., 2006).

To allow the ELN to interface with it, the triplestore had a model server with its own bespoke API (Hughes et al., 2004). This API allowed specific queries to be made using the Resource Description Query Language (RDQL). It also allowed experiment plans to be created, modified, instantiated through experiment instances and annotated with observations.

Comb*e*Chem's choice of a triplestore over a more conventional relational database system provided a flexibility of design through an extensible RDFS model for experiments. RDFS also provided the ability to inference over data created that conforms to that model. In RDFS this is mainly through the class and properties hierarchies that have been defined, e.g. before a process is inferenced it has a single type, once inferenced it may have multiple types that are the more generic forms of that process. In the case of Comb*e*Chem this was particularly useful for search and even categorization of experiments, as experiments that share the same or similar process-product pairs could be searched for.

2.1.1.5 Other Comb*e*Chem Projects

There have been a number of other projects that were either part of CombeChem or have collaborated with it. One in particular was the Schools Malaria project, in collaboration with the e-Malaria project[12] (Frey et al., 2006). The focus of this project was quite different: its aim was to get school students more interested in science. The idea was to provide students with an interactive web interface for designing and testing, through computational simulation, their own anti-malaria drugs. This would lead to further discussion, allowing the students to gain some appreciation of the world of research.

[12] `http://chemtools.chem.soton.ac.uk/projects/emalaria/`

Building the web interface required pulling together several tools and applications, such as distributed drug databases, under a common search interface, as well as integrating molecule editing and visualization tools.

EliCIT was designed as a web-based knowledge elicitation system from planned experiments (Dupplaw et al., 2003). Although the system was not specifically designed for chemistry experiments, the Comb*e*Chem project and the testbed provided by the ELN presented an interesting use case. EliCIT could examine an experiment plan and highlight the factors / ranges to be investigated. It could also help manage the results obtained to see how these were affected by changing parameters and capture any experiences / insights gained through this process using dynamic questionnaires with free-form comments. This was particularly useful when the collaborators of an investigation are widely distributed.

2.1.2 myGrid: Directly Supporting the e-Scientist

The myGrid project was, in its first phase, a collaboration between the Universities of Manchester, Southampton, Nottingham, Newcastle, and Sheffield together with the European Bioinformatics Institute and industrial partners: GSK, AstraZeneca, IBM and SUN.

The first goal of the project was to understand the bioinformatic tasks undertaken by a biologist (Stevens et al., 2001). This would allow tools to be built to support and, in some cases, automate these tasks. At a high level these tasks arranged themselves in a similar way to many fields in e-Science, as a five step lifecycle (Oinn et al., 2006). As a new user the steps proceeded as follows:

1. Discover and reuse services and previous experiments.
2. Build new experiments, using the discovered services / experiments.
3. Execute and monitor these experiments as they run.
4. Collect results and provenance and analyse.
5. Share the new experiments / services with community.

Understanding this lifecycle, the project set out to build two applications, "one that supports the analysis of functional genomic data, and the other to support the annotation of a pattern database" (Hey and Trefethen, 2002). To be able to support these two applications, a somewhat complex architecture was required.

The $^{my}Grid$ architecture originally had three levels (Goble et al., 2003). At the bottom there were existing external services that are used by bioinformaticians. Some of these

were legacy services wrapped in the Soaplab framework, so that services could be accessed using the Simple Object Access Protocol (SOAP), others were provided directly. Above this there were the *myGrid* core services that provided a high level middleware layer. This accessed the underlying external services using Web Service and Grid communication protocols. These core services included the FreeFluo workflow enactment engine, an Open Grid Services Architecture (OGSA) distributed query processor and an information repository for the user to store experiment data. It also provided its own registries and discovery tools for both workflows and the services they encompassed, as well as management for ontologies and metadata for describing with these workflows / services. Further services supported personalisation, event notification and provenance management.

Above the core services middleware layer was the application layer. Applications could access the core services through a gateway that provided a single point of access to the whole system, making it easier to interface. Talisman[13] was one such application for rapid prototyping that used the gateway to directly interact with the FreeFluo workflow enactment engine. Two other applications included *myGrid* Workbench and Taverna Workflow Environment, these are now considered as a single application, called the Taverna Workbench.

2.1.2.1 Transparent Access to Multiple Bioinformatics Information Sources (TAMBIS)

TAMBIS was a precursor to the *myGrid* project with its original funding finishing in December 1998. However, it provided useful insights prior to the start of the *myGrid* project (Goble et al., 2001). The aim of the project was to allow bioinformaticians to express and gain results from source-independent rather than just source-dependent queries.

The world of Bioinformatics has and continues to produce many information resources (databases). In 2010 the Nucleic Acids Research Database contained 1230 separate databases compared to just over 200 in the equivalent Molecular Biology Database in 2000 (Cochrane and Galperin, 2010), (Baxevanis, 2000). Even at the time of TAMBIS's conception, multiple heterogeneous molecular biology databases existed and it was clear that some means of performing queries across two or more of them would be essential (Markowitz, 1995).

TAMBIS was designed to provide users with a visual query interface that would allow them to specify source-independent queries that use multiple information resources. The ability to provide such an interface and transcribe the source-independent queries into ordered sequences of source-dependent queries was facilitated by a terminology server

[13] `http://talisman.sourceforge.net/`

using an ontology that specified the available information resources. This sequence of queries could then be executed by a middleware layer. The flow of data between these queries is critical and the ontology is essential for defining the interoperation between heterogeneous and often restrictive information resource interfaces.

By 2001 TAMBIS had written a pilot Java^{TM} applet that integrated five protein information resources. TAMBIS could continually be developed to encompass more information resources and perform more complex queries. However, during development it was noted how some information resources had inadequate query interfaces, reducing recall and precision. Some of these interfaces were in a state of flux making it difficult to keep TAMBIS's ontology up to date (Stevens et al., 2003). Although TAMBIS provided the useful facility of working out which information resources needed to be queried and in what order it was limited to what the ontology defined as capable. Other possibilities may have existed that the user may have wanted to explore.

2.1.2.2 The myTea Project

The TAMBIS project provided some insight to a number of the tasks performed by a bioinformatician. To gain a more detailed picture of the bioinformatician's work, the myTea project[14] was undertaken. This project was similar to SmartTea, (see section 2.1.1.3), in that it used an analogy to a common simple activity to represent the more complex task the researcher was trying to tackle. myTea was designed to help discover how bioinformaticians could better integrate their work electronically, to save time and make it easier to share their results with others (Gibson et al., 2005). The analogy of assembling a jigsaw puzzle was used to represent the work of a bioinformatician.

A bioinformatician's workflow is quite different to that done by chemists in the SmartTea project. All their work is done *in silico*, so they do not need an electronic lab book in the same way chemists do. A bioinformatician uses many applications from different sources that provide little integration. This creates several problems: first, it slows the progress that the bioinformaticians can make because they need to copy and paste, download and upload information between applications; second, because of this piecemeal transfer of data it is very difficult for the user to keep track of their data. What applications have been used to produce a file? What was the original data used to produce it? Has a file been manually edited? This problem arises because despite there being plenty of web applications available, these are not integrated within the desktop where the bioinformaticians do their work. The myTea project attempted to elicit the requirements for this integration.

[14] `http://www.mytea.org.uk/`

2.1.2.3 Taverna Workbench

The Taverna Workbench's design was informed by the outcomes of both the TAMBIS and myTea projects. Like TAMBIS a main requirement of the Taverna Workbench was that it had to allow bioinformaticians to perform queries across multiple information sources. The Taverna Workbench also shared similarities with the ELN built as part of the Comb*e*Chem project. Like the ELN it allowed the user to interact with web-based applications seamlessly, as well as providing an interface to a number of different $^{my}Grid$ components (Oinn et al., 2006).

As previously discussed in section 2.1.2, the e-Science lifecycle played a critical role in the design of the Taverna Workbench. Sitting at the application level it made use of $^{my}Grid$ core services to support each step of this lifecycle. To be able to do this effectively the Workbench had to have certain characteristics.

Taverna users would typically be biologists / bioinformaticians, not expert programmers. So a simple interface was needed to allow ad hoc workflows to be rapidly designed. Workflows are visualized in a Graphical User Interface (GUI) as part of the Workbench, allowing the user to edit them, adding new services and connecting them up to existing ones. Once completed these visualizations could be exported as SVG files using Graphviz's dot package[15]. Figure 2.1 is an example visualisation of a Taverna workflow. Visualisation of the workflow made designing them easier for the non-expert user. However, the underlying model still needed to be understood. The Taverna Workbench used a dataflow-centric model, with which most bioinformaticians are already familiar.

Taverna Workbench users had to be able to use the services they wanted, so the onus was on the Workbench to support them rather than the other way round. A consequence of this was that that it often created a high-level of complexity when linking two heterogeneous services together. Therefore a multi-tiered hierarchy was necessary to hide from the users the complexity of the middleware that allowed two services to interface but still provide sufficient information so they could see how the services linked up and allow them to manage control flows and fault tolerance policies.

Fault tolerance is another consequence of integrating distributed heterogeneous services. At any time a service may not respond, may return erroneous data or simply fail; therefore the overall workflow needed to handle these faults gracefully, to execute the workflow the best is could. However, it was up to the user to define these fault tolerance parameters so that the results it finally returned to the user were of a high enough standard.

To support the last step of the e-Science lifecycle, the Taverna Workbench also allowed users' experiments to be stored and processed within a Web-accessible system, which can

[15] `http://www.graphviz.org/`

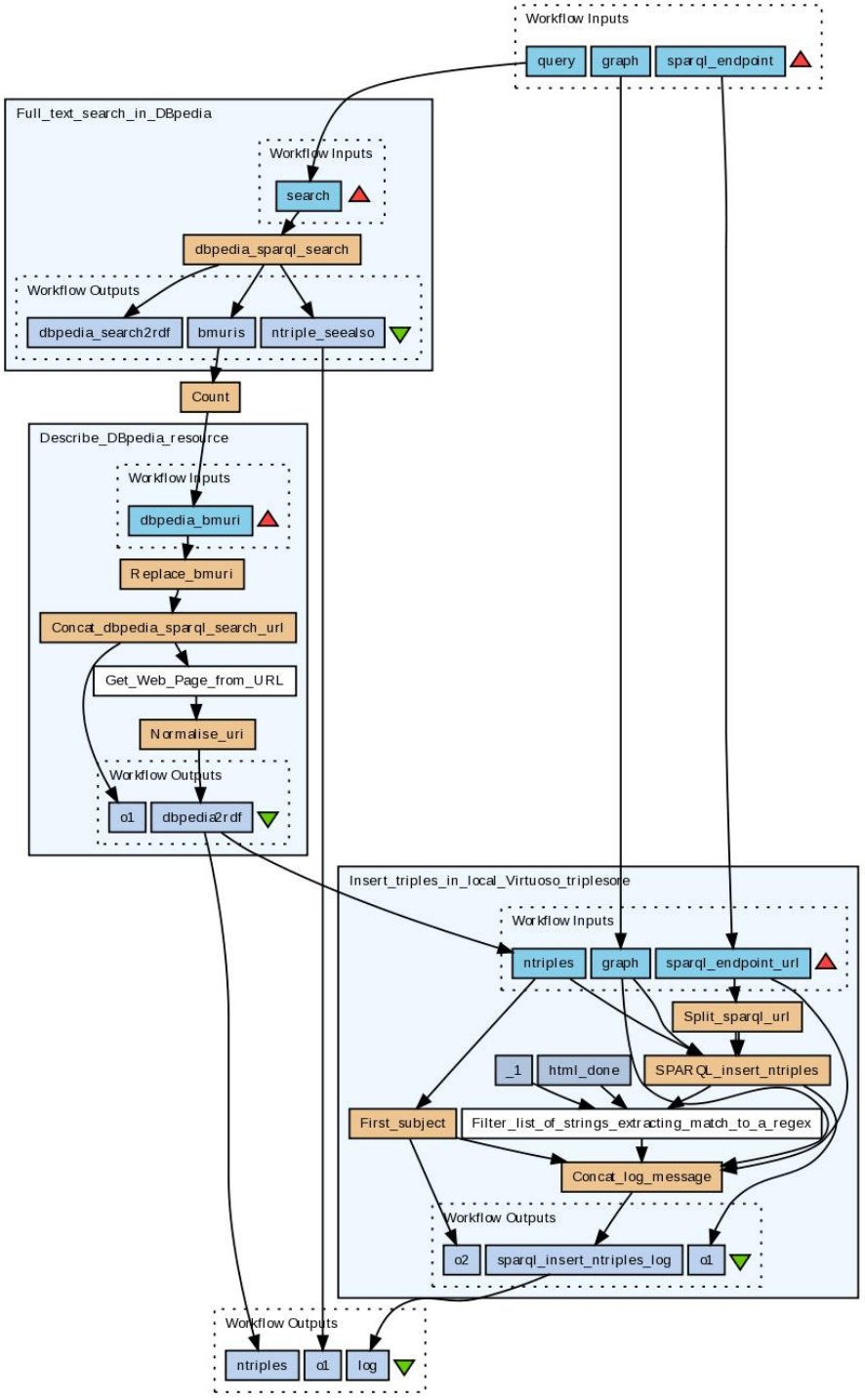


FIGURE 2.1: Taverna Workflow Visualisation (Due to `http://www.myexperiment.org/workflows/966`)

made use of Semantic Web (SW) technologies such as triplestores and ontologies, further to this the system could generate reports from this analysis which could be annotated by the owner and then shared with other users of the owner's choice. Annotation was an important component of the project to allow the user to specify how reliable the data is, terms like *finished*, *unsatisfactory*, *useful* and *needs attention* helped keep track

of the work, alongside the provenance data captured. This functionality provided some ideas towards the critical goal of closing the loop between steps five and one in the e-Science lifecycle. However, there was a clear need to expand upon these ideas to make appropriate reuse easier and more reliable within this community.

The Taverna Workbench has continued to evolve since the original $^{my}Grid$ project created it. As of October 2011 is still widely used by bioinformaticans and in the last few years, the Taverna 2 Workbench has been released to allow users to design ever more complex workflows.

2.1.3 e-Research and the Semantic Web

Sections 2.1.1 and 2.1.2 have demonstrated how UK e-Science pilot projects have made use of tools and applications of e-Infrastructure. Many of these, as well as the overall architecture, i.e. OGSA, takes advantage of Web specifications such as the Simple Object Access Protocol (SOAP) and the Web Services Description Language (WSDL). Some of these tools and applications go further and make use of early Semantic Web specifications, such as Resource Description Format (RDF), RDF Schema (RDFS), the Ontology Interchange Language (OIL) and triplestores.

The Semantic Grid community believes that the new technologies emerging for the Semantic Web should be used to extend the original Grid functionality, allowing information and services to be described in a much richer way to promote both sharing and reuse (De Roure et al., 2001). Both Comb*e*Chem and $^{my}Grid$ have been actively involved in the Semantic Grid community and have been closely associated with knowledge technologies projects such as AKT. Follow-on projects have continued to use emergent SW technologies to facilitate interaction with electronic infrastructure (Bachler et al., 2004b), (Goble et al., 2003). Section 3.1 gives an overview of the most generic SW technologies and then focusses on those that can be more specifically applied to e-Research.

2.1.4 Socializing e-Research

Sections 2.1.1 and 2.1.2 have also identified how both projects have a strong focus on the user and their place within the community. This is particularly critical where the end of the e-Science lifecycle loops back to the beginning, namely where outputs from one lifecycle gets reused by the next.

The Comb*e*Chem project produced experimental work plans that could be used to instantiate experiments for many different use cases, such as verifying results, allowing multiple runs with different parameters, a protocol for reproducing a product for a further experiment, etc. The experimental work plans themselves could be re-purposed borrowing chunks of process-product pairs and splicing them into a new experiment

work plan. These use cases are very much aligned with those of the Taverna Workbench, where *in silico* workflows may be reused in similar ways. In both, these use cases may be performed by the originator or a third-party, in the latter case the supplementary information about these work plans / workflows is essential both for discovery and for the third party to understand whether the item is suitable for their intended reuse. Some of this supplementary information can only be provided by the originator, but as the item is being shared with the community, there should be some onus on them to provide additional information to verify these details, and sometimes the claims, made by the originator. To make this possible some framework is required to support the community in this endeavour. One such means that exists in the wider world is social networking websites, which allow users to distribute information, which the community can then augment with supplementary information.

2.2 Social Networking

Social Networking has existed since the beginning of mankind and indeed is exhibited within primates and other species (Krause et al., 2009). The concept of building relationships with others and them in turn making relationships with further people to build a network is innate to human civilisation (Johnson and Earle, 1987). However, it has only been since the latter half of the twentieth century that this social network has gone online. Computer networking and the advent of the Internet have allowed humans to send and receive messages from each other using computers, through email, newsgroups, bulletin boards, Internet Relay Client (IRC), Multi-User Dungeons (MUDs) etc. If logs from these were analysed it would be possible to visualise some of the first online social networks.

2.2.1 Social Networking on the Web

The intention of the Web and similar technologies at the time was to facilitate the sharing of human-readable information in a way where it could be easily navigated by humans. In the case of the Web this was achieved through hyperlinks (Berners-Lee, 1989). Initially the only real benefit the Web provided to online social networks was to allow them to publish mailing list archives and chat logs online. Some early pioneers saw the potential of the Web for supporting online social networks. By setting up web-hosting services they hoped to facilitate the development of virtual communities for these social networks. Examples of such services include Tripod.com Angelfire and GeoCities.

GeoCities in particular focussed on creating community spaces. It setup 29 neighbourhoods, allowing users to choose which community their website was hosted in. (GeoCities, 1995). Communities include *Broadway*, *CapitolHill*, *CollegePark*, *SiliconValley*, etc. each designed for a particular user group, namely actors, politicians, students and technologists.

These projects may have allowed anyone to host websites and be part of a community of like-minded people but in the early world of Web there could only be limited interaction between website host and user. Simple applications such as guest books and hit counters allowed hosts to gain feedback from users but these interactions were very basic and did not facilitate the building of a fully-interactive online social network.

As the Web evolved and in particular the development of scripting languages, such as PHP and Active Server Pages (ASP), that allowed the generation of dynamic Web pages, making it possible to design more interactive applications such as wikis and forums. Such applications provide the facility for an online social network to come together to perform a particular task, such as collaborative document editing or discussion of particular topics. However, since the beginning of the 21st century a number of successful websites

have sprung up that are oriented around the social network rather than achieving a specific task.

2.2.2 Social Networking Websites

Prior to the advent of social networking sites that as of 2009 hold the greatest market share of users i.e. MySpace Facebook, Twitter etc., a number of less well known sites were set up in the late 1990s, such as Six Degrees and SocialNet (comScore, 2010).

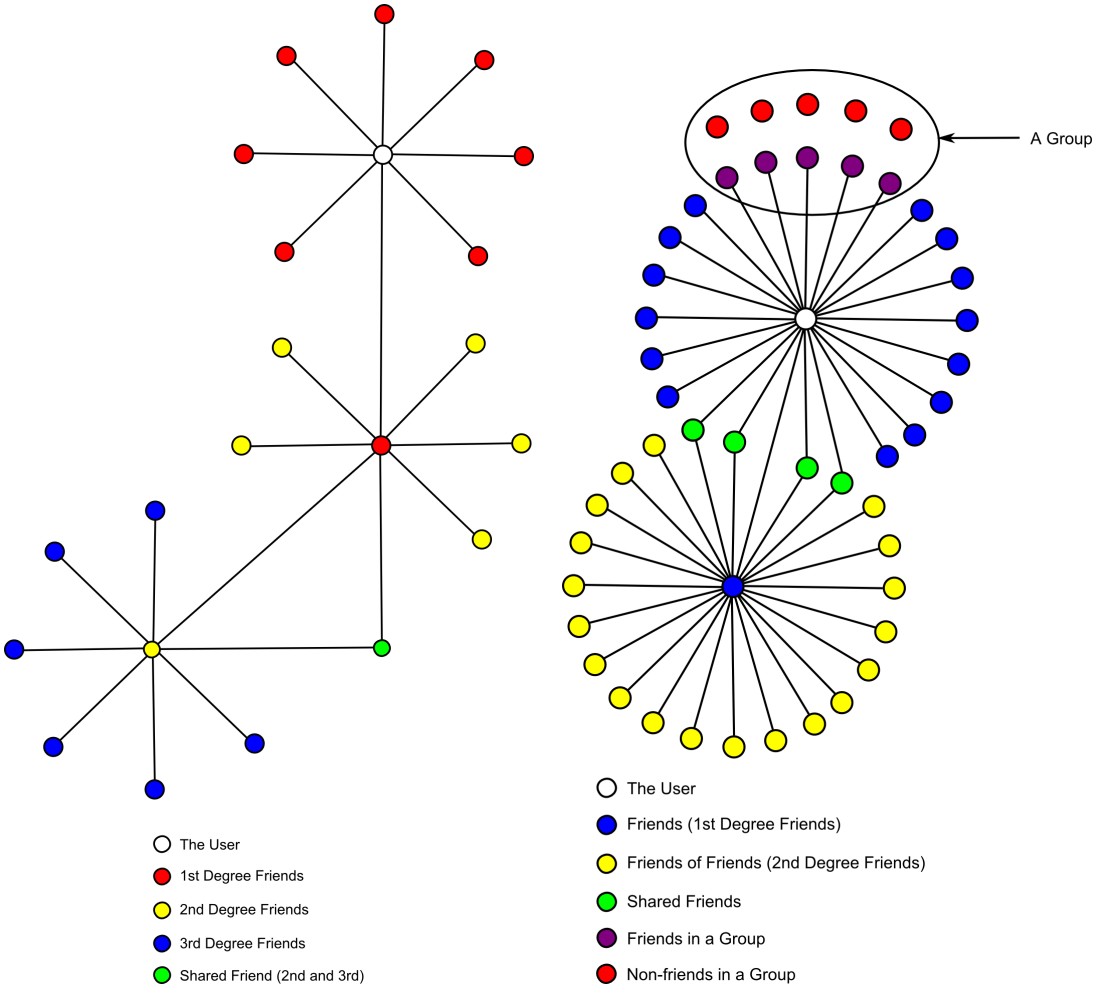
Six Degrees's social networking model follows the concept of the same name first eluded to by Frigyes Karinthy in his 1929 book *Everything is Different* and later tested by Stanley Milgram (Barabási, 2002). It is based on the idea of each member having a set of contacts and those contacts having their own contacts and so on (Bedell, 1998). As this continues "circles of associations" appear as new members list contacts who are existing members. However, the main aim of the site was to widen a member's list of contacts by allowing them to contact all members up to three degrees of separation away. A member could also explore all the links between members potentially allowing them to find any other member on the site. An example given for how this might be used is a student who has just graduated university and wants to become an environmental lawyer in Dallas. The member may well have contacts in Dallas and in environmental law but not any environmental lawyers in Dallas. However, there is a fairly high probability that one or two further degrees of separation away there will be such a contact.

More recent social networking sites such as Facebook or MySpace have had a model slightly different from that of Six Degrees. They have been more interested in the interaction between fairly close-knit groups of friends. Figure 2.2 helps to highlight the three main differences. MySpace and Facebook's models have:

1. Users with a greater number of first degree friends. Friends beyond the second degree (friends of friends) are not relevant.
2. Users have a greater amount of shared friends.
3. Explicit groups of users that may not all be friends with each other.

New members to these more recent social networking websites may have been sent an invite to join by an existing member. However, these sites have been able to create sufficient hype for users to sign up independently. In the case of Facebook, targeting the student community was particularly effective in generating this hype (Phillips, 2007). Once a user has joined the site they may then start building their social network. There are two means for a user to achieve this:

1. Request to become a friend with another user.



(a) Six Degrees Social Network Model (b) More Recent Social Networking Website Model

FIGURE 2.2: Comparison of Social Network Models

2. Request to join an explicit group.

Alternatively, a user can be invited by another to become a friend or join a group. In the latter case the inviting user must at least be a current member of that group. Groups may be created by any user for any legal purpose. They may reflect an existing real world group or may consist of people who have never met but share a similar interest or viewpoint. Groups allow users to share content with each other without all the users having to be friends with each other.

Once a user has established their initial social network, they can start to take advantage of the other features provided by the website. The most common feature is the ability for a user to send messages to members of their social network. This may be a direct message to one or more other users or an announcement sent out to a group. Some sites have the concept of the user's 'Wall', where users can post messages that can be viewed by all the members of that user's social network.

Other common features include a user being able to share photos and videos (content) within their social network or parts thereof. This then allows members of the the social network including the user themselves to tag the content with key words and users that appear within it. Beyond this permitted users can comment on this content. Being able to create an event and ask members of their social network whether they plan to attend is also a recurring feature.

Different social networking sites have additional features such as Facebook's applications or 'poke' feature. These features help differentiate social networking sites. Often these additional features might be designed to meet a particular user requirement of the user community, which is unique to that particular social networking site. Beyond the main social networking sites smaller sites exist, which are designed for a particular user community. SocialGo[16] even allows people to design their own social networking site for a particular community.

However, there are two main activities that social networking sites support:

1. The ability to socialise with other users. Although this is nothing new, the type of social interaction this allows goes beyond what was previously possible. As section 2.2.3 describes.

2. The ability to share and collaboratively undertake tasks, such as tagging photos. This does not just have implications in the social world but also in the business world.

2.2.3 The Pros and Cons of Social Networking Websites

As is often the case with new technologies there are both pros and cons that need to be weighed up to evaluate the benefits to both the individual and the wider community.

One of the first social networking sites to become popular in the United Kingdom was called Friends Reunited[17]. This was based on an equivalent website in the US called Classmates.com[18] (Clark, 2003). The main purpose of these sites was to allow old school friends to find each other after leaving school and losing contact. Many social networking sites' inherent architecture facilitate such a task. Beyond this some sites even provide applications to assist with this, such as suggesting users who share more than one friend in common. Before social networking sites, finding old school friends could be extremely time consuming. One problem was determining where to start, a second was coming to a dead end. Social networking sites are not only a good starting point but they generally provide a number of routes of investigation, i.e. multiple mutual friends.

[16] http://www.socialgo.com/
[17] http://www.friendsreunited.com/
[18] http://www.classmates.com/

There are many different means of media and communication that can be used to organise events. Social networking sites have several useful features that combined allow them to stand out from other options. First, the potential audience for the event is already there as the user's social network. This may be friends with whom a user wants to organise a night out, a social group that a user wants to organise a day trip for or even a larger group with a shared cause that want to organise an activity. This may be something quite jovial like a Flash Mob or more serious such as a political demonstration (Leigh, 2008), (Schlesinger, 2007).

A second feature of social networking sites is they combine features such as being easy to adapt, rapidly interactive and good at collation. This first allows quite specific functionality to be designed for sending invitations and receiving responses from invitees in a short time. To an extent this is possible with a group email or text message, however when responses come back, it requires the organiser to collate these responses. Social networking sites such as Facebook provide the invitee with a set of options (e.g. yes, no and maybe), responses can then be collated, allowing the organiser not only to know who but how many people are likely to be attending.

Before the advent of social networking sites staying in contact with more than a small group of friends was difficult. Some estimates suggest an inner circle of 15 friends, beyond which the chances of losing contact with friends that are less close is significantly higher (Geoghegan, 2009). Many sites provide the user with news feeds that alert users about the statuses of all their tens if not hundreds of friends. This may not in itself be direct contact but with the ease of sending a message between users, this can help prompt them to stay in contact, even if they no longer meet up in person.

Although there are many good things about social networking sites they also have their share of issues that may dissuade someone from using them. It has been regularly reported that Facebook's privacy settings are too complicated, causing users to allow access to things they would prefer to keep more private (Bilton, 2010). Even when social networking sites try to simplify their setting this sometimes creates further problems because the new default settings may make things public that a user did not originally intend.

Even if a user masters the privacy settings they still have to deal with the potential problem of the leak of information between social friends and work 'friends'. This can easily occur when a user has friends that exist in both circles. Social networking sites have found it difficult to understand the context in which two friends may be sharing content, leading to this being shared with users neither friend intended it to be shared with. Users also have limited control over how other users share information about them. An example of this is the tagging of a user in a photo. A user may not want to ban friends from doing this but they may well not want these photos to be accessible by anyone who uses the site, including prospective employers.

Diaspora* proposes itself as an alternative to social networking sites like Facebook (Quick, 2010). The project team state their aim is to address the previously described privacy issues with Facebook. They intend to achieve this by allowing its users to create Diaspora* 'seeds' that are personal Web servers hosted by the users themselves for holding their content. Then through clear, contextual interfaces users will have complete control over who they share their content with. These web server 'seeds' interlink to build a distributed network rather than a centralised hub. This model helps tackle one of the other majors issues raised by social networking sites: "Who owns the content a user upload?" There is no concept of upload with Diaspora*, as data is stored locally and shared appropriately using GNU Privacy Guard (GPG) encryption peer-to-peer.

Although this does appear to solve a lot of issues that are prevalent within existing social network sites, it does create its own new problems. There is an intention to allow existing social networks to act as seeds within this new network to allow users to keep their existing friends and share their existing content. However, it is currently uncertain how well this will work.

Another major problem is whether many users will have both the know how and hardware to be able to host their own 'seed'. Again, Diaspora* provide a "paid turnkey hosted service" that a user can use as their seed, but which they can decide to move to their own privately hosted seed at any time. It is as yet unclear whether this service will allow the user to maintain full Intellectual Property (IP) rights on the content they deposit.

A further issue may be that users are able to view the logs of who has accessed their content as these would presumably be hosted locally. This may at first appear to be a good idea, but it may affect user interaction with viewers of content concerned about what the user hosting the content might think about their viewing.

As has been highlighted, Diaspora* have a number of issues that require clarification, these should be come clearer when the system is launched and its user base begins to grow. The main question still remains if enough users of existing social networking sites will be so concerned about the privacy and IP rights of their content, that they will go to the effort of switching.

2.2.4 Social Networking for e-Research Community

As the community for a social networking site grows and the list of requirements becomes longer and more complex, inevitably some proportion of the users will become unhappy with certain aspects. As has been described, effort can be made to address these concerns but it is very difficult to fully resolve them. New social networking sites may better provide for a certain type of user but ultimately the question is whether the niche of a particular site outweighs the benefits of using a larger more generic one.

Designing a social networking site for an e-Research community is no different from any other niche. If the site can provide specific features and functionality to the user that a generic site cannot, then it has a place. From what has been identified in section 2.1, it is clear that the second activity described at the end of section 2.2.2, "The ability to share and collaboratively undertake tasks", is a key motivator for developing a social networking site in e-Research and specific functionality would be a necessity for a site such as this.

However, if this functionality is only used occasionally, it will be difficult to build up real user interaction within the social network. As observed in section 2.2.3 by Diaspora*, asking a user to abandon or even demote an existing social networking site might be a step too far. Being able to interact with these major sites provides the potential to facilitate greater user interaction between users of the niche site. Section 2.3 considers how this might be achieved as part of a detailed overview about the design of the e-Research focussed social networking site myExperiment.

2.3 The myExperiment Project

The myExperiment project started in 2007 and has been a collaboration between the Universities of Manchester and Southampton originally funded by JISC under the Virtual Research Environments programme and by Microsoft's Technical Computing Initiative (De Roure, 2010a). It has involved collaborators from computer science, bioinformatics, social science and chemistry, amongst others. It was conceived out of a need for communities to be able to share their experiments. The initial user group envisaged was that of the *myGrid*'s Taverna community who produce workflows representing in silico experiments (De Roure et al., 2007). As described in section 2.1.2.3, the Taverna Workbench has allowed a user to integrate heterogeneous web services from different sources to build a workflow, saving them both the time and complexity required to manage this manually. However, building a workflow from scratch still requires extensive expertise, that takes time to learn. By allowing the community to share workflows, it both aids this learning process, saves re-invention and facilitates rapid innovation and re-purposing to build new workflows.

The effort required to build a workflow is not trivial, therefore it is not unreasonable for a designer to only be happy to share it with a limited set of colleagues. Unlike other e-Science community websites that either mandate against this or do not have such a facility, (e.g. OpenWetWare[19] (Kelly et al., 2008)), the myExperiment project acknowledges that this is very important to the user and if they are confident that only certain users can access their workflow, they are more likely to share, albeit in a limited fashion, and become part of the community.

Being aware of existing social networking projects like those discussed in section 2.2.2, myExperiment knew the design and functionality of the website would be heavily user-driven. For this reason myExperiment describes itself as being in "perpetual beta", adding new features as users require them. For this reason at the outset of the project myExperiment chose to use a web scripting framework that supported agile development and rapid innovation, namely Ruby-on-Rails (De Roure et al., 2009), (Thomas et al., 2007).

Ruby-on-Rails uses a Model-View-Controller (MVC) architecture. Using such an architecture means there are separate files of code for database management (in myExperiment's case a MySQL database), the design of the Web page and everything in between. This facilitates the generation of skeleton files for each task, which can then be easily modified for the particular user requirement. It also supports the creation of intuitive URLs, (e.g. the group with ID equal to 9 is http://www.myexperiment.org/groups/9), making it possible for someone to type in the URL by hand rather than having to search for a link. Although the myExperiment website was designed for an initial community of

[19] `http://openwetware.org/`

Taverna Workbench users because of the the Ruby-on-Rails architecture it was easy to expand support for other workflow communities such as Triana[20] and Kepler[21], as well as other research areas such as Chemistry, Astronomy and Social Sciences (De Roure et al., 2007). This has been assisted by the ease of deploying Ruby-on-Rails plugins or Gems to support the codebase.

The myExperiment codebase underlying the website is licensed under the 3-clause *New BSD License*[22] and is available to download from RubyForge[23], a SVN-backed code repository hosting site (De Roure, 2009). By doing this, as well as providing configuration settings, has allowed other similar projects to reuse the myExperiment codebase to a greater or lesser extent or simply use it for ideas capture. Projects that have reused the codebase in some way include SysMO-DB[24], BioCatalogue[25], MethodBox[26], SKUA's Spacebook[27] and HPC/NA's APACE.

Over time, the perpetual beta development of the myExperiment has allowed four capabilities to be distilled that are requisite for building what has come to be described as a Social Virtual Research Environment (VRE, i.e. a VRE supported by a social network (De Roure et al., 2008):

1. Facilitate management of *Research Objects (ROs)*.
2. Support a *social model*.
3. Provide an *open extensible environment*.
4. Provide a platform to *action* research.

The third capability is in part achieved by having a design ethos like that of myExperiment, namely and agile development environment, open source licensing and customisable settings. However, like the other capabilities the model is key in being able to implement these. When the fourth capability refers to action research this means it in some ways facilitates research to be carried out. In the most basic sense this is the ability to run a workflow through the myExperiment website and collect the outputs it produces. A more implicit interpretation is that making research outputs accessible, it makes it possible for users to help other make use of their research contributions to do their own research.

[20] `http://www.trianacode.org/`
[21] `https://kepler-project.org/`
[22] `http://www.opensource.org/licenses/bsd-license.php`
[23] `http://rubyforge.org/projects/myexperiment`
[24] www.sysmo-db.org/
[25] `http://www.biocatalogue.org/`
[26] `http://www.methodbox.org/session/new`
[27] `http://code.google.com/p/skua/wiki/Spacebook`

2.3.1 The myExperiment Model

The myExperiment model focuses on three main areas (Newman et al., 2009):

1. Content Management.
2. Social Networking.
3. Object Annotation.

myExperiment first and foremost manages workflows and files, collectively known as "contributions", that users upload to the site. As described earlier, a key requirement for prospective users is that they can manage how these are shared with other users. For this reason the myExperiment model defines additive policies for each contribution that describe the permissions other users have to view, download and edit a particular contribution entry. However, to ask a user to define each individual permission every time they upload a contribution would be excessively time consuming and this is one of the main motivations for supporting a social model.

After a user has signed up to myExperiment, much like MySpace, Facebook or other social networking sites, (see section 2.2.2), a user may make friends by either sending or accepting friendship requests from other users (Goble and De Roure, 2007). They may also join groups using a similar membership requests mechanism. When it comes to sharing a contribution a user may en masse assign permissions to their friends and groups, whilst still retaining finer-grained access control to assign permissions to individual groups or friends. This task is further made straightforward by providing a list of choices restricted by the user's friendships and memberships.

The purpose of the social network stretches beyond the facility to simplify the sharing of contributions. It also structures the data management making it easy for user to search or discover particular contributions. For example, myExperiment has a shared contributions tab for each group and news feeds for when a user's friend uploads a new contribution to which they have access. This facilitates the final aspect of the myExperiment model, object annotation.

When a user uploads a contribution they will assign basic metadata to it, e.g. title, description, tags (i.e. keywords), etc. This is useful for helping other users to find it and understand what it is. However, this is just the opinion of the uploader. Object annotation in myExperiment is also a community activity where other users with access to that contribution can assign their own tags and rate, review, comment on, add a citation and bookmark it. This social curation, as well as enhancing search, provides the user with information about the quality of a contribution and potentially its usefulness to them.

Beyond the three main aspects of the myExperiment model, the perpetual beta ideology has helped to unearth additional features that a Social VRE may require. Assembling a workflow is not necessarily a task undertaken by a single person and even if it is, they may not be the person to upload it. As is highlighted by the need for sophisticated access control, workflow designers are often deeply concerned about the intellectual property of their work. Therefore, the model was modified to separate the uploader from person(s) credited for a contribution.

With any website that makes intellectual property available to others, whether it be completely public or constrained to a set of users, it is essential that a license is assigned to it and this is no different for myExperiment. However, an outcome of users being able to share workflows is that a downloaded workflow may be redesigned or re-purposed for a new task and, although the license for most workflows on myExperiment permits this, it also requires that the original workflow is attributed if the workflow is made available. For that reason the model was modified to allow contributions to be annotated with attributions.

Workflows are rarely standalone entities. They may have input or output files, documentation explaining their use or associated papers where they has been cited. A common scenario posed by users was the need to group together a set of workflows being used in a Taverna training session. Therefore the myExperiment model was modified to provide a new contribution called a 'pack'. When a user creates a new pack they assign metadata

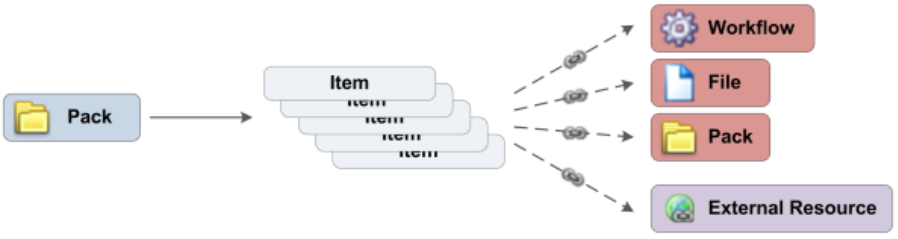


FIGURE 2.3: The Architecture of a Pack (Due to `http://www.myexperiment.org/packs`)

much like they would for any other contribution they may also add items to the pack. As shown in Figure 2.3, these items may be existing contributions, including packs, as well as references to remote entities (via URLs), for each of these items that can assign metadata describing it within the context of the pack. Users have employed packs in various ways:

1. Collating workflows to provide example or benchmarking sets for new releases of an editor.
2. Bringing together workflows of a similar purpose.
3. Building a tutorial or demo for a training course or presentation.

4. Grouping together presentations that they have given.
5. Aggregating external resources on a particular topic, e.g. web pages, ontologies, web services, enactors, etc.
6. Combining workflows with papers or presentations describing them.

To meet the final capability of a Social VRE, "providing a platform to action research", myExperiment provides an 'experiment' entity. This entity is a container for 'job' entities. Jobs take a workflow and run it in a remote enactor, i.e. a web service for running workflows, capturing both the inputs and the outputs of the workflow, as well as provenance data about the status of the job. Like contributions a user can set basic metadata, e.g. title, description, etc. for both experiments and jobs.

As described in section 2.1.2.3, Taverna workflows can be exchanged in a standard XML format called SCUFL. Other workflow types (e.g. Triana, Kepler, etc.) can also be exchanged in standard XML formats (Deelman et al., 2009), (Altintas et al., 2004). myExperiment uses scripts that can extract this data from Taverna workflow SCUFL files. These assist an uploader with metadata suggestions, as well as generate a SVG visualisation of the workflow and a structured HTML listings of its components.

Primarily the myExperiment model has been designed to support the website however to fully support capability three, "Provide an open extensible environment", there needs to be a way to access this model and the data it manages directly. A REpresentational State Transfer (REST) Web service can provide such a facility.

2.3.2 myExperiment's REST API

REST APIs are RESTful web services that give programmatic access to a particular data model. REST is similar but simpler than SOAP and WSDL-based web services (Rodriguez, 2008). RESTful web services have the following requirements:

- Use HTTP methods explicitly.
- Be stateless.
- Expose directory structure-like URIs.
- Transfer XML and/or JavaScript Object Notation (JSON).

These requirements make RESTful web services suited to providing the API for websites that have a complex underlying data model, as they resemble web page requests. Many social networking and community sites, including Facebook, Yahoo and Google use

REST APIs to allow developers to build third party application and this is also true of myExperiment.

myExperiment's REST API is built as a separate library integrated into the rest of Ruby-on-Rails code base. This approach allows for a single tabulated XML file to be used to manage all the API calls and how they interact with the underlying data model. This facilitates editing of the API via XML table editors such as Microsoft Excel, making it easy to modify or adapt for projects reusing the code base.

The REST API has several different types of request (Cruickshank, 2009). Index requests use GET requests to return listings of entities of a particular type, such as workflows, users, tags, announcements, etc. GET requests can also be used to return properties for a single entity. For entities such as users and workflows, beyond basic metadata and provenance properties, these responses include listings of associated entities. For users this includes other users with whom they are friends, groups they belong to and workflows they have uploaded. Workflows have listings of associated tags, comments, reviews, ratings, credits, attributions, citations and versions.

Associated entity listings, as well as those generated from index calls, have both a URI and resource property. The URI property is the API call required to return data for that entity whereas the resource is the entity that the results of that API call refer to. However, sometimes whatever is using the REST API may know the resource but not the URI for the API call. There is no guaranteed formulaic way to generate this from the resource, so the API has a call which takes the resource as a parameter and redirects to the URI for the API call that will return the appropriate response.

POST and PUT requests are also supported for a limited set of entities. Workflows can be uploaded, edited and have new versions of themselves submitted. Comments on contributions such as workflows can also be made using a POST request. Like requests for non-public contributions, user authorisation is required to authenticate them. The API also provides a 'whoami' call that simply authenticates a user and then redirects to a GET response for that user if successful. This call replicates the login of a user; it is not an ideal solution, as it requires the user to trust that the third party application will not use these credentials for anything but the requests they authorise.

myExperiment has a number of third party applications developed using the REST API (De Roure et al., 2010a). Although some of them, (including a couple of Google Gadgets[28]) only use public myExperiment data, others allow users to manage their account and/or contributions, (such as the Facebook application[29] and Taverna Workbench plugins). Therefore they require some form of authentication. For this reason the myExperiment API has deployed the OAuth authentication mechanism (Atwood et al., 2009).

[28] `http://www.google.com/ig/directory?synd=open&hl=en&gl=&q=myexperiment`
[29] `http://www.facebook.com/apps/directory.php?q=myExperiment`

OAuth allows a consumer application, e.g. the Facebook application, to send authenticated requests to the service provider, i.e. myExperiment. When a user sets up the consumer application they will be asked to specify a key and a secret. They can obtain this by registering that they want this application to be able to access/manage certain parts of their data on myExperiment. Once these details have been set, each time the user starts a session using the consumer application they will be required to give authorisation by being redirected via myExperiment (logging in if they have not already) and agreeing to this access. This generates an access token that they can use for the life of the session, allowing them to perform specified actions through the third party application as though they were logged into myExperiment.

One of the recurring features of myExperiment's third party applications is search. The REST API has its own call for searches that uses a Solr server to find appropriate results. Solr is a Java application that uses the Lucene Java search library to provide full-text indexing and search (Au et al., 2010). It has a REST-like HTTP/XML interface making it easy to integrate with the existing REST API, as well as the myExperiment website itself. Solr allows the application developer to specify the query syntax of their choice but by default and in the case of myExperiment it uses Apache Lucene query parser syntax (Carlson, 2006). This is a powerful syntax that allows wildcard, fuzzy, proximity and range search. Boolean operators, such as *AND*, *OR* and *NOT* can be used to build up a query and restrictions can be placed on terms so they are only searched for in particular fields.

2.3.3 Comparisons to Drupal

Drupal is a GPL-licensed open source content management platform (Buytaert, 2009) It has various features that allow it to support different sorts of websites including blogs and community-driven websites, i.e. projects similar to myExperiment. It shares many similar features:

Friendly URLs Both myExperiment and Drupal make manual URL entry feasible, rather than having to find a link for a particular entity, such as a user, group or file.

Online help Both have wikis detailing how each can be used.

Open source Both projects have open source licenses and download sites.

Platform independence Ruby-on-Rails allows myExperiment to run on most platforms. From the start Drupal has been designed to be multi-platform.

Database independence myExperiment's Ruby-on-Rails MVC architecture and Drupal's database abstraction layer provide database independence.

Caching Both employ caching to reduce the number of database queries required to load a page.

Permalinks Both have links that will not change, although the content behind them might cease to exist.

Content syndication Both have an extensive RSS feeds to allow content to be syndicated.

API support myExperiment uses a REST API whilst Drupal uses an XML-RPC API to support blogging from third party applications.

Authentication Both support external authentication, including OpenID[30] for users and OAuth for applications.

Permissions In myExperiment users control permissions to access particular entities making use of the social network. Drupal has a role-based permissions systems, where administrators define roles and each user can be assigned one or more roles.

Commenting Both allow commenting on published content. Drupal has a more complex threaded comment model.

Usage statistics myExperiment gives basic statistics for the usage of contributions, although additional information, such as the user agent, is logged but not visible. Drupal has more sophisticated tools for usage analysis tracking how a user navigates through the site and how they were referred to the site in the first place.

Searching Both have fully-indexed search.

Drupal has some additional features such as multi-language support, logging, metadata versioning and web-based administration. Ruby-on-Rails has many plugins and Gems that would allow these features to be integrated into myExperiment if they become required by its users.

Drupal also has discussion forums, blogs and polls. the first two of these were originally available in myExperiment but were disabled as they had very limited user uptake. This highlights one of the inherent differences between myExperiment and Drupal. Although Drupal supports content management it is often more focussed on the discussion of the content. For myExperiment the content and how it is shared is central. How it is annotated through tags, comments, etc. is important but this is more to improve search and curation.

The reason myExperiment and Drupal differ is down to the requirements of the user communities. Drupal is focussed on providing a generic means for managing content, (commonly web-based content, e.g. blog posts, web pages, etc.). myExperiment comes

[30] `http://openid.net/`

from a more specific requirement of workflow communities to be able to securely share their workflows with colleagues. These workflows are not web-based, i.e. they are generally files generated in a workflow editor and then uploaded. Their content is machine-readable requiring bespoke tools and user interfaces to manage their upload and representation. Being machine-readable, the need for discussion is more focussed on the usage of the workflow, e.g. "I couldn't get the workflow to run using this dataset as an input" rather than the semantics of the content itself, e.g. disagreeing with a statement in a blog post or document, which more complex to analyse.

Drupal is used by many community-driven websites but myExperiment's codebase has been reused or adapted in a number of other projects. The former provides a generic solution that could be adapted to suit particular projects, the latter is more specific and provides solutions for e-Research communities to manage, share and curate the files used in their experimentation.

2.3.4 myExperiment into the Future

After myExperiment's initial funding by JISC and Microsoft's Technical Computing Initiative ended, it received further JISC funding from 2009 as part of the Repository Enhancement programme. The purpose of this funding was to both enhance the existing functionality of myExperiment and allow other repositories to be enhanced by being able to interact with myExperiment seamlessly. This consisted of a number of tasks, one of the most prominent being the interaction with the University of Southampton's institutional EPrints repositories[31][32] and the University of Manchester's institutional Fedora[33] repository eScholar[34] (De Roure et al., 2010b).

In particular, the new phase of the project has required the ability to harvest metadata from these repositories when remote pack entries (essentially external links) are added to a pack. There are a number of ways of achieving this and each repository has differing support for these:

HTML Meta Tags Tags in the 'head' part of a web page that contains information about the page, e.g. in a institutional repository this could provide the title, authors, year of publication, etc. for the paper the web page describes

RDFa A means for embedding RDF triples (see section 3.1.1) within an HTML web page (Adida and Birkbeck, 2008). This is not too dissimilar to HTML Meta Tags but it allows existing attributes within the web page to be augmented and allows them to be interpreted by both the human and data webs.

[31] http://eprints.soton.ac.uk/
[32] http://eprints.ecs.soton.ac.uk/
[33] http://fedora-commons.org/
[34] https://www.escholar.manchester.ac.uk/

Link Alternates These can be specified 'head' part of an HTML page and can be used to describe where an alternative format for the page can be found. This alternative format may be more suitable for harvesting metadata.

Linked Data This is thoroughly described in section 3.1.7. Its key feature is that the acceptable content type(s) issued in the request determine what is returned, allowing a metadata harvesting tool to specify the content type from which it is easiest to harvest metadata.

OAI-PMH Open Archives Initiative Protocol for Metadata Harvest. A bespoke protocol that software clients can use for harvesting metadata from repositories (Lagoze et al., 2002).

OAI-PMH being bespoke should be the most reliable and accurate means to harvest metadata. However, this relies on a repository or a user-specified external resource providing such support. In some cases it would be considered either too much effort or inappropriate to implement OAI-PMH, as the type of website is too wildly different from the model of a repository. Therefore there is a need to implement a number of these methods to provide a high level of coverage. As already stated, this enhancement is a two way street. This involves improvements to the markup of myExperiment's content so that other repositories can make use of its content. In particular there has been a focus on providing "Linked Data" for myExperiment that is described in section 3.2.3 and makes up part of the intellectual contribution of this thesis.

One of the other goals of the enhancement project is to make it easier for users to find content on myExperiment. Several different approaches have been undertaken to achieve this. One is to facilitate the curation of content uploaded to myExperiment. This has been a two step process. First implementing functionality to allow the curation to take place and second getting specifically chosen curators to go through and do this curation. This is a step change from myExperiment's original model. This relied on the "wisdom of the crowd" ensuring that best content would float to the top, through good metadata and ratings and high levels of comments, viewings and downloads. Even though myExperiment has a large user base (over 3000 users according to `http://www.myexperiment.org/` as of 13/09/2010), the community shares certain traits. Some users who upload many workflows, sometimes in the same session, have insufficient time to mark all these up with the best possible metadata. Although content may be viewed and downloaded by many users, it is uncommon for them to give feedback, in the form of comments, ratings, reviews, etc. Inevitably as a website becomes more successful it is more likely to be attacked by spammers who provide unwanted content that needs to be filtered out. Expert curation is the centrepiece to this task but improved spam filtering and tag suggestion provided through content analysis is also necessary to keep down the workload of the chosen curators.

Making the content the user wants easier to find is just one side of the picture, providing enhanced functionality to perform the searching is the other. As described at the end of section 2.3.2, myExperiment uses the Solr search engine. This has a sophisticated query syntax that allows quite complex queries to be expressed. This can be incredibly useful to someone who understands the syntax but less so to everyone else. However, it can also be used as the basis for building a more sophisticated yet simple user interface. myExperiment has recently been working on providing a web-based interface to allow users to iteratively build more sophisticated queries through faceted browsing. Section 4.3 describes this interface in greater detail, as well as another potential interface that may be even more powerful for users. Before this, chapter 3 describes how myExperiment's model and data can be encoded and queried to provide the underlying search framework needed to support such a user interface. Section 4.1 goes on to explain how this encoding of data can be part of an architecture that allows researchers to exchange machine-readable representations of their research and research processes.

Chapter 3

A Society with Semantics

myExperiment is a Social VRE allowing users to share experimental data, within a social networking framework allowing it to be curated by the community through several forms of annotation. However, it is not just the data that is being shared that is interesting, the social network and user annotations are also very important.

Semantic Web technologies are a means for taking existing data and metadata and making it available for machines to understand the interrelations between objects encapsulated within the data in a standardized and widely-used way. Being able to provide this machine-readable representation is essential in making it possible to analyse the large amount of information captured within myExperiment. To be able to achieve this it is important to understand the tools available, so choices can be best made on how to represent, deliver and facilitate use of this information.

3.1 Semantic Web Technologies

One of the first attempts to capture machine-usable descriptions on the web was by using Meta Content Framework (MCF) (Guha and Bray, 1997). MCF used Directed Labelled Graphs (DLGs) comprising sets of labels, nodes and arcs, where an arc was a triple that connected up two nodes using a label. These graphs could then be represented using eXtensible Markup Language[1] (XML) (Bray et al., 2004). In combination with Minsky and other frame-based representation systems, MCF inspired the development of Resource Description Framework[2] (RDF) (Minsky, 1974) (Manola et al., 2004) (Lassila, 1998).

The new generation of the Web aims to increase the amount of machine-readable data it retains. This is unlikely to be achieved without a standard approach for representing knowledge. RDF is recognised by the World Wide Web Consortium (W3C) as the foremost standard for achieving this, after being made a recommendation in 1999 (Lassila and Swick, 1999), although latterly revised in 2004.

3.1.1 RDF and RDF Schema

Like MCF, RDF uses triples to capture data and the relationships between it. Similarly it can use XML amongst other formats (N3, NTriple and TURTLE) as a concrete syntax. According to Berners-Lee, RDF is the third layer of his Semantic Web "Layer Cake" (Berners-Lee, 2002). There are many envisioned levels above this to produce a web that stores machine-readable data that can be trusted and can be used to produce more information than the sum of its parts, i.e. through inference and other logic-based techniques. However the next layer, RDF Schema, mainly deals with the structure of the RDF data.

RDF Schema[3] (RDFS) allows RDF data to be grouped into a class/property structure, in some ways like an object-oriented programming language, such as Java (Brickley and Guha, 2004). Structuring RDF data into classes with properties and instances of those classes, allows additional triples to be generated beyond those explicitly specified in the RDF. This is however limited to the transitive properties subClassOf and subPropertyOf that allow hierarchical relationships to be defined, e.g. *dog* is a subclass of *mammal* that is subclass of *animal*, a triple can be inferred that *dog* is a subclass of *animal*. To restrict where a particular RDF property can be used, its domain and range can be defined. This constrains the type of the subject or object for a triple predicated by this property.

RDFS cannot define functional, inverse or any set theory, e.g. intersections, unions, etc., relationships. Two projects originally tried to develop a language that could represent

[1] http://www.w3.org/TR/REC-xml/
[2] http://www.w3.org/TR/rdf-primer/
[3] http://www.w3.org/TR/rdf-schema/

these relationships, the DARPA Agent Markup Language[4] (DAML) in the US and Ontology Interchange Language[5] (OIL) in Europe, these projects were eventually merged to form DAML+OIL and eventually the Web Ontology Language (OWL) (Horrocks et al., 2002).

3.1.2 OWL

The complexity of relationships that need to be defined in a domain's ontology can vary immensely depending on the domain. It is important that even the most complex logical relationships can be represented but it is as important that if only simple logical relationships are required, only these are used. For this reason there are three main species of OWL, that can represent increasingly more complex logical relationships. They are OWL Lite, OWL DL (Description Logic) and OWL Full. However, since the original specification of these three species, a subset of OWL Lite, called OWL Tiny has also been defined by European SW Advanced Development (SWAD-Europe) group[6].

OWL Lite is equivalent to the *SHIF(D)* description logic. It reuses the class / property structures provided by RDFS to define OWL classes and two types of OWL property, *DatatypeProperty* that have literal values and *ObjectProperty* that have non-literals. OWL Lite allows two or more classes or properties to be defined as equivalent and two more instances as being the same. Classes that are the intersection of two or more other classes can also be defined. OWL Lite also allows more sophisticated restrictions on the value a property can have and the characteristics a property can have such as transitive, symmetric, functional, inverse functional and the inverse of another property.

OWL DL is equivalent to the *SHOIN(D)* description logic (Horrocks and nei der, 2003). One of the main differences between the two is that OWL DL allows properties to be defined that have cardinality restrictions, e.g. a football match can have only 2 *hasTeam* relationships whereas OWL Lite only allows a functional restriction, i.e. the subject may have no more than one relationship with a particular predicate. The other main difference is that OWL DL allows singleton classes which can only ever have one instance.

OWL Full provides more flexibility than OWL DL but it is neither sound nor complete. There is no reasoning algorithm that is guaranteed to be decidable across all the data that can be expressed in OWL Full. OWL Full should only be used if it is impossible to represent a domain with OWL DL. Surveys have been carried out that have found many ontologies that are defined as OWL Full should really be OWL DL or even OWL Lite (Bechhofer and Volz, 2004).

[4] `http://www.daml.org/`
[5] `http://www.ontoknowledge.org/oil/`
[6] `http://www.w3.org/2001/sw/Europe/reports/dev_workshop_report_4/`

As OWL stores logical assertions it is possible to reason over these to provide additional information through inference. In general this can be achieved in one of two ways, at the time triples are imported or at the time of querying a triplestore. Both techniques have their advantages, but more often than not, greater processor time to perform inference at the point of query is at a higher premium than extra storage space.

3.1.3 Data, Information and Knowledge

One of the main purposes of knowledge technologies, such as the Semantic Web (SW) technologies of RDF, RDFS and OWL, is to take human concepts and convert them into machine-readable knowledge. By doing this the machine does not just store data that a human has to interpret to infer knowledge but through using inference and other logical processes can produce knowledge itself. This is far from trivial, especially with a large complex database. One of the greatest problems is that there are several stages that data must go through to become knowledge. Figure 3.1 shows how both relations and

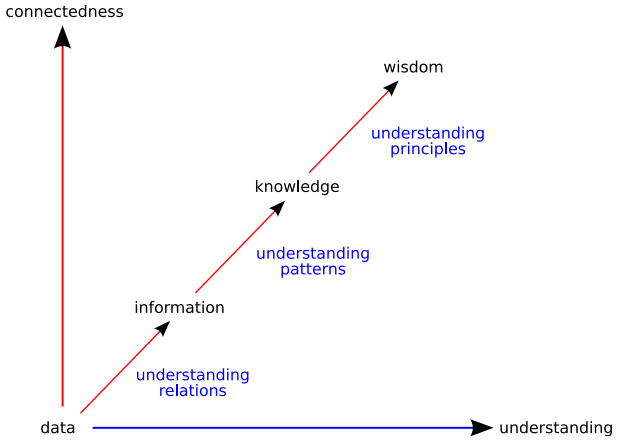


FIGURE 3.1: Data, Information, Knowledge and Wisdom (Due to `http://www.systems-thinking.org/dikw/dikw.htm` - Copyright Gene Bellinger)

patterns must be understood before data can be transformed into knowledge (Bellinger et al., 2004).

3.1.3.1 How RDF turns Data into Information

RDF uses the principle of a triple which is analogous to the construction of a sentence, namely, each triple must have a subject, an object and a predicate linking the former with the latter (Hughes et al., 2004). By using such a structure, data is converted to information because the machine understands the relationships because they have been explicitly defined.

Humans do not necessarily need relationships to be explicitly defined. Take the digital lab book replacement from the Comb*e*Chem project as an example, (see section 2.1.1.2);

a chemist using their paper-based lab book may record two related results next to each other, it may be clear to them and their colleagues that by their proximity, these results are related but a machine would not be able to make such an inference. Even if a system could be designed that reads in lab book pages and could determine that results are related, guaranteeing a high-level of accuracy would be difficult and just a single false positive could render significant amounts of further analysis invalid.

3.1.3.2 How OWL turns Information into Knowledge

Getting machines to understand relationships is only the first step towards producing knowledge and not just data; the next step is unfortunately much more difficult. As shown by Figure 3.1, it is necessary to understand patterns to convert information into knowledge. Patterns are more difficult to explicitly define than relationships. RDFS provides some assistance by defining classes and properties that can be built into hierarchies using its *subClassOf* and *subPropertyOf* properties (Manola et al., 2004). Most if not all ontology designers will enforce some sort of hierarchy, as without this structure most RDF data becomes very difficult to manage. In general, a pattern can be considered as a set of relationships that can be grouped together using one or more logical assertions, for which objects instantiated from RDFS classes can be considered an example.

Using logical assertions allows reasoning to be performed, which generates inferenced relationships/triples. Inferenced triples are a crucial part of SW technologies giving extra meaning to data without making consistency preservation more complex. Inferencing using an RDFS model can produce additional RDF type triples for all the super classes of the class an object instantiates. Further triples can also be produced by inferencing the properties of an object that have super properties, producing similar triples where the predicate is replaced with that super property. RDFS model inferencing is fairly basic compared with even OWL Lite model inferencing. Applying such an inferencing model allows a lot more triples to be inferenced through the use of logical assertions such as *unionOf*, *complementOf*, *inverseFunctionalPropertyOf*, etc. The number of inferenced triples determines how deeply an RDF schema or OWL ontology can conceptualise the patterns within the RDF data. A low number of inferenced triples does not necessary imply a poor schema / ontology; a simple domain may not have too many patterns that need to be elucidated.

3.1.4 Semantic Web Rule Language (SWRL)

SWRL is the integration of the OWL Lite and OWL DL sub-languages with the Rules Markup Language (RuleML) to allow rules to be defined (Horrocks et al., 2004). A rule consists of a body and a head, each of which are made up of zero or more atoms. An

atom may be one of several different types but mostly it represents some sort of RDF triple:

Class Has one variable that is an instance of a particular class.

Same As Has two variables, each a resource, which are the same as each other. (Equivalent to OWL's sameAs property).

Different From Has two variables, each a resource, which are different from each other. (Equivalent to OWL's differentFrom property).

Individual Property Has two variables that represent resources, the first is the subject and is associated to the second through a particular property.

Data-valued Property Has two variables, the first is a resource that is the subject; it is associated to a literal value through a particular property.

Data Range Has one variable that is a literal and in a particular data range, e.g. a list of potential values.

Built In Has one variable that is a literal, which has a built-in function applied to it. A built-in function may be a type of comparison or a mathematical, boolean, string, date/time, list or URI operation.

The purpose of the rule is to imply the atoms of the head if the atoms of the body are matched. An example of this is:

$$hasParent(?x1, ?x2) \wedge hasBrother(?x2, ?x3) \Rightarrow hasUncle(?x1, ?x3) \tag{3.1}$$

If the *hasUncle* and *hasBrother* properties have triples that match then a third triple with a *hasParent* predicate and *?x1* and *?x3* as subject and object respectively can be implied. This allows additional triples to be generated beyond those that could ever be inferred using a reasoner and an appropriate schema/ontology. As rules can define specific implications not just basic axiomatic inferences, they go beyond the knowledge representation capabilities of OWL.

3.1.5 RDF Triplestores

A relational database has its data and its structure separate,(i.e. the tables and the relationships between them. A database that stores RDF is commonly referred to as an RDF triplestore and is considered to store its data and structure together, as the structure itself is stored as data. However, care must be taken with how structural and instantiation data intermingle. Often instantiation data is more contentious, e.g. there may be disagreement with exactly what toppings make up a four seasons pizza. Including

too much instantiation data within an ontology can often make it less flexible and it is also less likely that people will use an ontology if they disagree with the creator's instantiations. A potential solution to this is to provide an ontology for defining a domain's structure and then an ontology extension to define useful instantiation data.

RDF Triplestores provide a more flexible way to manage data than relational databases because they make storing relationships as generic as it is possible to achieve, i.e. a flat list of one-to-one relationships. This can be highly beneficial as whenever some new triples are generated they can just be added directly to this list. This is not always the case in relational databases as the data may not be compatible with the tables' enforced structure. An RDF triplestore's method for storing RDF data implements the conjecture that RDF captures information in the form of relationships.

3.1.5.1 Triplestore Applications

There are many projects that have produced triplestore applications for managing RDF triples:

- Sesame[7] (Broekstra et al., 2002)
- Jena[8] (Carroll et al., 2004)
- AllegroGraph[9] (Franz Inc., 2010)
- Kowari[10] (Wood et al., 2005)
- Virtuoso[11] (Erling, 2009)
- 4Store[12] (Harris et al., 2009)

Sesame was designed as an RDF framework to support RDF Schema inferencing and querying. It supports storage by various means (i.e. relational databases, in-memory, filesystems, keyword indexers, etc.). Sesame also defines a standard HTTP protocol for interacting with the triplestore that has been adopted by many other applications.

Jena is essentially a Java library for working with RDF. Part of the library allows for data to be stored in a number of persistent datastores such as a MySQL database. Another part of the library allows communication using the Sesame HTTP protocol.

[7] `http://www.openrdf.org/`
[8] `http://jena.sourceforge.net/`
[9] `http://www.franz.com/agraph/allegrograph/`
[10] `http://kowari.org`
[11] `http://virtuoso.openlinksw.com/`
[12] `http://4store.org/`

AllegroGraph is another application that also supports the Sesame protocol and is designed to scale so that it can store billions of triples in a specialised persistent datastore. One thing that makes it different to other applications is that it has a Lisp as well as a Java client for direct access to the triplestore.

Kowari describes itself as a massively scalable "metastore" database for storing RDF and OWL metadata.

Virtuoso is a hybrid data server that can manage relational, RDF-graph and full text document data by extending an Object-Relational Database Management System (ORDBMS). It also supports the Sesame HTTP protocol.

4Store evolved from 3Store[13] triplestore and builds on Redland RDF libraries (Raptor and Rasqal) to provide an efficient, scalable and stable RDF database (Beckett, 2001).

Each of these applications have been built with a particularly purpose in mind. However, the critical factors are commonly:

- How quickly can data be imported?
- How quickly can data be queried?
- How many triples can it store before it exhibits a significant decrease in performance?
- How complex can queries become before performance declines?

Each of these features need to be considered in the light of the requirements of the project that requires a triplestore. There are further criteria that may be considered, such as developer support, inferencing support and a common querying interface. A thriving developer community at very least provides a means of reporting bugs and asking questions. As a consequence such applications are often more robust. As inferencing is computationally expensive, this is usually done at or before import time and therefore any tool can be used without being tied to a triplestore application. Almost all triplestore applications now support SPARQL as their primary querying language or at least provide it as a plugin. Therefore the last two of these criteria, i.e. inferencing support and common querying interface, can be largely overlooked.

3.1.5.2 Triplestore Querying

The primary purpose for a triplestore is to query it. SPARQL is designed specifically for querying RDF Triplestores (Prud'hommeaux and Seaborne, 2006). SPARQL uses a syntax that resembles both SQL and TURTLE. TURTLE is an alternative concrete

[13] `http://sourceforge.net/projects/threestore/`

syntax to XML for RDF. It better reflects the nature of RDF as just triples, rather than objects with properties that the XML syntax can imply. In its simplest form a SPARQL query will allow the user to search for certain patterns of RDF triples, where one or more elements of the triples are unspecified. Listing 3.1 is a query to find a group of human siblings.

```
PREFIX exp: <http://www.example.com/ontology#>

SELECT ?sibling WHERE {
  ?sibling exp:hasParent <http://www.example.com/people/Bob_Smith> .
  ?sibling exp:hasParent <http://www.example.com/people/Jane_Smith>
}
```

LISTING 3.1: Example SPARQL Query

SPARQL is not the only language that has been designed to query RDF triplestores, Resource Description Query Language (RDQL), Resource Query Language (RQL) and Sesame's Resource Query Language (SeRQL) have also been defined (Seaborne, 2004), (Karvounarakis et al., 2002), (Broekstra et al., 2002). However, as already stated in section 3.1.5.1, most triplestore applications provide a SPARQL interface and this has generally been accepted as the standard querying language.

3.1.6 Schemas and Ontologies for an e-Research Society

So far this chapter has described the basic Semantic Web technologies for defining, inferencing, storing and querying knowledge. With this insight it is possible to use these technologies to represent concepts that are relevant to the domain of an e-Research society. Many of these concepts have already been defined in existing schemas and ontologies. Dublin Core, SKOS, FOAF, SIOC, Creative Commons and OAI-ORE all provide conceptualizations for particular facets required to represent an e-Research society sharing content.

3.1.6.1 Dublin Core

The purpose the Dublin Core Metadata Initiative (DCMI) has been to define standards, vocabularies and practices for metadata (Powell et al., 2007). DCMI has an abstract model which builds on the work of RDF and RDF Schema (see section 3.1.1). This abstract model breaks down into three separate sub-models:

The Resource Model Resources can be described by properties that have either a literal or non-literal value.

The Description Set Model A resource may have one or more descriptions that make up that resource's description set. Each description may have one or more

statements, i.e. property-value pairs. The value of the statement may be a literal or a non-literal that could either be a vocabulary encoding scheme URI, a regular URI or string described by a vocabulary encoding scheme.

The Vocabulary Model Vocabularies have terms that may be classes, properties, vocabulary encoding schemes or syntax encoding schemes, i.e. classes of literals. Classes and properties may have sub-classes/sub-properties of each other and properties may have domains and ranges that can also be classes. A resource can then be an instance of a class or a member of a vocabulary encoding scheme.

DCMI provides a schema called DCMI Metadata Terms that can be specified as a RDF schema[14] (DCMI Usage Board, 2008). Table 3.1 shows the nine vocabulary encoding schemes and eleven syntax encoding schemes, (for languages, countries, geographic coordinates, time intervals, etc.), captured by this schema. DCMI Metadata Terms reuse

Vocabulary Encoding Schemes	Syntax Encoding Schemes
DCMI Types	DCMI Box
Dewey Decimal Classification	ISO 3166-1
IANA Media Types	ISO639-2
Library of Congress Classification	ISO 639-3
Library of Congress Subject Headings	DCMI Period
Medical Subject Headings	DCMI Point
National Library of Medicine Classification	RFC 1766
Getty Thesaurus of Geographic Names	RFC 3066
Universal Decimal Classification	RFC 4646
	IETF's URIs
	W3C Date and Time Formats

TABLE 3.1: Vocabulary and Syntax Encoding Schemes in DCMI Metadata Terms

the legacy properties from the DCMI Elements Set[15] and defines additional properties and classes. These properties cover standard bibliography fields and properties required in the publication process. The classes allow for new instances of non-literal values that some properties require, such as location, license document, file format, media type, linguistic system, etc. as well as any agents or classes of agent that may be involved in the publication process.

3.1.6.2 Simple Knowledge Organisation System (SKOS)

SKOS is designed to provide a means for representing various types of concept schemes in RDF (Isaac and Summers, 2009). Concepts schemes are things such as thesauri, classification schemes, taxonomies, folksonomies and controlled vocabularies. Figure 3.2 shows the main aspects of SKOS. SKOS's unit of currency is a *Concept* these must be

[14] `http://dublincore.org/2008/01/14/dcterms.rdf`
[15] `http://dublincore.org/2008/01/14/dcelements.rdf`

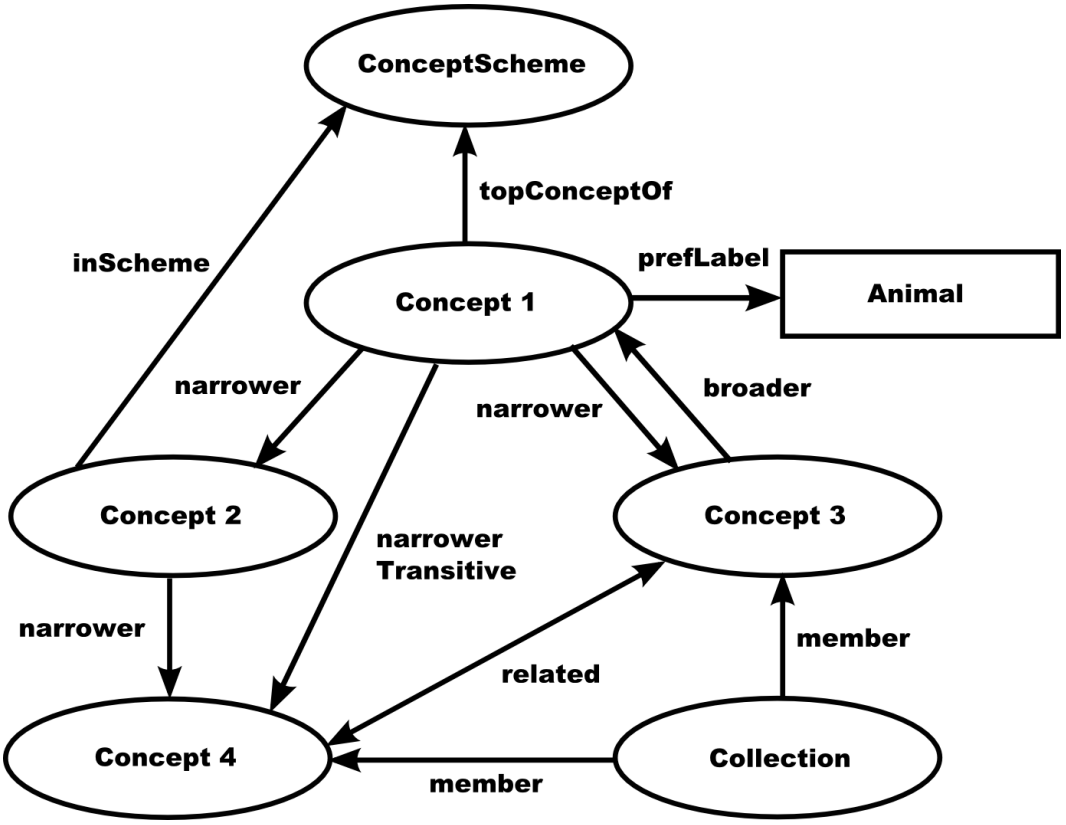


FIGURE 3.2: Example of SKOS Entities and Relationships

in a *ConceptScheme* but may also be a member of a *Collection*. If a concept is the most generic it should be related to a ConceptScheme as the *topConceptOf*. However, the interrelation between Concepts is the most important feature. More generic concepts have a *narrower* relationship to those more specific. Narrower's inverse property is *broader*, linking the specific to the generic. These two properties cannot be transitive, to allow *Concept 4* to be described as being *narrower* than *Concept 1*, because the hierarchy would be lost when inference is performed. Therefore, the transitive properties *narrowerTransitive* and *broaderTransitive* have been defined, allowing direct relationships from any *Concept* to all *Concepts* more specific or generic than it. If two *Concepts* are similar to each other but sit on different branches of the hierarchy, the *related* property can be used to link them together. To make *Concepts* human-readable, SKOS provides three types of label: preferred, alternative and hidden.

3.1.6.3 Friend of a Friend (FOAF)

The goal of the FOAF project has been to support people in creating FOAF profile documents[16] for themselves (Brickley and Miller, 2010). The main purpose of a FOAF profile document is to allow a person to specify a "basic expression of community membership." (Dumbill, 2002). These documents are managed by their owners, so the data is

[16] `http://users.ecs.soton.ac.uk/drn/foaf/me.rdf`

decentralised, an approach adopted by the Diaspora* social networking site (see section 2.2.3). They are designed so they can be aggregated into community directories. FOAF documents can be produced using a RDF vocabulary that is specified in the FOAF ontology[17] that can also be processed by RDFS tools. A number of social networking sites allow users to export FOAF profiles including LiveJournal and FriendFeed (FOAF project team, 2009).

FOAF allows the creation of a *Person* entity to represent the person for whom the FOAF profile document is being created. This provides a URI allowing the person to express their globally unique identity, which is essential before they can start associating information with themselves. This information can include their name, email address, phone number, various homepages, (e.g. workplace homepage, weblog, etc.), interests, age, location, publications, an image of the themselves, etc.

After defining a person's information, by using the *knows* property, a person can associate themselves with other people, providing "the first degree" of a social network, (see section 2.2.2). Although a FOAF profile is generally intended to allow the owner to describe information about themselves it also allows them to express information about other people as long as they know their URI. It is then possible to merge this sets of information to generate a more complete FOAF profile as part of community directories.

FOAF allows the specification of *Groups* that may have one or more people as *members* and *Organisations* to represent institutions, companies, societies, etc. Projects can also be defined for which a person may describe them as being a current or past project of theirs. Online communities mean that a person may have several accounts, one for each community. FOAF allows each of these accounts to be associated with the person through the *account* property.

All the entities described may have documents associated with them. FOAF provides the *page* property to represent this association. Inversely each *Document* has a *topic* property for every entity associated with it that can be automatically generated by using the ontology to perform OWL DL inference over the RDF. Finally it is useful to define information about the profile document itself, so that at a later date it is possible to disseminate where information came from.

3.1.6.4 Semantically-Interlinked Online Communities (SIOC)

The SIOC project aims to provide "the main concepts and properties required to describe information from online communities", including weblogs, messageboards and wikis (Berrueta et al., 2007). There is an urgent need for this because these technologies are starting to replace libraries and publishing as a means for keeping a community informed. In some respects SIOC is an extension of FOAF to allow greater detail to be

[17] `http://xmlns.com/foaf/spec/index.rdf`

specified about how a *Person* via an *OnlineAccount*, for which SIOC *User* is a subclass, interacts with documents. In the case of SIOC these are online collaborative documents generically referred to as *Containers*, which hold specific pieces of content called *Items* Figure 3.3 shows an example of the use of *Containers* and *Items* to represent a *Forum*

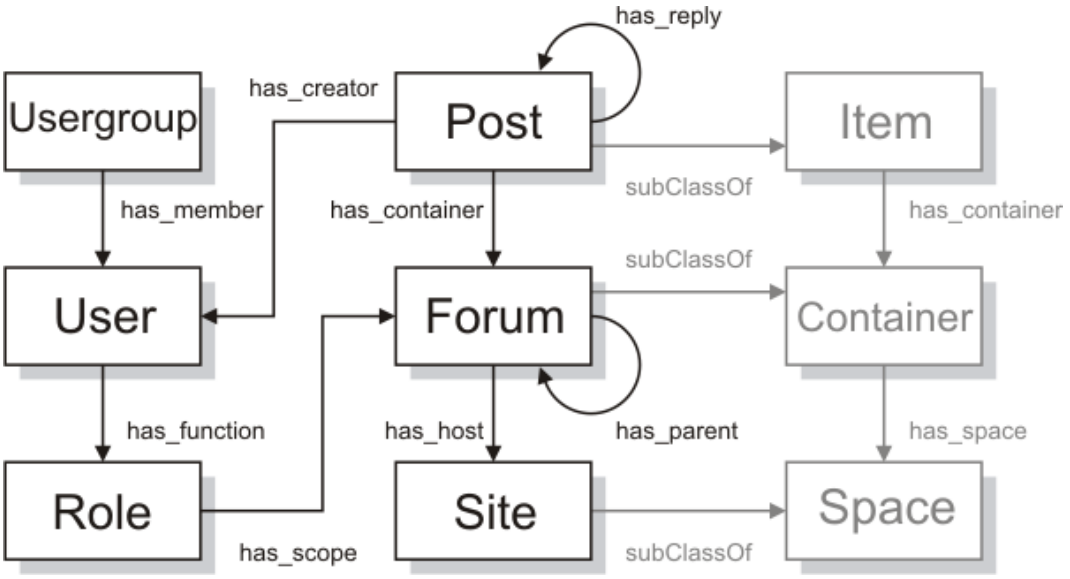


FIGURE 3.3: SIOC Overview (Due to `http://www.w3.org/Submission/2007/SUBM-sioc-spec-20070612/`)

with *Posts*. A *Forum* may have nested *Containers*, e.g. *Forums* or *Threads*, building a hierarchical structure. Each top-level Container can then be hosted within a *Space*, e.g. a Forum on a *Site*. Each Item can have various properties to define its position within a container, based on version, date or reply status. Basic metadata is also provided that can be used along with Dublin Core to describe the Item and manage the content it captures.

A *User* in SIOC may have many *Roles*, some of these are pre-defined in the ontology, such as *member_of* a *UserGroup*, *administrator_of* a *Site*, *subscriber_of* a Container *modifier_of* an *Item*, etc. Others can be specified as *Role* entities as a they are required by the particular community.

SIOC also has a number of extension modules:

Access [18] for assigning *Permissions* to *Roles* and *Statuses* to *Items*.

Types [19] defines different types of *Containers*, *Items*, *Forums* and *Posts*.

Services [20] allows *Services* that are provided by a *Site* to be associated with it.

[18] `http://rdfs.org/sioc/access`
[19] `http://rdfs.org/sioc/types`
[20] `http://rdfs.org/sioc/services`

3.1.6.5 Creative Commons

Creative Commons is a project that provides "free licenses and legal tools" to allow content creators to licence their work "so others can share, remix and use commercially" (Creative Commons, 2008). An RDF schema has been defined to specify the properties a license can entail[21]. The schema allows a piece of *Work* to be defined that has a *License* that comes under a *Jurisdiction*. Each *License* may have a number of properties

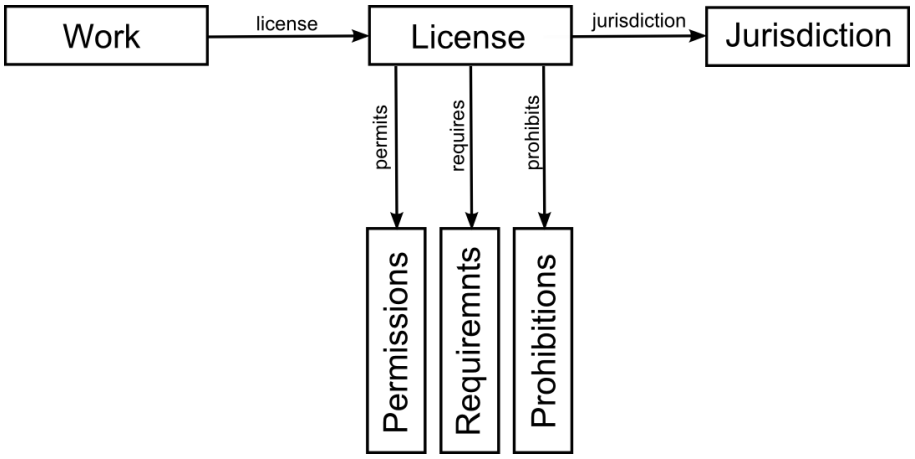


FIGURE 3.4: Creative Commons overview

assigned to it to describe what it *permits*, *requires* and *prohibits* as listed in table 3.2, as well as properties linking to the legal text for a license or stating the date the license became deprecated. A piece of *Work* may also have properties to reference additional

Permissions	Requirements	Prohibitions
Reproduction	Notice	Commercial Use
Distribution	Attribution	
Derivative Works	Share Alike	
High Income Nation Use	Source Code	
Sharing	Copyleft	
	Lesser Copyleft	

TABLE 3.2: Permissions, requirements, prohibitions available in Creative Commons licenses

permissions and alternative licenses as well as a name or URI for its creator.

3.1.6.6 Open Archives Initiative - Object Reuse and Exchange (OAI-ORE)

OAI-ORE "defines standards for the description and exchange of aggregations of Web resources" (Lagoze et al., 2008). This is required because many things in the physical world are often grouped together such as tracks on a CD or photos in an album. So

[21] `http://creativecommons.org/schema.rdf`

when it comes to referring to these electronically, e.g. bookmarks in a browser, photos on Flickr, etc., there needs to be a way of maintaining this aggregation.

OAI-ORE cites an example of a document in the arXiv[22] repository. Each document has an HTML web page that lists metadata for the document, which is described as a *human start page*. There are two core issues that ORE tries to address with this human start page. First, the use of its URI to represent the URI of the whole document, when in fact it is only a single representation of it. Second, the ambiguity to a machine agent of links that point at constituents of the document, e.g. different formats: PDF, PostScript, etc., navigational links for the archive that are outside the scope of the document. ORE proposes an explicit machine representation of what a documents contains to resolve this.

The ORE model has three main entities:

Aggregated Resource A resource that is part of the *Aggregation*, e.g. a PDF or Postscript of a paper.

Aggregation A specific collection of *Aggregated Resources* brought together because they all relate to a similar concept, e.g. formats, versions and supporting resources for a paper.

Resource Map A machine-readable representation of an *Aggregation* with its own URI, perhaps serialised in a number of formats including , RDF/XML and Atom XML (Adida and Birkbeck, 2008), (Nottingham and Sayre, 2005).

Figure 3.5 demonstrates how these entities interact. All of these three entities contain

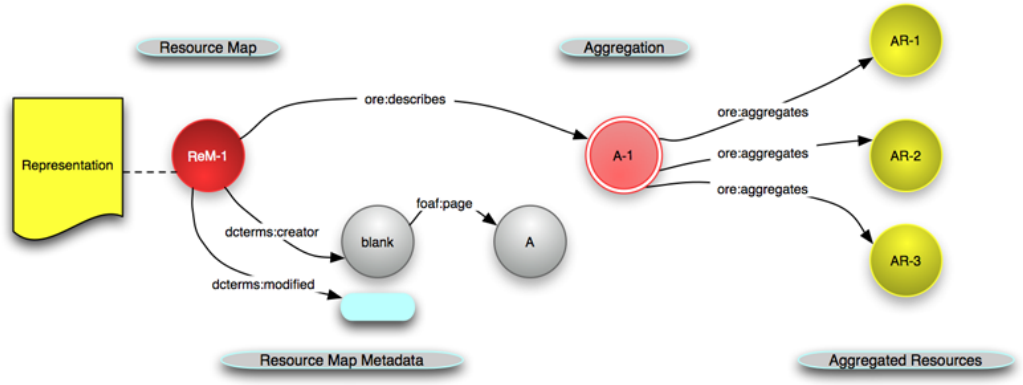


FIGURE 3.5: Resource Map and Aggregation in ORE (Due to `http://www.openarchives.org/ore/1.0/primer.html`)

Dublin Core or other appropriate metadata to describe themselves. Table 3.3 gives an example of the type of metadata each might have. It is important to note the difference

[22] `http://arxiv.org/`

between the creator of the *Aggregation*, that in the case of a paper would be its authors, and the creator of the *Resource Map*, that would be the repository that holds the paper. The example of an *Aggregation* representing a grouping of different formats and versions

Aggregated Resource	Aggregation	Resource Map
format	creator	date created
title	title	date modified
language		rights (license)
type		creator

TABLE 3.3: Metadata for ORE Entities

of a paper has a very specific central concept. ORE can be used to build aggregations of more general concepts such as a set of bookmarks. In this case *Aggregated Resources* may need to be described in the context of the *Aggregation*. ORE describes these contexts as *Proxies*. These can capture metadata, such as when the *Aggregated Resources* was added to the *Aggregation*, its position relative to other *Aggregated Resources*, e.g. chapters in a book, its title or any other piece of metadata in the context of the *Aggregation*.

3.1.6.7 Open Annotations Collaboration

The Open Annotations Collaboration (OAC) data model allows annotations to be assigned to resources (Sanderson and Van de Sompel, 2010). It draws significantly from the Annotea model (Kahan et al., 2002). A basic OAC *Annotation* uses two properties *hasBody* and *hasTarget*. *hasBody* links to the content of the annotation, this may be a defined through a non-resolvable URI, i.e. a URN, in the case of a tag or comment, or a URI such as for a review. *hasTarget* links to the resource being annotated. These two relationships imply a third *annotates* relationship between the *Body* and *Target* objects. This relationship can be explicitly defined or the following SWRL rule could be defined, which when applied will generate the *annotates* relationships.

$$oac : hasBody(?a, ?b) \wedge oac : hasTarget(?a, ?t) \Rightarrow oac : annotates(?b, ?t) \qquad (3.2)$$

An OAC *Annotation* may have further properties taken from existing schemas and ontologies such as Dublin Core's *title*, *creator* and *created*. *Annotations* themselves may be nested such as for a reply to a comment. When an *Annotation*'s *hasTarget* relation is to another *Annotation* then the class can be inferred as a *Reply*. Another feature of OAC is the ability to have *Annotations* that apply the same *Body* object to multiple *Targets*.

Being able to constrain what an *Annotation* covers is important. A *Constraint* can be applied to both the *Target* and the *Body* of an *Annotation*. A *Constraint* can be a URN, so that a string description can be captured locally or it can point to a resource. Another alternative is to define multiple predicates to describe the *Constraint* in greater

detail. An example of a *Target Constraint* might be describing the co-ordinates that enclose the area of the image that is being annotated. For a *Body Constraint* this might be that only one section of a review is about the *Target*.

Time Constraints are a special form of *Constraint* and use the *when* predicate. They may be applied to an *Annotation*, e.g. to clarify that a comment is about a web page as seen on a particular day. They also may be attached to the *Target* and *Body* simultaneously. This is necessary where the *Body* makes a relative time reference, e.g. "Yesterday's main BBC news headline was interesting" will have a *Target* with a *when* property one day earlier than that of the *Body*.

One of the major considerations in the design of the OAC model is how it conforms to Linked Data principles. As by its very nature it will commonly be used to link together resources from different providers.

3.1.7 Linked Data

"Linked Data refers to the set of best practices for publishing and connecting structured data on the Web" (Bizer et al., 2009). Linked Data sets out four 'principles' for publishing data that will facilitate a "single global data space":

1. Anything being described must have its own URI.
2. These URIs should be resolvable over HTTP to retrieve information about that particular thing.
3. The information must be useful and provided in a standard format, e.g. RDF, SPARQL, etc.
4. Where appropriate this information should link to other URIs to support discovery.

An example of the implementation of these principles is demonstrated by the Linking Open Data project[23] that has linked together datasets from many different projects from varying fields with around 142 million links as of May 2009.

From an application perspective, it is common to want to publish a human-readable form, i.e. an HTML web page, as well as the data itself in a machine-readable format such as RDF/XML. However both of these are referencing to the same thing that has a single URI, referred to as the *non-information resource URI* (Bizer et al., 2007). Publishing Linked Data requires this URI to be resolvable to a standard machine-readable format, but the requester may want the human-readable form. There are a number of ways to resolve this issue. One is to embed the RDF as RDFa in the web page. However, the

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

preferred way for Linked Data is to dereference the URIs. There are two ways to do this, Hash URIs or 303 redirects. Hash URIs keep data for multiple resources in the same file so are generally not suitable for some Linked Data if there is a large number of resources and/or the these resources are highly dynamic. 303 redirects work by performing content negotiation. The server looks at the HTTP request header for an *Accept* parameter to determine whether to redirect to an RDF/XML or HTML representation. This does put some extra load on the server processing the request, as they have to handle two requests rather than one (see Figure 3.6) but it provides a much more flexible mechanism than Hash URIs.

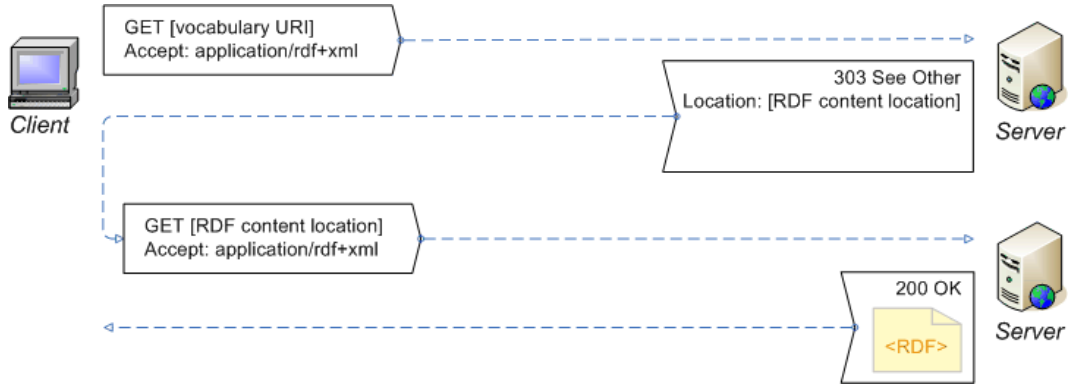


FIGURE 3.6: Linked Data Content Negotiation (Due to `http://www4.wiwiss.fu-berlin.de/bizer/pub/LinkedDataTutorial/20070727/`)

3.1.7.1 The DBPedia Project

One of the largest dataset in the Linked Open Data project is that of the DBPedia project[24]. The DBPedia dataset is generated by parsing Wikipedia's[25] pages. Although these do not represent data using SW technologies, they can capture a large amount of structured data just through HTML markup and linking and wiki categorisation and templates, amongst other things (Auer et al., 2007). By parsing this data it is possible to generate significant numbers of triples. Beyond extracting this data DBPedia has tried to capture the structure of information presented by Wikipedia with their own ontology[26]. This ontology provides a class hierarchy, DBPedia's own properties and URIs for instances such as places, people, species, organizations, etc. As DBPedia is one of the most comprehensive and well-known resources for 'instance' URIs, it is linked to by many external datasets.

[24] `http://dbpedia.org/`
[25] `http://www.wikipedia.org`
[26] `http://wiki.dbpedia.org/Ontology`

3.1.7.2 Vocabulary of Interlinked Datasets (VoID)

There can often be lots of elements to the Linked Data published by a project and there needs to be a means to discover all of these. The vocabulary of Interlinked Datasets (VoID) ontology allows a project to define a file that contain all the salient information about the data they are publishing, i.e. their *Dataset*, and how they are being published (Zhao et al., 2008). This includes the vocabularies used, i.e. RDF schemas and OWL ontologies, data dumps of the RDF, querying interfaces such as SPARQL and URI lookup endpoints, as well as examples resources, statistical items, e.g. number of triples and URI regular expression patterns to describe the Dataset.

Beyond describing the Dataset, VoID also allows the definition of links, i.e. RDF triples, between Datasets called *Linksets*. These are necessary to support navigation in a Linked Data world. A Linkset can either be a subset of one of the Datasets being linked (classic Linked Open Data (LOD)) or its own Dataset, i.e. a third-party Linkset. Each Linkset must define the two Datasets it links. It may also specify the predicate that links the Dataset. Often these predicates are bi-directional, i.e. symmetric properties, such as OWL's *sameAs*, RDFS's *seeAlso* and FOAF's *knows*. However, if these predicates are uni-directional (e.g. FOAF's *based_near*) the Linkset must define which Dataset is the subject and which is the object. Sometimes it is feasible to build these Linksets manually, like when the Linkset is a subset of one of the Datasets but commonly this process is time consuming and/or difficult and therefore some automated assistance such as co-reference resolution is required.

3.1.7.3 Co-reference Resolution

In this context, Co-reference resolution is the concept of determining that two URIs actually represent the same thing (Glaser et al., 2009). Tools exist for automating this process by comparing the triples for two different subject URIs and if a threshold number and percentage is exceeded then the two URIs can be considered to be referencing the same thing. Further sophistication could be added to analyse text in the literals to find near matches or inferencing could be performed on the Datasets to see if the additional triples help to meet the thresholds required.

Sindice[27] (the Semantic Web Index) allows for searching on keywords, URIs and property-value pairs as well as more advanced searches, where triple patterns are matched using the wildcard character (*) to represent unknown subjects, objects or predicates. Once a *Linkset* has been defined a tool such as sameAs[28] allows a Linked Data URI to be looked up to see if it is the "same as" any other URIs. Sindice and sameAs both rely

[27] `http://sindice.com/`
[28] `http://sameas.org/`

on indexing of all the Linked Data in a central store making the ability of a triplestore application to store huge amounts of triples ever more important.

3.2 Adding Semantics to myExperiment

3.2.1 The myExperiment Ontology

The purpose of an ontology is to specify the conceptualization of a model, in this case the myExperiment model as described in section 2.3.1. Although the data for this model is taken from a MySQL database it would not be appropriate to use the structure of this database to generate the ontology despite a number of tools existing, (e.g. RDBtoOnto (Cerbah, 2008), DB2OWL (Cullot et al., 2008)). There are several reasons for this. First, in Ruby-on-Rails' MVC architecture the underlying database is not the model it is just a data structure that the model can be built on. The model itself is defined within the ruby code of the model files and this would be lost by just converting the database. Second, the main purpose of the model is to support the myExperiment website, therefore certain database tables exist solely for this purpose, which would not be useful to conceptualize within an ontology. Furthermore, the organisation of the model is intended to best support this website and make it as efficient as possible. This may not coincide with the most succinct, logical representation. Finally, because of myExperiment's sharing model, some of the data captured by the model should not be published. Although this does not dramatically affect the model's structure, it reinforces the need for careful consideration of the ontology's design to ensure only appropriate data is made publicly available and why a tool to convert the database schema into an ontology would be inappropriate.

3.2.1.1 Reusing Ontologies

As described in section 3.1.6 various schemas and ontologies already exist that are capable of representing an e-Research society at least in part. The myExperiment ontology reuses classes and properties from Dublin Core, FOAF, SIOC, SKOS, Creative Commons, OAI-ORE and DBPedia. Table 3.4 indexes all the classes and properties reused by myExperiment and how and where they are used. Sections 3.2.1.2 - 3.2.1.4 explain in greater detail how these properties have been incorporated.

3.2.1.2 Simple Network Access Rights Management (SNARM) Ontology

myExperiment's sharing model is a key feature in making myExperiment unique. It is essential to provide an extensible specification for representing this so that each contribution in myExperiment can define a *Policy* for its access rights. Simple Network Access Rights Management (SNARM) is a simple ontology that allows additive policies to be defined (see Appendix A.2.1). Figure 3.7 provides an overview of this ontology. A SNARM *Policy* contains one or more *Access* entities to describe the access available

to the contribution that uses it. An *Access* defines the type of access allowed. In the case of myExperiment there are three types: *View*, *Download* and *Edit*. Some *Access* entities are universal, whereas a subset (*RestrictedAccess* entities) are restricted to an *Accesser*. Critically, an *Accesser* may represent one of many different things, a single *User*, a *Group* or a virtual group. In the case of myExperiment a virtual group could be the contributing user's friends or all of the groups that he or she belongs to. What is important is that this virtual group can be resolved to determine the underlying users. In the case of friends and groups on myExperiment, SWRL rules (see section 3.1.4) could be defined to achieve this. The two *?virtualgroup* unknowns can then be used as

Reused Class/Property	How Reused	Reusing Module	Reusing Class/Property
Dublin Core			
title	owl:onProperty	*multiple*	*multiple*
description	owl:onProperty	*multiple*	*multiple*
created	owl:onProperty	Base	Actor, Submission
modified	owl:onProperty	*multiple*	*multiple*
identifier	owl:onProperty	Base, Components	License, Dataflow
Agent	rdfs:subClassOf	Base	Actor
RightsStatement	rdfs:subClassOf	SNARM	Policy
BibliographicCitation	rdfs:subClassOf	Annotations	Citation
hasVersion	rdfs:subPropertyOf	Base	has-version
isVersionOf	owl:onProperty	Base	Version
Friend Of A Friend (FOAF)			
name	owl:onProperty	Base	User
mbox	owl:onProperty	Base	User
mbox	rdfs:subPropertyOf	Base	email
homepage	owl:onProperty	Base	User
based_near	owl:onProperty	Base	User
Semantically-Interlinked Online Communities (SIOC)			
name	owl:onProperty	Base	Actor
avatar	owl:onProperty	Base	User
has_owner	owl:onProperty	Base, Packs	*multiple*
has_owner	rdfs:subPropertyOf	Base	has-annotator, has-announcer
has_member	owl:onProperty	Base	Group
User	owl:equivalentClass	Base	User
UserGroup	owl:equivalentClass	Base	Group
Item	owl:equivalentClass	Base	Submission
Simple Knowledge Organisation System (SKOS)			
Concept	rdfs:subClassOf	Annotations	Tag
prefLabel	owl:onProperty	Annotations	Tag
Creative Commons			
permits	owl:onProperty	Base	License
prohibits	owl:onProperty	Base	License
requires	owl:onProperty	Base	License
License	rdfs:subClassOf	Base	License
Open Archives Initiative - Object Reuse and Exchange (OAI-ORE)			
aggregates	owl:onProperty	Packs	Pack
isDescribedBy	owl:onProperty	Packs	Pack
proxyFor	owl:onProperty	Packs	Entry
proxyIn	owl:onProperty	Packs	Entry
Proxy	rdfs:subClassOf	Packs	Entry
DBPedia			
residence	owl:onProperty	Base	User

TABLE 3.4: Classes and Properties Reused by the myExperiment Ontology

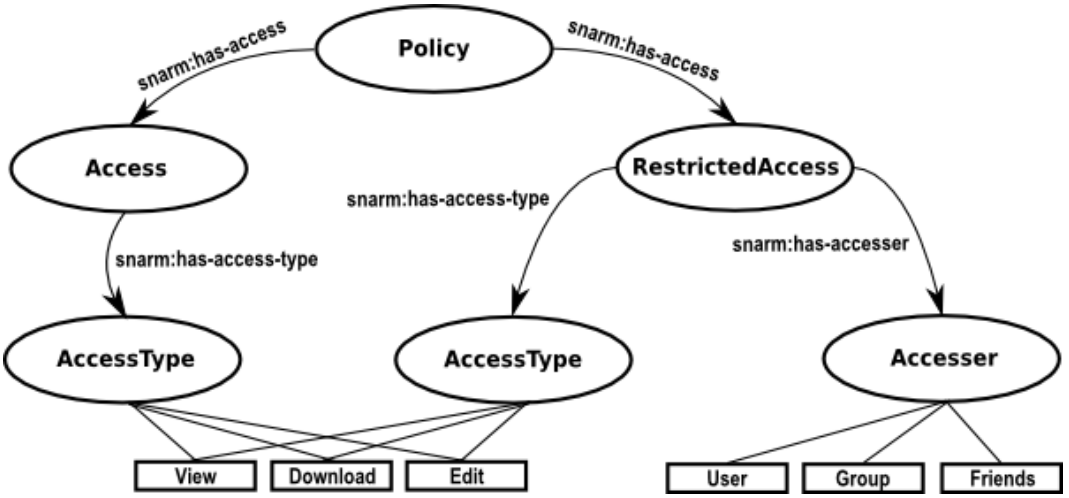


FIGURE 3.7: Simple Network Access Rights Management Ontology Overview

these virtual groups in a policy.

$$has\_owner(?contribution, ?user) \wedge knows(?user, ?friend) \Rightarrow has\_member(?virtualgroup, ?friend)$$

$$has\_owner(?contribution, ?user) \wedge has\_member(?group, ?user) \wedge has\_member(?group, ?member) \Rightarrow has\_member(?virtualgroup, ?member)$$

SNARM is comparable to SIOC's *Access* module (see section 3.1.6.4). Both share the goal of defining who can perform certain actions on something. However, the means for organizing these permissions differs. SIOC assigns *Permissions*, e.g. view, edit, etc., to *Roles*, e.g. a forum moderator, that are assigned to one or more *UserAccounts*. As each *Role* has a scope, e.g. a forum or a site, only entities within that scope are subject to that permission. SNARM does not have a concept of scope, this is due to the need to be able to assign different permissions to each individual contribution.

3.2.1.3 Base Module

Section 2.3.1 helped to determine the three main features of the Base module, namely content management, social networking and object annotation. There are various specifics for how these features are applied in myExperiment but there is an underlying generic model that needs to have a logical representation. The Base ontology module (see Appendix A.2.2) provides this representation.

Figure 3.8 highlights how the *User* is the central focus of myExperiment. They are required for content management through ownership of contributions. They are the

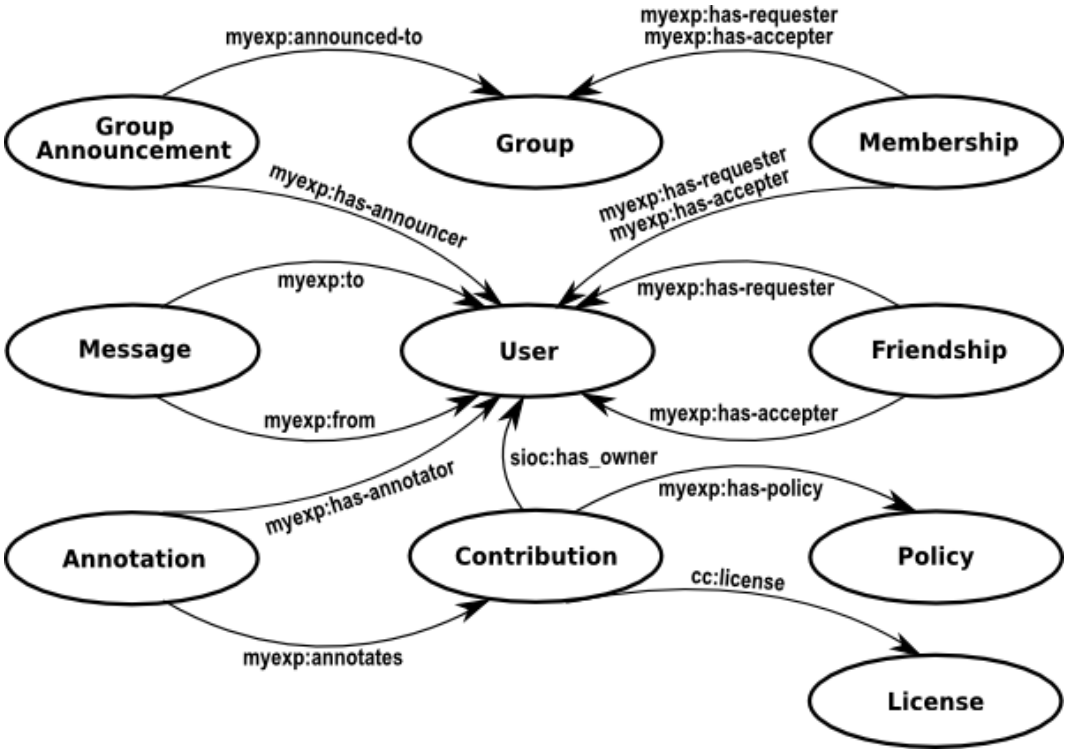


FIGURE 3.8: Overview of myExperiment's Base Module

building bricks of the social network, having friendships with other users, memberships to groups and by communicating messages and announcements. Users are also responsible for all types of object annotation. By reifying the annotation of objects by users as entities in themselves conforms with a tripartite ontology model encompassing users, objects and annotations, implemented as three bipartite graphs (i.e. users and objects, users and annotations, objects and annotations). This has been recognised as a means of supporting structured community-based annotations, such as folksonomies (Mika, 2005). This model also conforms in part with basic *Annotations* of OAC, (see section 3.1.6.7), that was defined after the myExperiment ontology was created. It should be possible for all myExperiment *Annotations* to be defined as OAC *Annotations*. Some thought will be required to fully achieve this alignment, in particular how to map string/integer literals for *Comments*, *Ratings*, etc. to URNs. Also how existing composite *Annotations* such as *Reviews* that have a *title* and *body* may be moved so that the *Review* resource is a separate OA *Body* object from the *Annotation* applying it to a *Contribution*.

Figure 3.8 also shows the importance of policies and licenses for contributions. The *Policy* for a *Contribution* is provided by the SNARM ontology, (see section 3.2.1.2), whereas the *License* is provided by extending the Creative Common's *License* class to allow for additional properties to be mandated. SIOC's *has_owner* is such a property; it is used to describe who is responsible for maintaining the metadata for the license on myExperiment or whatever website the license is being provided for use on.

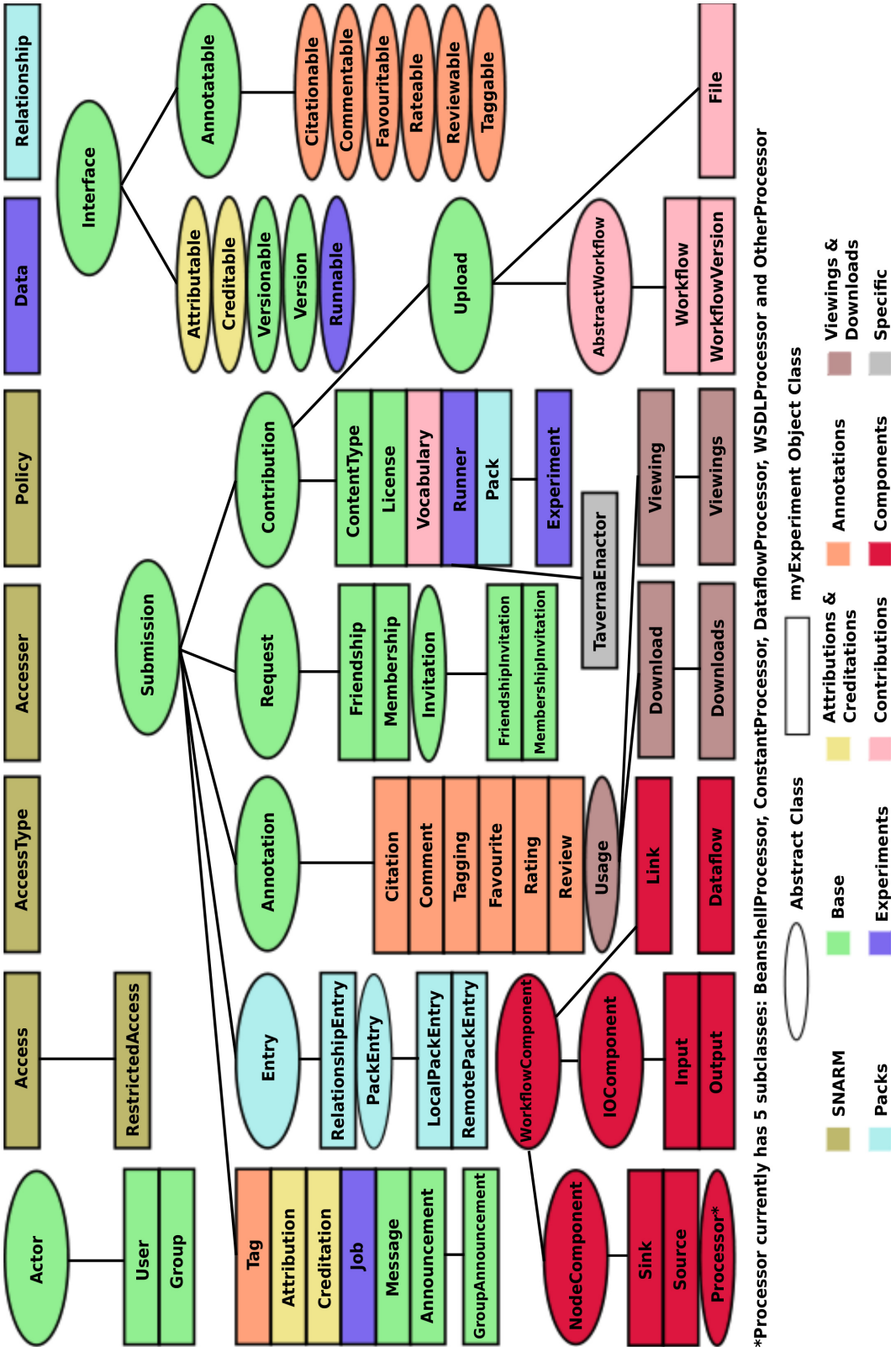


FIGURE 3.9: Class Hierarchy for myExperiment Ontology

As Figure 3.9 demonstrates, the Base module has a number of abstract classes. *User* and *Group* are subclasses of *Actor*. This is because some actions and roles in myExperiment

are ambiguous for example the requester and accepter of a *Membership* can either be the *User* or the *Group* depending on who initiated the *Request*. To align with the SIOC ontology, *Contributions*, *Annotations* and *Requests* are all subclasses of *Submission*, so this can be made equivalent to a SIOC *Item* that defines a unit of content.

Various classes in myExperiment have specific behaviours, e.g. what annotations can be made about them, whether they can be versioned, etc. These behaviours may mandate an instance of a class being the subject or object of more than one property. There is not a hierarchical model for assigning these behaviours / mandate properties, so a further abstract class is needed, this is called the *Interface* class. This *Interface* class can be extended to define the mandated properties of a behaviour, e.g. *Annotatable*: Something that can be the subject of annotations; *Versionable*: Something that may be versioned. Then by using RDFS's *subClassOf* property these behaviours can be assigned to individual classes in a consistent manner.

3.2.1.4 Extension Modules

The Base module has been designed to allow additional modules to be added to it to extend the ontology's specification to more specific concepts. Figure 3.10 diagrams how these modules extend the Base module.

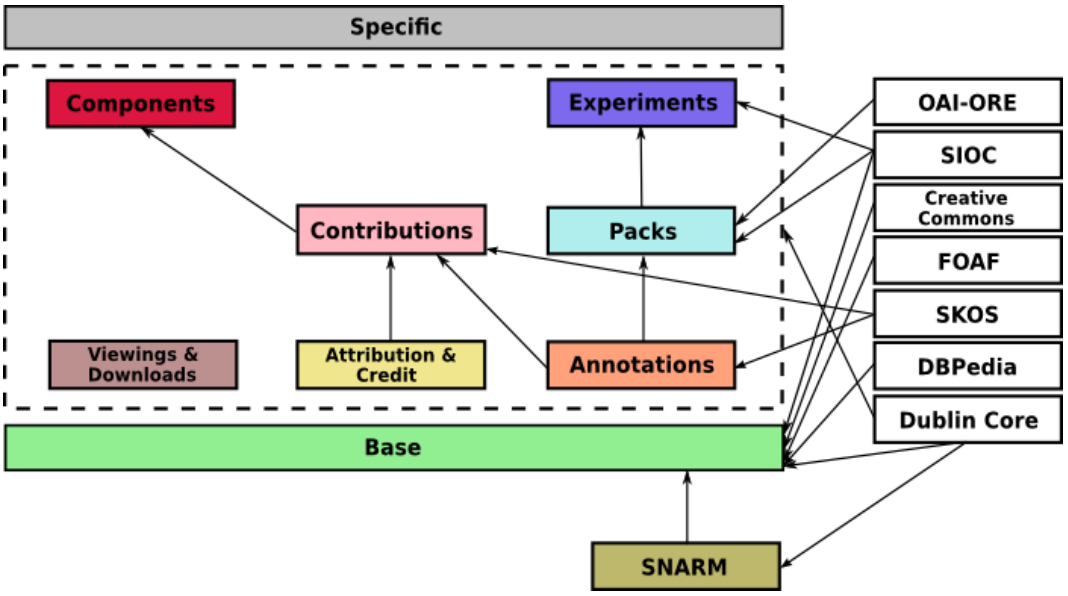


FIGURE 3.10: Ontology Modules Architecture

Section 2.3.1 observed the user's requirement for a means to credit other users for the creation of a contribution or to attribute the design of a contribution to one uploaded to myExperiment at an earlier date. The Attributions and Creditation module (see Appendix A.2.3) provides interfaces that can be used to define which contributions can

be attributed and those for which credit can be paid. It also provides *Attribution* and *Creditation* entities to encapsulate metadata about these assignments.

The Viewings and Downloads module defines usage statistics for contributions. This includes properties for contributions to specify their viewing and download counts as well as classes that can be instantiated to specify each incidence. To capture a *Viewing* or *Download* instance for each incidence would often be excessive, especially as many of these are generated by web crawlers. Therefore *Viewings* and *Downloads* are defined that combine all instances of the same type with an accompanying *count* property. Where "same type" means they share the same user, user agent, e.g. Firefox, Internet Explorer, etc. and contribution.

The *Annotations* module defines specific annotations as well as interfaces that can be applied to contributions that use them (See table 3.5).

Annotation	Interface
Citation	Citationable
Comment	Commentable
Favourite	Favouritable
Rating	Rateable
Review	Reviewable
Tagging	Taggable

TABLE 3.5: Annotations and their Interfaces

The Contributions module defines contributions used within myExperiment, namely *Workflows*, *Files* and *Vocabularies*, along with specific properties associated with these contributions. Each of these contributions are subclasses of Interface classes to specify the behaviours they exhibit (see table 3.6). *Workflows* are *Versionable* and have *WorkflowVersions* that share similar properties to *Workflows*. Having an *AbstractWorkflow* superclass to define the properties required saves having to specify them twice.

The Packs module defines the *Pack* contribution and the *PackEntries* that it contains. A *Pack* is essentially the same as an OAI-ORE *Aggregation* but rather than just making it a subclass, the module places restrictions on it to have OAI-ORE *aggregates* and *isDescribedBy* properties, which is what logically makes a class an *Aggregation*. *PackEntries* are a subclass of OAI-ORE *Proxy*, as they allow the entity that each entry represents to be described in the context of the *Pack* in which they reside. As *Packs* can contain both references to entities within myExperiment as well as remote URLs, either *LocalPackEntry* or *RemotePackEntry* should be used respectively, as they support the appropriate properties for each type.

The *Experiments* module defines the *Experiment* contribution that is a specialisation of a *Pack*, specifically for aggregating sets of *Jobs*. A *Job* is the execution of the *Runnable* entity such as a *Workflow* on a *Runner*. Inputs and outputs for a *Job* are defined as

Contribution / Behaviour	Workflow	Workflow Version	File	Pack	Experiment
Attributable	X	X	X		
Citationable	X	X			
Commentable	X	X	X	X	X
Creditable	X	X	X		
Favouritable	X	X	X	X	X
Rateable	X	X	X		
Reviewable	X	X			
Taggable	X	X	X	X	X
Versionable	X				
Version		X			

TABLE 3.6: Behaviours of Contributions

Data entities that may contain a text string and / or a URI to a file that contains the input / output data. These are then collated with provenance data about the *Job*, i.e. current status and submission, started, completed and last status change time.

The *Annotations*, *Contributions*, *Packs* and *Experiments* modules are comparable to SIOC's *Type* module (see section 3.1.6.4). The classes in both are defined as subclasses of SIOC's *Item* class. The main difference is that to accurately represent the myExperiment model these types need to be described more specifically using a complex class hierarchy and the use of OWL *Restriction* sub-classes. Whereas SIOC's *Type* module just defines each type very basically, i.e. whether it is a sub-class of *Container*, *Item*, *Forum* or *Post* and then zero or more RDFS *seeAlso* relations to similar types from other ontologies or schemas.

The *Components* module allows the inner workings of a Workflow to be defined. As myExperiment's original user group was that of Taverna users, this module is an abstraction of the Taverna workflow model (see section 2.1.2.3) introducing generalisations where appropriate so that only minor extensions should be required to support other workflow models. There are six main types of components that are encapsulated within a *Dataflow* that a Workflow can execute:

Source An initial input node for a workflow.

Sink A final output node for a workflow.

Processor A node that performs some processing.

Input A data entry point for a node.

Output A data exit point for a node.

Link Links an Output to an Input.

A Processor may have both Inputs and Outputs but a Source can only have Outputs and a Sink only Inputs. There are currently five types of processor defined in the Components module (see Appendix A.2.9):

BeanshellProcessor Runs a local Beanshell script that is stored within the workflow.

ConstantProcessor Prints a constant value as an output.

DataflowProcessor Runs a nested workflow.

WSDLProcessor Calls a remote Web Services Description Language (WSDL) service.

OtherProcessor Is a catch all for all other types of processor. The *processor-type* property allows a string to be defined for the specific type of processor.

Figure 3.11 shows how all these components can fit together to build a workflow. As can be seen by comparing to Figure 2.1 it is not too dissimilar to the structure of a Taverna workflow.

3.2.1.5 Specific Module

The Specific module is used to pull together the Base module and all the extension modules using OWL's *imports* property. It also contains classes and instances that are highly specific to myExperiment such as the *TavernaEnactor* class and particular *Accesses*, *AccessTypes* and *Accessers* for use in SNARM *Policies*, as shown in table 3.7. The module also specifies an anonymous user instance. This allows any property that

		Accesser: Public	Accesser: Friends
AccessType	View	PublicView	FriendsView
	Download	PublicDownload	FriendsDownload
	Edit	PublicEdit	FriendsEdit

Table 3.7: Accessers, AccessTypes and their associated Accesses

links to a user, e.g. SIOC's *has_owner*, to explicitly specify that the user is unknown, a necessity when using the Open World Assumption (OWA). In particular this is used

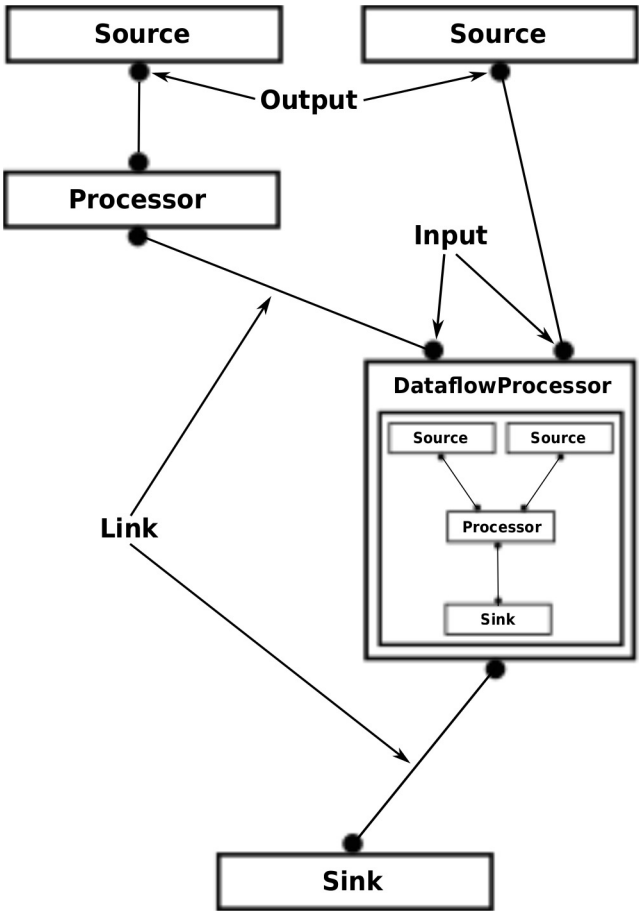


FIGURE 3.11: Generic Workflow with Components

to represent usage statistics for users that are not logged in. The last function of the Specific module is to retroactively assign certain conditions applicable to myExperiment. Namely, *Group* being a subclass of *Commentable* and *Taggable*, as myExperiment allows groups to have both comments and tags and *AbstractWorkflow* being a subclass of *Runnable* because myExperiment allows *Workflows* to be run as a *Job* on a *Runner*.

3.2.1.6 Promoting Reuse

Unless an ontology is being defined for a very specific task, it is sensible to design it in a reusable way so that others can either use or extend upon it to save reinvention. There have been several rules adopted whilst designing the myExperiment ontology to encourage reuse.

First, the ontology has attempted to abstract similar concepts. Figure 3.9 demonstrates how only ten classes are not the subclass of another, and four of these are from the SNARM ontology module. This facilitates assigning additional properties and constraints as the hierarchy is descended, moving from the generic to the specific. For

someone reusing the ontology, these abstracted classes can be extended to create their own classes, such as new types of annotation or contribution.

The advent of the *Interface* class for which subclasses can be created to define particular behaviours also encourages reuse. It both provides a means for someone extending the ontology to assign behaviours to their own classes and create further interface classes to define their own behaviours.

The myExperiment ontology is an extensive specification. Someone reusing it may not be concerned with certain aspects. By having a modular design they can choose to only reuse the modules they need. Consideration was given to what is core to the myExperiment model and thus should be defined in the Base module. This left the rest of the specification to be broken up into its different aspects to create the extended modules. All these require the Base module but care was taken to limit the reliance upon each other. Figure 3.10 shows how there are only five dependencies between the seven extension modules.

Finally, by making use of more generic ontologies and schemas, i.e. Dublin Core, FOAF, SIOC, Creative Commons, OAI-ORE and DBPedia, helps to set the myExperiment ontology in some context. This allows someone reusing or extending the ontology to better understand the purpose of the ontology, as they can more easily compare the classes and properties to those in other ontologies. To further improve on this SWRL rules can be used to imply properties from more generic ontologies using specific patterns of triples, (see table 3.8).

Friendship(?friendship) ∧ accepted-at(?friendship,?accepted_at) ∧ has-requester(?friendship,?requester) ∧ has-accepter(?friendship,?accepter)	⇒	foaf:knows(?requester,?accepter) ∧ foaf:knows(?accepter,?requester)
Membership(?membership) ∧ accepted-at(?membership,?accepted_at) ∧ has-requester(?membership,?requester) ∧ User(?requester) ∧ has-accepter(?membership,?accepter)	⇒	sioc:has_member(?accepter,?requester)
Membership(?membership) ∧ accepted-at(?membership,?accepted_at) ∧ has-requester(?membership,?requester) ∧ has-accepter(?membership,?accepter) ∧ User(?accepter)	⇒	sioc:has_member(?requester,?accepter)

Table 3.8: SWRL Rules for Friendships and Memberships

These rules generate FOAF *knows* and SIOC *has_member* from *Friendship* and *Membership* instances. An additional benefit of maximising the number of reused properties is that the RDF generated for myExperiment can be understood and compared with other RDF without needing to refer to the ontology. This is particularly useful in the

Linked Data world for co-reference resolution (see section 3.1.7.3). When two subjects share a number of predicate-object pairs, a degree of certainty can be apportioned to them being equivalent. As the number of these pairs increases this certainty grows.

3.2.1.7 The Evolution and Evaluation of the myExperiment Ontology

The myExperiment ontology was first conceived in 2008, by which time the myExperiment data model had sufficiently stabilised after the initial design of the myExperiment website, to allow users to make friends, join groups, upload workflows and files and then rate, review, comment and tag them. At this stage myExperiment also had forums and blogs for users to discuss and write their own opinions about particular topics. These were included in the original ontology but as it became clear that the users did not find a particular use for these features, they were disabled in the myExperiment website and consequently removed from the ontology.

The original myExperiment ontology used both the structure of the MySQL database and the model of the myExperiment as it is described as part of the Ruby-on-Rails MVC architecture to aid its design. For the reasons discussed at the start of section 3.2.1, it was not appropriate to use an automated tool to generate the ontology. Instead this process was performed manually. The advantage of taking such an approach was that very careful consideration could be given to how the abstract the data model and capture the generic features and entities that underlie myExperiment and more generally e-Research societies. The development of the ontology and RDF for myExperiment entities that it describes was an iterative process; using the insight gained from producing a script to generate the RDF, to inform how the ontology could be further abstracted and made more concise and consistent.

After discussion with some of the members of the W3C HealthCare and Life Sciences Scientific Discourse subtask group and analysis of the model for the SWAN ontology[29], it became clear that the myExperiment ontology would benefit from being modularised and this was undertaken towards then end of 2008. As discussed in section 3.2.1.6 the extensible nature of a modularised ontology helps promote reuse of the ontology, allowing other projects to choose to just reuse generic modules and produce their own specific modules. Another motivation for this modularisation was to aid alignment with other ontologies, as is discussed later in section 4.2.2.5.

Both prior to and after the modularisation of the ontology, myExperiment's data model was evolving with the addition of Packs, Announcements, Licenses, Attributions, Creditations and Workflow Components. As each of these were included in the data model, how they could be added to ontology in a concise and consistent manner had to be

[29] http://swan.mindinformatics.org/ontology.html

considered. Workflow Components is an example of an interaction with a user of myExperiment's RDF data to provide a representation of the components of a Workflow so that the user could make use of this in building a workflow recommender system. Other requests have been made by users, so they can make greater use of RDF data provided by myExperiment.

One example of a such a request was when some users wanted to visualize myExperiment's RDF in a Semantic Web browser such as Zitgist[30]. These browsers allow users to navigate the RDF via the resources referenced by object properties. At the time, Annotations, Friendships and Memberships referenced the Contribution or User but the Contribution or User did not have the inverse relationships. This meant that the amount of navigation from the User or Contribution RDF was limited and the only way to find all the Annotations, Friendships or Memberships for Contributions or Users was through a SPARQL query. Introducing properties such as has-tagging, has-friendship and has-membership increased the ability to navigate around myExperiment's RDF in a Semantic Web browser.

Since the myExperiment ontology has been modularised a change log[31] for alterations to the ontology has been kept. Appendix B.3 has a listing for this change log. It contains references to some of modifications made to support users as just described, as well as other modifications such as the addition of DBPedia residence predicated triples that is described in greater detail in section 3.2.3. The main purpose of change log is to inform those performing SPARQL queries across the data, of modifications that might effect the results they get. Although changes that are liable to significantly affect SPARQL queries are generally avoided, it is sometimes necessary if an inconsistency is identified or elements need to be modified because the underlying data model has changed.

Like myExperiment itself the ontology is considered to be in perpetual beta with an agile approach to its evolution, allowing new features added the myExperiment data model to be incorporated into the ontology. Feedback from users whether it be to include additional features, modify the ontology to aid alignment with a relevant ontology or just to report an inconsistency are all evaluated and changes to the ontology are made as appropriate.

3.2.1.8 Comparisons to Drupal RDF

In section 2.3.3, the myExperiment project was compared to Drupal. The Drupal model has also been mapped into RDF. Figure 3.12 shows Drupal's RDF model (Corlosquet, 2008). The Drupal model is intended to be more generic than myExperiment and this is reflected in its RDF schema. It is therefore more appropriate to compare it with

[30] http://dataviewer.zitgist.com
[31] http://rdf.myexperiment.org/ontologies/CHANGELOG

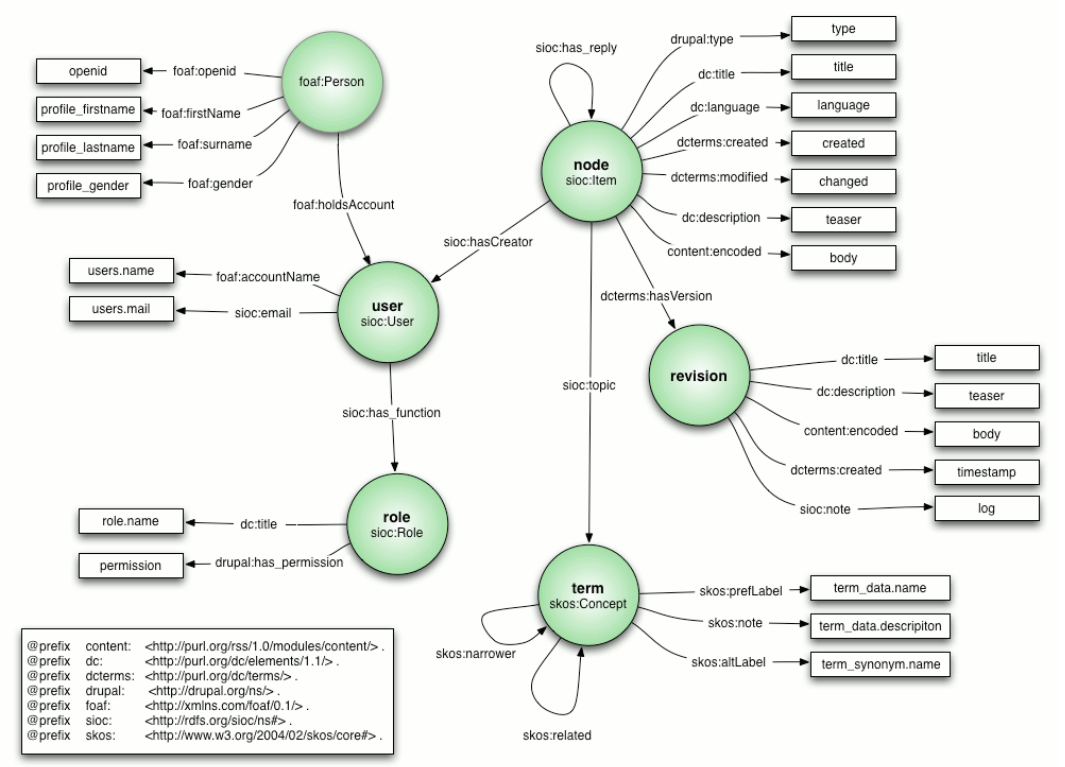


FIGURE 3.12: Drupal's RDF Schema Model (Due to `http://groups.drupal.org/node/9311`)

myExperiment's Base module rather than the whole ontology. Like myExperiment, Drupal reuses classes and properties from ontologies and schemas such as Dublin Core, FOAF and SIOC. Drupal however chooses to always use these classes/properties rather than specifying equivalence or sub-class/sub-property relationships in an ontology or schema.

There are several main differences in the model. First, Drupal separates a user's account from its profile as prescribed in the FOAF schema. myExperiment does not currently make this distinction but it may be necessary to change this in the future to better support Linked Data (see section 3.2.3). Second, Drupal uses SIOC's Roles model for permissions. The differences compared with myExperiment's Policy model has already been discussed in section 3.2.1.2.

Drupal's model also uses SKOS (see section 3.1.6.2) to represent terms (keywords) that can be associated with an Item. The myExperiment ontology also uses SKOS making a *Tag* a sub-class of *Concept* and using *prefLabel* as the human-readable title for a *Tag*.

Finally, Drupal's model supports encoding the content of an *Item* within the RDF. myExperiment does not do this for a number of reasons. First, the size of encoded data could be in the order of tens if not hundreds of megabytes, which is not really appropriate to encode in RDF. Second, the permission to view the metadata for a contribution is separate from that for downloading it. Therefore, it was deemed easier

to have a property that links to the URL where the contribution can be downloaded, so access can be managed separately.

The comparison between the myExperiment ontology and Drupal's RDF model reinforces the conclusions of section 2.3.3, that Drupal is designed to be very generic so it is able to support many different types of community-driven websites. Whereas, myExperiment has some very specific use cases, which is exemplified by the need for various extension modules to support them.

3.2.2 myExperiment's RDF API and SPARQL Endpoint

RDF data for myExperiment entities is generated using a specially designed script to map from the MySQL database model to that specified by the ontology. This bespoke script was required because as like in the design of the ontology, certain complex mappings were required that are not supported by standard relational database to RDF mapping tools. The RDF produced is accessible from the main myExperiment site through content negotiation using the same URL, namely the Non-Information Resource (NIR) URI as for the equivalent HTML page. Further to this a REST API XML version can also be requested through content negotiation. This is achieved by specifying the MIME type of the format wanted in the accept line of HTTP request, this will generate a 303 response containing the URL for the particular format requested, which can then be followed to obtain this format. Alternatively, the URL for the format wanted can be requested directly. Table 3.9 shows the MIME type and URLs returned when different formats for http://www.myexperiment.org/workflows/16 are requested.

MIME Type	Format URL
text/html	http://www.myexperiment.org/workflows/16.html
application/rdf+xml	http://www.myexperiment.org/workflows/16.rdf
text/xml	http://www.myexperiment.org/workflows/16.xml

Table 3.9: MIME Types and URLs for 303 Redirects

As much of this data has restricted access the API checks that the entity being requested is not restricted; if it is, it returns a "401 Unauthorized" response. In a web browser this prompts for an entry of a username and password, assuming there is not a user logged into myExperiment who is authorised to access the entity. Wget and other HTTP clients require the username and password to be specified in the command, as illustrated in listing 3.2.

```
wget --header "Accept: application/rdf+xml"
     --http-user=yourusername
     --http-password=yourpassword
     http://www.myexperiment.org/experiments/33
```

Listing 3.2: Requesting myExperiment RDF with User Authorisation

Assuming a username-password pair for an authorised user is provided then the RDF for the entity will be returned. Otherwise another 401 response will be sent.

All unrestricted RDF is imported into a triplestore once a day. Jena was originally chosen as the application for providing this triplestore because of its flexibility for designing scripts for importing, querying, inferencing and performing various analyses of the data. These analyses include counting the number of triples and enumerating the graphs in the triplestore. However, when 4Store became Open Source it made sense to migrate to this because it is quicker at both importing and querying data. Most importantly the degradation in performance is much less noticeable as queries become more complex, which had been noted by users as a significant problem whilst Jena had been used. 4Store has built-in scripts comparable with those written to support the Jena triplestore. Scripts that were not built-in were fairly easy to adapt to work with 4Store. Like Jena, 4Store has an active developer and user community[32], which gives some assurances that any bugs in the application will be reported and patched and it will remain state of the art.

With myExperiment's RDF imported to a triplestore it can now be queried. As discussed in section 3.1.5.1, SPARQL is generally accepted as the standard query language for RDF and myExperiment has its own SPARQL endpoint[33]. 4Store has a built-in HTTP server that provides a basic web interface with a SPARQL endpoint. However, a SPARQL endpoint interface already existed that was designed when the endpoint interfaced with Jena, so to maintain consistency this interface was adapted to work with 4Store.

Providing a SPARQL endpoint to query RDF data is useful but it relies on users being able to form queries. For projects where the main users are not computer scientists, significant effort is required to make querying accessible. Although myExperiment's main users are not computer scientists, they tend to have experience of building and using workflows, which gives some background in machine-readable entities for automating tasks. The myExperiment SPARQL endpoint attempts to support these users to build and submit queries in a number of ways:

1. Providing both a web form as well as a standard automated means (by URL encoding the query in the HTTP GET header) for submitting queries.

2. Supplying data about the status of the triplestore. Namely, the time of the database snapshot that makes up the triplestore's data and the number of triples.

3. Assistance in building queries by providing useful prefixes. These can be added to the query by clicking on them. Alternatively, the endpoint will add this automatically if the prefix's name appears somewhere in the query.

[32] `http://4store.org/support`
[33] `http://rdf.myexperiment.org/sparql`

4. Providing a user guide[34] that gives advice on how to form SPARQL queries with real-world interactive examples to query the myExperiment triplestore.

5. Formatting the error and warning messages generated by 4Store, so they are clearly visible to the user and providing troubleshooting advice on how to resolve these errors / warnings.

6. Returning useful formats for results. The standard SPARQL results format that can be used by applications. A HTML table format that is human-readable, a Comma-Separated Values (CSV) format to allow data to be ported to other applications such as Microsoft Excel and a JSON format so the data can be used in a JavaScript based application.

7. Allowing users to take their finished query and generate a service. Essentially, this is just stripping the appropriate parameters from the HTML POST form and adding them to the URL as part of the HTTP GET header. However, the concept of generating a service allows the user, who might typically be someone who puts services together to build a workflow, to be confident they have got the exact service incantation (URL) they need.

There have been a number of specific use cases that have or could benefit from using SPARQL queries. One is for querying the components of a workflow. These queries need to be able to analyse which processors are used by a workflow uploaded to myExperiment and how these are situated within a dataflow. These queries could then be used to provide a recommender service for processors / web services to help users build new workflows. These queries require looking at how processors link together. SPARQL is ideally suited to this task as its syntax makes it easy to define this interlinking. Assuming a REST API call could perform the same task it would still be a continual task for a developer to add nuances to the call whereas the SPARQL query could easily be adapted to incorporate any nuances required by the user.

myExperiment as well as being a Computer Science / Bioinformatics project has also had Social Scientists involved. Being able to find out statistics about the user base is a key requirement. The tools that social scientists use often require CSV formatted data and this is one of the reasons the SPARQL endpoint provides this format. One particular example is analysing the social network on myExperiment. The SPARQL endpoint can supply a CSV matrix for all the friendships using the SPARQL query in listing 3.3.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX mebase: <http://rdf.myexperiment.org/ontologies/base/>
SELECT DISTINCT ?friend1 ?friend2
WHERE {
  ?friendship rdf:type mebase:Friendship . {
```

[34] http://rdf.myexperiment.org/howtosparql

```
    ?friendship mebase:has-requester ?friend1 .
    ?friendship mebase:has-accepter ?friend2
  }
  UNION {
    ?friendship mebase:has-requester ?friend2 .
    ?friendship mebase:has-accepter ?friend1
  } .
  ?friendship mebase:accepted-at ?accept_time
}
```

Listing 3.3: SPARQL Query for generating CSV Matrix of Friends

Another user statistic that has been requested is the number of users from Europe. This is difficult because the field that describes which country the user comes from is defined by the user through a free-form albeit guided text field. It is consequently represented as a literal value with a string datatype. Due to the text field being guided, generally the specified value is fairly consistent, it is conceivable that URIs could be created for each country that a user comes from, but this does not solve the problem of determining how many users come from Europe. To achieve this the URIs for countries need to have a property stating which continent they are in. The best way to do this is to find a provider that already captures this information as RDF. Performing such a task is the main purpose of Linked Data.

3.2.3 myExperiment Linked Data

As described in section 3.2.2, RDF, HTML and REST API XML representations of a myExperiment entity can be returned by requesting the NIR with a specific accept type in the request, i.e. content negotiation. A means of obtaining both a human and a machine readable versions of an entity is essential for supporting Linked Data (see section 3.1.7). Implementing this aspect of Linked Data support in myExperiment was more difficult than first envisioned. myExperiment has been developed in response to user requirements and was not originally intended to be a Linked Data project. This has led to some divergence between the HTML, REST API XML and RDF representations, issues which other Linked Data projects have not had to deal. This divergence has made meeting some requirements to fully support Linked Data more challenging.

The first issue encountered due to this divergence was the choice of a URI scheme for the Non-Information Resource (NIR). myExperiment had three very different URI schemes for the three different formats, as listed in table 3.10. Creating another URI scheme for NIR would not have been appropriate as it would have quite possibly confused users. Although meeting all the requirements to support Linked Data was important, any detrimental impact on existing users had to be limited. For this reason it was decided that HTML should be considered the primary format as it is the only one accessed by the majority of users. Therefore its URI scheme was chosen as the one to be used for the NIR.

HTML	$http://www.myexperiment.org/<type>/<id>$ (e.g. http://www.myexperiment.org/workflows/16)
REST API	$http://www.myexperiment.org/<type>.xml?id=<id>$ (e.g. http://www.myexperiment.org/workflow.xml?id=16)
RDF API	$http://rdf.myexperiment.org/<type>/<id>$ (e.g. http://rdf.myexperiment.org/Workflow/16)

TABLE 3.10: myExperiment's Current URI Schemes

The next decision was what method to use for delivering these different formats. Section 3.1.7 discusses two means for doing this, Hash URIs and 303 redirects, and the differences between them. Based on these differences and due to both the scale and dynamism of myExperiment's data, 303 redirects was chosen as the most appropriate method. This choice may already have been apparent from the description of content negotiation in section 3.2.2.

The most common way of retrieving documents over HTTP is with a web browser, which generally has no means of specifying the format required. Therefore URI schemes similar to that used for the NIRs need to be defined for the various formats. Generalising, Linked Data projects use three different templates to provide URI schemes for different formats:

1. $http://<format>.example.org/<id>$
2. $http://example.org/<format>/<id>$
3. $http://example.org/<id>.<format>$

In myExperiment's case it was decided that option 3 would be the most suitable as it would create the least change from a user perspective and provide the most intuitive means of requesting different formats, as it uses the file extension paradigm that is generally well understood.

myExperiment uses Apache HTTP Server[35] to deliver its data in all three forms, although for HTML and REST this goes via a Ruby-on-Rails Mongrel Web server[36]. Apache can be extended with various modules to provide extra functionality; mod-rewrite[37] is such a module that allows rules to be defined to determine when a request should be redirected or rewritten before attempting to return a response. Much of the alignment between the new URI schemes with the existing URI infrastructure for myExperiment was achieved through a set of mod_rewrite rules, (see table 3.11). However, a number of amendments needed to be made to the website and RDF API codebase to fully support Linked Data:

[35] `http://httpd.apache.org/`
[36] `http://github.com/fauna/mongrel`
[37] `http://httpd.apache.org/docs/2.0/mod/mod_rewrite.html`

- The website's codebase was modified to respond to the .html URI scheme. However, links on the website still use the original URI scheme that now represents the NIR.
- Web pages now display the NIR allowing it to be bookmarked.
- Web pages now define link alternates to the RDF and REST API XML formats.
- URIs referred to in the RDF API now use the new NIR scheme.
- The RDF API now has a separate object to refer to the RDF graph representing the entity. This uses FOAF's *primaryTopic* property to refer to the entity the graph describes.
- Entities in the RDF API use FOAF's *homepage* property to refer to the HTML format of the entity and two Dublin Core *hasFormat* properties to refer to the RDF and REST API XML formats of the entity.

Fortunately, the REST API specifies both the URI of the call and the resource that it returns data for (see listing A.1 in Appendix A.1.1) so did not require any modifications. To best support users adapting to the new Linked Data URI schemes, certain aspects of the mod_rewrite rules in table 3.11 differ from what might be expected for Linked Data. To understand the reasons for this, it necessary to see things from a user's perspective.

Many users will have only ever interacted with one format, the HTML of a web page through a Web browser. This means that they are likely to be more familiar with the '.html' format rather than the NIR format, as it is what they will see in the address bar of their browser. Even by making the NIR visible in the page and providing the facility to bookmark it, it is still likely that if they want to share the resource being described, e.g a workflow, this will be done via the .html URI copied from the address bar of the Web browser.

Using the '.html' URI rather than NIR does not create a problem until the person receiving this URI wants to get the data for this resource for their third-party application (via the REST API) or for a SW application (via the RDF API). When requesting this URI they will specify that they want XML or RDF/XML returned in the response. Typically, a '.html' URI would ignore what format the requester is willing to accept, as it is explicitly defined in the format extension as part of the URI. However, in this scenario the recipient of the URI would not be able to make use of the resource unless they discovered they had been sent the HTML URL not the NIR, at which point they could correct it manually. Using myExperiment's mod_rewrite rules described in table 3.11 allows the URI recipient to get hold of the format of the resource they required without needing to make a manual correction. This is achieved by redirecting with a "301 Moved Permanently" response to the NIR keeping the same accept line in the request. This then causes a 303 content negotiation redirect to the correct format.

Request URI	Accept	Redirect	Redirect URI
http://www.myexperiment.org/workflows/16	text/html	303	http://www.myexperiment.org/workflows/16.html
http://www.myexperiment.org/workflows/16	application/rdf+xml	303	http://www.myexperiment.org/workflows/16.rdf
http://www.myexperiment.org/workflows/16	text/xml	303	http://www.myexperiment.org/workflows/16.xml
http://www.myexperiment.org/workflows/16.html	text/html		http://www.myexperiment.org/workflows/16.html
http://www.myexperiment.org/workflows/16.html	application/rdf+xml	301	http://www.myexperiment.org/workflows/16
http://www.myexperiment.org/workflows/16.html	text/xml	301	http://www.myexperiment.org/workflows/16
http://www.myexperiment.org/workflows/16.xml	text/html	303	http://www.myexperiment.org/workflow.xml?id=16
http://www.myexperiment.org/workflows/16.xml	application/rdf+xml	303	http://www.myexperiment.org/workflow.xml?id=16
http://www.myexperiment.org/workflows/16.xml	text/xml	303	http://www.myexperiment.org/workflow.xml?id=16
http://www.myexperiment.org/workflow.xml?id=16	text/html		http://www.myexperiment.org/workflow.xml?id=16
http://www.myexperiment.org/workflow.xml?id=16	application/rdf+xml		http://www.myexperiment.org/workflow.xml?id=16
http://www.myexperiment.org/workflow.xml?id=16	text/xml		http://www.myexperiment.org/workflow.xml?id=16
http://www.myexperiment.org/workflows/16.rdf	text/html		http://www.myexperiment.org/workflows/16.rdf
http://www.myexperiment.org/workflows/16.rdf	application/rdf+xml		http://www.myexperiment.org/workflows/16.rdf
http://www.myexperiment.org/workflows/16.rdf	text/xml		http://www.myexperiment.org/workflows/16.rdf
http://rdf.myexperiment.org/Workflow/16	text/html	301	http://www.myexperiment.org/workflows/16.rdf
http://rdf.myexperiment.org/Workflow/16	application/rdf+xml	301	http://www.myexperiment.org/workflows/16.rdf
http://rdf.myexperiment.org/Workflow/16	text/xml	301	http://www.myexperiment.org/workflows/16.rdf

TABLE 3.11: Mod_Rewrite Rules for myExperiment Linked Data

It is possible for there to be other mismatches between the accept line in the request and the format a particular URI scheme specifies, e.g. requesting `http://www.myexperiment.org/workflows/16.rdf` with a text/html in the accept line of the header. In this case it would be inappropriate to redirect back to the '.html' URI as it would prevent a user from being able to access the RDF via a web browser. Also, if the user had retrieved the RDF they could either copy the NIR it specified into a web browser and allow content negotiation to return the web page or use the URI specified by the FOAF homepage property to access it directly.

myExperiment's mod_rewrite rules treat the adoption of the new URI schemes for RDF and the REST API somewhat differently. To encourage users to adopt the new RDF URI scheme all requests using the old URI scheme are redirected with a "301 Moved Permanently" to the new scheme. This would not be appropriate for the REST API because it allows various parameters to be specified about a particular entity, e.g. a workflow, that cannot be aligned with the Linked Data URI scheme. It is therefore appropriate to only incorporate a reduced form of the REST API as part of Linked Data, i.e. basic requests for single resources, e.g. `http://www.myexperiment.org/workflow.xml?id=16`, allowing users to retrieve resources in the REST API XML format using a Linked Data approach. However, the REST API is often used intensively by third-party applications so it would be inappropriate to redirect API requests to the new URI scheme, as this would have put significant extra load to the myExperiment REST API server as well as slow down third-party applications. Redirecting from the new scheme to the old one allows a primary URI for REST API requests to be maintained and avoid any impression that the two schemes might return different data, whilst still supporting a consistent means of requesting the same resource in different formats.

A further issue that the REST API raises, is support within Linked Data for anything besides HTTP GET requests, i.e. POST, PUT and DELETE. There does not yet appear to be consensus about how Linked Data might support write, as well as read, and this is why myExperiment has avoided trying to integrate the REST API too tightly and restrict the options available when consensus does develop.

One facet of Linked Data that has not yet been considered is versioning. The properties of a resource may change over time but the resource to which they refer is still the same thing, e.g. the M5 motorway has always had that designation but over time its route and junctions can and have changed. When any modifications have been made these changes need to be captured but the data about the previous layout of the M5 is still useful, so each version should be snapshotted. This creates the problem of what to deliver when the data for the M5 is requested. There are two suggested approaches for addressing this problem (Tennison, 2009):

1. Having 'current' and 'dated' resource URIs, with the 'current' one redirecting to an appropriate 'dated' one.

2. Having 'dated' document URIs that become named graphs.

The first approach will return entities whose URI will not correspond with the URI requested, e.g. requesting `http://transport.data.gov.uk/id/road/M5/` would return data for the current version `http://transport.data.gov.uk/id/road/M5/2009-09-01`. The second would return the entity requested including a RDF Schema *isDefinedBy* property to the current version, which may then contain references to older versions.

Currently, only workflows in myExperiment are versioned. myExperiment's concept of versioning is different from the example previously given. A later version of a workflow does not necessarily supersede the previous version, although the distinction between the current version and older versions is important. Further to this, the metadata for the workflow is not just an exact replica of that of the latest version, so an adapted version of the second approach would be the most appropriate. The workflow itself refers to the current and other versions of itself, which the user can then decide to make further requests. Each version references what number version it is and of which workflow it is a version.

Proposing a standard way of versioning in the Linked Data world may not be appropriate because the relative importance of versions differs from project to project and defining how this relationship should be treated may prove controversial. The myExperiment approach to versioning demonstrates a greater equality between current and previous versions whereas other projects may consider older versions less important or even irrelevant.

So far only the means for delivering Linked Data for myExperiment has been described, not how the data is linked to other projects. There are various different facets of myExperiment that could be linked. Currently myExperiment's RDF links to three other Linked Data projects:

- ECS EPrints[38].
- The Digital Bibliography and Library Project[39] (DBLP).
- DBPedia.

For the first two projects these links come from the remote URLs that users can define as items within a Pack and therefore are represented as OAI-ORE's aggregates properties of that Pack and as OAI-ORE's *proxyFor* properties for the appropriate Remote Pack Entries. In the case of DBPedia the links are drawn from a *User's* FOAF's *based_near* and myExperiment Base's *country* properties. Using a simple text parsing algorithm,

[38] `http://eprints.ecs.soton.ac.uk`
[39] `http://dblp.uni-trier.de/db/index.html`

these values are taken to produce a DBPedia URI for those places and countries. This can never be 100% successful at producing URIs that exist within the DBPedia dataset because the values used are specified by the user without any significant restriction. However, because these values are entered by the user through a guided interface and the text parsing algorithm can clean up the data to make it more consistent, the success rate of mapping to DBPedia URIs is over 80%. These DBPedia URIs are then referenced by the *User* entity through DBPedia's *residence* property.

One of the main reasons for linking with DBPedia using its residence property is to take advantage of the information it provides about what continent a country is in. The end of section 3.2.2 described a user requirement of the SPARQL endpoint to find out how many users of myExperiment were based in Europe. Although this linking still does not make it possible to use the SPARQL endpoint to make this query, by pooling both these datasets into a single triplestore, a query like listing 3.4 would be possible.

In the future it should be possible to link to more datasets. The ability to define remote URLs as items in a Pack, means users can help with linking up myExperiment where appropriate. Beyond these there are other aspects of the myExperiment model that could also be linked. Users of myExperiment may have other URIs representing them such as their own FOAF profile URI or the URIs created for them by other Linked Data projects such as EPrints, Flickr[40] or CiteSeer[41]. As discussed in section 3.2.1.8, a separation between the myExperiment user and their account may be needed so that the predicates linking myExperiment to these other projects are appropriate. This could be achieved by creating a FOAF *account* property to create a link between the person using myExperiment and their account. Then any other Linked Data projects that refer to the person, e.g. as an author of an EPrint, could be linked using OWL's sameAs property and any accounts, e.g. a Flickr account could be linked with additional FOAF account triples in the RDF for a user in myExperiment.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX mebase: <http://rdf.myexperiment.org/ontologies/base/>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>
SELECT DISTINCT ?user
WHERE {
  ?user rdf:type mebase:user .
  ?user dbpedia-owl:residence ?country .
  ?country rdf:type dbpedia-owl:Country .
  ?country skos:subject <http://dbpedia.org/resource/Category:European\_countries> .
  ?country rdfs:label ?name
  . FILTER ( lang(?name) = ''en'' )
}
```

LISTING 3.4: SPARQL Query using DBpedia's SPARQL Endpoint for European Countries

[40] http://www.flickr.com/
[41] http://citeseerx.ist.psu.edu/

BioCatalogue, as mentioned in section 2.3, is another Ruby-on-Rails project based loosely on the myExperiment codebase. BioCatalogue is a Web 2.0 site for registering, browsing and annotating web services used by the Life Sciences community (Bhagat et al., 2010). Many of these web services are used in the workflows uploaded to myExperiment. Web services are described by the Processor class in the Components module of the myExperiment ontology, as described in section 3.2.1.4. BioCatalogue is in the process of building its own ontology and RDF representation for web services. This will make it possible to use a Linked Data approach to associate myExperiment Processors with BioCatalogue web services using OWL *sameAs* relationships. Determining where these links can be made is fairly straightforward because both entities will refer to the same URL, where the web service is available. Once defined, these links will benefit both myExperiment and BioCatalogue. For myExperiment, being able to get the annotations for web services would help to provide more detailed information about the make up of a workflow. Currently detail about web services in a workflow is limited to what can be extracted from its XML markup. Adding BioCatalogue annotations would provide a human aspect, helping myExperiment users to better understand the workflows to which they have access. For BioCatalogue it would be possible to list all the myExperiment workflows in which a web service is used. Being able to see where a web service is used in real world example may allow BioCatalogue users to more fully appreciate the uses of a web service. It may even allow them to discover a workflow they can reuse, either completely or in part, rather than writing their own from scratch.

As explained in section 3.1.7.2, VoID provides a vocabulary for describing both RDF datasets generated by a project and the links that these datasets make to other projects' datasets, known as *Linksets*. For myExperiment these *Linksets* are continually changing because they are created through user-generated content. Therefore, to produce a VoID document that remains accurate, an automated means of updating this document is required. myExperiment achieves this by performing appropriate queries across the public dataset each day after it has been updated. The results of these queries help both calculate the number of links for the VoID document and produce files listing these links in NTriple format. To make this extensible as myExperiment links with more projects, a configuration file that specifies the name of the linking project, the path to the objects in the linking project and the predicate that links to these objects, was required. Table 3.12 lists the parameters required to generate the current *Linksets* as they are defined in the configuration file. Taking the DBPedia *Linkset* as an example, this translates into the SPARQL query in Listing 3.5.

Link Project	Object Path	Link Project VoID Document
DBPedia	http://dbpedia.org/resource/	http://dbpedia.org/void.ttl#Geonames
DBLP	http://dblp.l3s.de/d2r/resource/	http://dblp.rkbexplorer.com/id/void
ECS-EPrints	http://eprints.ecs.soton.ac.uk/id/	http://eprints.rkbexplorer.com/id/void

Table 3.12: Parameters for Linkset Generation

```
SELECT *
WHERE {
  ?s ?p ?o .
  FILTER(
    isURI(?o) &&
    REGEX(STR(?o),'^http://dbpedia.org/resource/+','i')
  )
}
```

Listing 3.5: SPARQL Query for determining myExperiment-DBPedia Linkset

3.2.4 A Semantic myExperiment

A number of lessons have been learnt through the process of using and applying SW tools and techniques to an e-Research society, namely myExperiment. The first of these was how to transform the existing Ruby-on-Rails data model, built on a MySQL database, into an ontology. Despite there being a number of automated options, manually constructing the ontology allowed greater flexibility and provided the option to incorporate a number of features to improve the usability of the ontology and data it represented:

- Reuse of classes and properties from well-known ontologies and schemas.
- Abstraction of the data model to define the three main underlying aspects of myExperiment: social networking, content management and object annotation.
- Modularization of the ontology to keep apart separate aspects of functionality, allowing partial as well as full reuse.
- Capability to evolve the ontology based on new user requirements.

The next task was to deliver RDF data that had been mapped from the existing data model to the ontology's specification. This was achieved via a specially designed script. It was originally kept separate and available via a separate server but was later integrated with the main myExperiment website to meet one of the requirements of supporting Linked Data. Keeping them separate allowed for greater flexibility to evolve the ontology without having to conform too closely with that of the Ruby-on-Rails data model. After the ontology had stabilized it was appropriate to integrate with the website, to make it easier for users to access the machine-readable format.

Like the design of the myExperiment website, much consideration had to be given to design of the SPARQL endpoint. It was not just important to make it user-friendly but make it accessible to users that do not come from a Computer Science background. Therefore, a combination of information about the SPARQL endpoint and how to use it, considered choice of triplestore and design of the user interface and multiple request

methods and results formats have all been implemented so users can make the best and most efficient use of the SPARQL endpoint.

Retroactively making myExperiment a Linked Data project from one that had three separate APIs, (HTML, RDF and REST XML), was one of the greatest challenges. Many things needed to be considered to determine how changes could be made to existing APIs to meet Linked Data principles, without impacting dramatically on how existing users make use of them. Furthermore, redesigning existing interfaces so inexperienced users could make use of Linked Data with only minimal understanding, e.g. providing links to the different formats from the Web page and redirecting to '.rdf' URIs from '.html' URIs when the request accept type is set to RDF.

Although myExperiment is not on the most recent Linked Data Cloud[42], it does now meet all the requirements to appear. The final part of fulfilling the requirements was to generate a VoID document to describe the myExperiment RDF dataset and linksets to other projects. Ensuring this is consistent with the downloadable dataset and the data that can be queried through the SPARQL endpoint, which is updated each day, was an additional feature required to manage a dynamic dataset. By providing a consistent and regularly updated VoID specification gives the best support for other projects to link with myExperiment.

Many SW technologies were used to build, deliver and support a machine-readable interface to myExperiment. As well as users employing this interface, it is possible to build upon it to provide additional tools to the emergent e-Research society of myExperiment.

[42] `http://richard.cyganiak.de/2007/10/lod/lod-datasets_2010-09-22_colored.png` (September 2010)

Chapter 4

Applications of a Semantic Platform for an e-Research Society

Building an ontology for myExperiment has provided a semantic representation for an e-Research society. There are numerous applications that could make use of various aspects provided by this representation. This chapter will discuss three of them:

1. Research Objects.
2. Experimental Data in Scientific Discourse.
3. Question-Answering Systems for e-Research societies.

Research Objects (ROs) are a means of publishing a machine-readable representation of a person's research output, as opposed to a human-readable one in the form of a conference paper of journal article (Bechhofer et al., 2010b). myExperiment already supports a crude representation of someone's research output in the form of a pack. The ontology therefore begins to provide some insight of how ROs might be defined and if this link is maintained it should make myExperiment well-placed to support ROs when they come to fruition.

Another significant aspect of the myExperiment ontology has been to represent experiments that collate the enactments of workflows, along with their inputs and outputs. Scientific Discourse (also known as Scholarly Discourse) is the argumentation of competing claims and hypotheses to produce theories. In producing formal specifications for scientific discourse there needs to be a means of representing experimental evidence that supports a claim of which myExperiment's *Experiment* class is a generic example.

Section 3.2.2 identified that there are numerous different queries that people want to make about myExperiment's dataset. These people come from different backgrounds and expecting them all to be able to make use of a SPARQL endpoint without significant amounts of effort on their part is unrealistic. Therefore, a further application of a semantic platform for an e-Research society is to provide a query interface, which is similarly as flexible as a SPARQL endpoint, but more intuitive to users and requiring somewhat less training to use effectively. A Question-Answering (QA) System that makes use of the semantic representation provided by the myExperiment ontology could provide such an interface.

4.1 Research Objects (ROs)

Human-readable representations of the research process, such as journal articles and conference papers, have long been an accepted means of publishing research. However, since the advent of e-Science / e-Research, the research process has evolved making it increasingly more difficult to capture all of this in a human-readable form. The Climategate incident of November 2009[1] exemplifies where human-readable statements have been misinterpreted. This is due to the increasing amount of digital content that makes up the research process. Digital content is often highly complex and nearly impossible to describe in human-readable form to an extent where it can support the reproduction of the research it represents.

Between the University of Manchester and University of Southampton there exists a number of VRE and similar projects that are described under the umbrella of e-Laboratories. All these projects produce or consume some form of potentially highly complex research process. It is often necessary to exchange data between two or more of these projects. It has therefore become apparent that to make this exchange efficient and consistent there needs to be a standard protocol and representation.

The term Research Object (RO) has been coined to give this standard protocol and representation a name. A RO will provide a machine-readable representation, allowing the whole research process to be captured, including data, methods, processes and human-readable elements, supporting reproducible research in this ever more digital age (Bechhofer et al., 2010b). Having such a representation means that any ambiguity in the human-readable aspect can immediately be resolved by analysing or even replicating the accompanying research process to avoid any possible misinterpretation. By encapsulating all parts of the research process, there is a guarantee that important information will not be lost in any exchange between two systems.

The basic concept of an RO, as a means of flexibly grouping associated research entities together and then describing links between them, is not new. A number of more recent models are described in section 4.2 in relation to how they can support scientific discourse. However, the Boundary Objects concept from the field of sociology is one of the first published examples (Star and Griesemer, 1989). This concept focusses on the premise that often research is heterogeneous, with researchers that have different viewpoints and understanding. Therefore to make it possible for all of these researchers to work together they need an object they can share and all understand. Boundary Objects propose two main ways of achieving this, namely standardisation of methods and support for the translation between viewpoints. They also identify two important features of any model for capturing research and the research process, namely adaptability and robustness.

[1] BBC News Report Climate e-mails row university 'breached data laws' http://news.bbc.co.uk/1/hi/uk/8484385.stm

4.1.1 The Features of ROs

Reproducible research is one of the key outcomes of having a RO but reproducibility is only one of a number of features that was originally identified within the e-Laboratories projects as behaviours ROs should exhibit in assisting researchers to carry out their work (Bechhofer et al., 2010b):

Reusable A RO encapsulates and makes available data, methods and processes allowing them to be reused as part of another study, that may then produce its own RO.

Repurposable It should be possible to modify an existing RO to fulfil a new purpose, saving the need to write a new research process from scratch.

Repeatable In an e-Research world it is common that a research process needs to repeated multiple times in a consistent fashion with different datasets or parameters. An explicit machine-readable representation can both speed this up and provide guarantees of consistency for these repetitions.

Reproducible As already described, reproducing research is critical in giving confidence to a research process and the outputs it generates. For an RO to facilitate this it must contain all the data and intermediate results to allow the process to be verified.

Replayable Many ROs represent purely in silico processes. Therefore it should be possible to replay a research process with a single click and then observe it step by step.

For all of these features and several more that have been identified since, e.g. *Reliable*, *Referenceable*, *Re-interpretable*, *Respectable*, *Retrievable*, *Refreshable* and *Recoverable* (De Roure, 2010b), to be supported requires several specific properties. As part of reproducing research, there needs to be provenance to reinforce the validity of results that give confidence to the research process. A RO can contain many items; these can be both incorporated locally or referenced as a remote source. To achieve this, a model for aggregating these items and a concrete syntax for exchanging them is required. Because ROs can be reused and re-purposed, this increases the likelihood of two or more being very similar to each other, so it is essential that each one is uniquely identifiable. The files and remote links encapsulated by an RO may be well annotated but it is also necessary that the RO itself has sufficient metadata describing its purpose, structure and attributes so that someone can make sense of it.

Although ROs could be built after a research process is complete, one intention is that they are used to assist a researcher in constructing their research process, all the way from inception to publication. To support reuse, ROs must be aware of this lifecycle model

and each instance must specify which stage it is currently at, so a user can determine how it can be reused. As a RO develops through this lifecycle, changes to its metadata and the items that it aggregates need to be tracked. Tracking these changes provides both provenance for supporting reproducible research and a versioning capability, allowing ROs to be reverted or branched but still retain their original history. By doing this a user would be able to determine whether two ROs have the same origin.

Support for managing all the features previously described is essential. One critical requirement of this management is the ability to keep ROs secure, so that sharing only takes place with those who are appropriate until publication. This is essential in ensuring that credit for research and/or its process can not be usurped before publication. It is also extremely important that all aspects of a RO can be attributed to a person or process, to both make certain appropriate parties are credited at publication and to enhance a RO's provenance and ultimately the confidence in its research process as a whole.

To be able to exchange ROs between different systems and make use of various services it is necessary that the concrete syntax previously described can support a "graceful degradation of understanding". A RO has meaning on a number of levels; it may be interpreted as a simple aggregation of resources, the representation of a research process or a specific workflow detailing very domain-specific operations. Depending on the system or service being used this interpretation will vary. A graceful degradation of understanding allows more generic tools to have only a superficial understanding of the RO to perform their operations. This is analogous to the central concept of Boundary Objects. An RO could potentially have a level of detail that could be understood by parties from all research disciplines and then a more detailed level where some parts might be understood by one discipline and other parts by another.

4.1.2 Types of RO

There are many use cases of ROs that can encapsulate one of more of the five features previously described. This leads to a number of different types of RO being required (Bechhofer et al., 2010b):

Publication Object Supports reproducible research by providing an immutable record of activity so that a research process can be verified.

Work Object Captures a particular activity that may not specifically be a research process, e.g. auditing, curating, etc., so that it can be repeated, replayed or re-purposed.

Live Object Sometimes a research process may be designed by a single person but ROs provide a means for collaborative development. This is why versioning is

critical, so that multiple users can work on a single RO at the same time but still end up with something that accurately represents the research process and can be reproduced.

Exposing Object Because a RO can encapsulate multiple items and assign metadata, this provides a useful means of presenting a set of related data files that someone might want to reuse.

View/Context Object Defines a means of processing and/or presenting data not the data itself. It can then interact with an appropriate dataset to provide a specific view of this data. It is immediately repeatable as the RO can be cloned or reset to interact with a different dataset.

Method Object A RO may encapsulate a methodology that is not re-playable in itself but can be used as a template for producing a specific research process that is.

Archived Object Once an RO is no longer required for its original purpose it should not be thrown away. It is important to maintain the information it contains in an immutable state, even if the process it represents was not intended, was never capable or can no longer easily be repeated, replayed or reproduced. This is because it may contain data that could be reused or analysis techniques that may be re-purposable.

A RO's use cases may evolve over its lifetime. It may start out as a means of collaboratively developing a research process. Once it is complete, this research process may be published. Finally the RO may become defunct but because it still contains useful data and analysis techniques it should be preserved. Each of these use cases is equally important and therefore each needs management systems that can support them.

4.1.3 From myExperiment Packs to Research Objects

Part of the inspiration for the concept of ROs has came from the observation of how users of myExperiment make use of packs (see section 2.3.1). The myExperiment ontology has made it possible to define a consistent representation for packs and the RDF API has provided a means for delivering this in the concrete syntax of RDF/XML. This RDF provides a semantic representation of the pack and its items allowing it to be queried via the SPARQL endpoint or discovered via a RDF crawler. Through the use of OWL restrictions that require a *Pack* to use OAI-ORE's *aggregates* and *isDescribedBy* properties, it is possible to infer that a *Pack* is a form of OAI-ORE *Aggregation*. When a user downloads the RDF representation of a pack, to ensure that a user can use this in OAI-ORE tools, the representation also includes the *ResourceMap*, *AggregatedResources* and *Proxies* that put the resources aggregated in context of the *Pack* (see Appendix A.5). This concrete syntax representation is essential to support all the properties described in section 4.1.1 and OAI-ORE is a suitable framework on which this can be built.

4.1.4 The Future of ROs and myExperiment

e-Research has always been about providing electronic support to researchers and, as more and more research and the processes it entails are performed *in silico*, supporting these research communities is becoming ever more essential. An important application of any future semantic platform for an e-Research society will be to manage these machine-readable representations of research. myExperiment is well-placed to provide such an application because it already provides such a semantic platform that shares a basic form of machine-readable research representations as *Packs*. This means as ROs evolve to support more sophisticated research processes with richer semantics, myExperiment is well-placed to evolve with them. To be able to achieve this, a number of modifications will be needed. Some of these modifications will need to be made to the underlying model (and ontology) and others to the user interface.

First, how the myExperiment model supports the interrelation of items within a RO needs to be considered. Figure 4.1 shows how a *Pack* in myExperiment has started to be evolved to incorporate these interrelations. It shows how the *Pack* aggregates the data, workflow and results files, all of which are encapsulated as *Entries*. It is important to recognise the difference between the items and the Entries that encapsulate them, i.e is the latter describes the former in the context of the Pack in which it is aggregated. For this reason a myExperiment Pack has ORE's aggregates and its own has-entry predicated triples that reference the item and Entry respectively.

For a RO, the interrelations between items are just as important as the items themselves, Figure 4.1 shows how these interrelations defined by the *Relationship* class are encapsulated within a RelationshipEntry, (like LocalPackEntry and RemotePackEntry a sub-class of Entry), just like items are encapsulated within Entries. The Relationship class can be described with three properties, RDF's subject, predicate and object. The predicate refers to an OWL ObjectProperty that can be described by a user-defined ontology. In Figure 4.1 this shows how a data file has a relationship with a paper through some unspecified property, this might be something like *is-referenced-by* or *is-analysed-in*. An example of the RDF/XML markup required for defining a Relationship and encapsulating it within a RelationshipEntry can be found in Appendix A.3.2.

The data file and paper are referenced by the Relationship rather than the Entries that encapsulate them because the statement that the Relationship object makes, e.g. the data file is referenced by the paper, must be either universally true or not. Whether they are being referred to in the context of the Pack has no effect on whether the statement is true. However, the encapsulation of the Relationship in a RelationshipEntry is important, as this provides the ability to assign provenance in context of the Pack rather than universally. Assigning provenance directly to the Relationship is inappropriate, as the provenance for the same Relationship within different Packs, may end up be combined and generate inconsistent data. As a Relationship is a universal statement to make

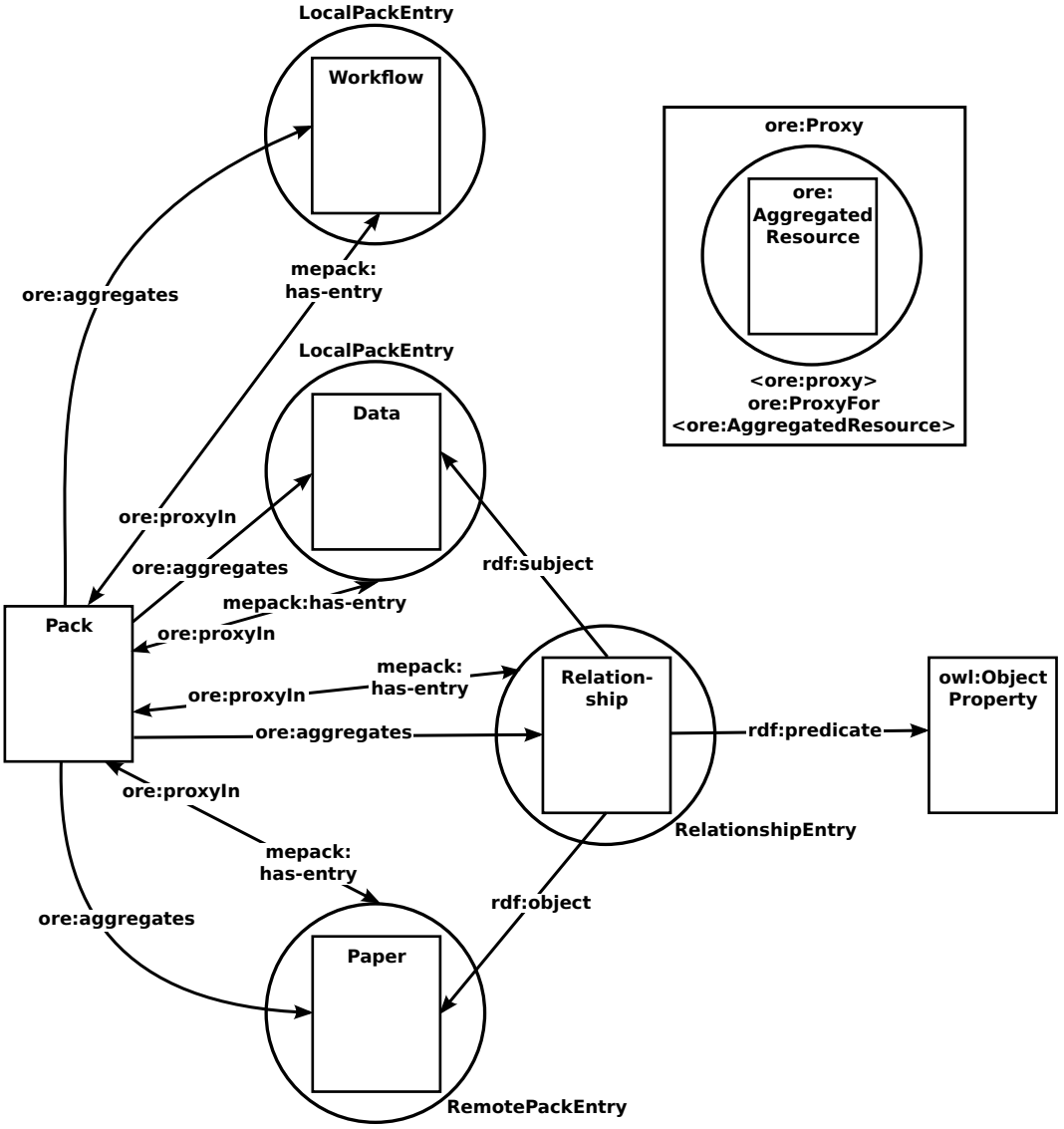


FIGURE 4.1: MyExperiment Pack evolving into a Research Object

it easier to reference, each Relationship should have a URN generated by hashing the concatenation of the subject, predicate and object URIs. UUID version 5[2] is probably the most appropriate scheme for performing this hashing, as it is preferred over version 3 by RFC4122, both of which are used for generating hashes valid for URNs from URLs. In very basic terms this simplifies any SPARQL queries but it also saves redefining the whole Relationship object each time to which it is referred. Also, whenever it is redefined it can be checked that the subject, predicate and object are consistent with the URN.

Figure 4.1 only demonstrates a Pack with one Relationship and three items. ROs of the future may well have tens of items and a comparable number of Relationships. Multiple uses of the same predicates in Relationships in the same Pack may cause ambiguity.

[2]http://www.ietf.org/rfc/rfc4122.txt

The simplest example of this is a Pack with a workflow, two input files and two output files. The workflow has 'has-input' Relationships with the input files and 'has-output' Relationships with the output files. in this case it will be important that there are also Relationships between the corresponding input and output files, so what input produced what output is clear. This case also demonstrates how even a fairly simple RO can have more Relationships than items and how a complex web of interrelations can develop. It also demonstrates how the builder of the Pack needs to have the ability to define their own properties, if they are not already provided, through user-defined ontologies.

Using SKOS was originally considered as a means for defining the properties for predicates as SKOS Concepts, which could be grouped together in a SKOS Concept Scheme. The use of SKOS's broader and narrower properties could have been used to define relationships between properties, such as *has-example-input skos:broader has-input*. SKOS's related property could also have been used to associate non-hierarchy relationships between properties. Using SKOS was considered as it was thought it would be easier than trying to define full-blown OWL ontologies. However, using OWL's subPropertyOf for representing hierarchical relationships and OWL's equivalentProperty and RDFS's seeAlso to define strong and weak versions of SKOS's related property. RDFS's isDefinedBy could also be used to ensure that all the properties are explicitly grouped together like SKOS's Concepts within a Concept Scheme. Appendix A.3.3 is an example of a basic user-defined OWL ontology with several properties that can be used as the predicates for Relationships.

Having a very basic version of an OWL ontology means that it should be possible to design a simple user interface to allow novice users to define their own properties. It should also be possible to build a similarly straightforward interface for defining the Relationships between Pack items. Both of these interfaces would benefit from a graphical user interface to encourage uptake. Having such an interface would make it easier for a user to visualise what is in their RO or ontology, how it fits together and ultimately provide a more intuitive (paradigm-based) means to associate together new items or properties. A canvas-based graph editor like Graphviz would probably be the most suitable GUI for this task. There are a number of this type of editor now being made available online, which myExperiment could use to provide a platform-independent interface. In the case of building user-defined ontologies, more experienced users may want to download their ontologies so that can edit them by hand or use a graphical application such as Protégé, before re-uploading. Providing this facility would allow more sophisticated ontologies to be developed where appropriate and therefore provide the potential for richer and more complex ROs to be constructed. However, permitting both methods of editing requires some safeguards to ensure the ontology does not become invalid.

There are many different reasons for developing a user-defined ontology. First it is important to state that user-defined does not necessarily, in fact generally, does not

refer to a single user developing an ontology in isolation. Typically a group of users or even a community may want to develop and ontology for their own Relationship predicate properties. Examples of this may be for different scientific domains such as Chemistry, Bioinformatics, Astronomy, etc. and beyond this particular sub-domains such as Organic Chemistry or gene expression. Also different ontologies may be needed for different types of RO, such as live, publication, archived, etc. Even within the same type of RO there may be a need be a need for individual ontologies to provide properties so an RO can conform to a particular standard. These different use cases demonstrate how the ontologies themselves can be hierarchical with more specific ontologies extending the more generic.

The need to able to facilitate user-defined ontologies demonstrates how just making research available is Linked Data is not sufficient to support users to share their research and/or research process in a way which allows a RO to exhibit all the features describe in section 4.1.1 (Bechhofer et al., 2010a). For this reason ROs have a model to help set these user-defined ontologies in context. As already discussed ROs are to be built on top of OAI-ORE. The specification for RO model itself will have two main layers:

- Research Object Upper Model (ROUM).
- Research Object Domain Schema (RODS).

The purpose the ROUM will be to incorporate OAI-ORE along with other e-Research ontologies, such as SIOC, OAC and the Annotation Ontology (AO), to provide a specification, so basic ROs of the types defined in section 4.1.2 can be instantiated. RODSs are the user-defined ontologies. As well as being able to define Relationship predicates, as already discussed, they will be able to define additional properties used within the RO and further restrictions on what makes a valid type of RO. Each RODS may just refer to entities defined in the ROUM but they may also refer to entities defined in one or more other RODSs. This allows for a rather loosely hierarchical relationship between RODSs with those more specific extending the generic either to define ontologies for scientific sub-domains or more specific features for types of RO. The more generic RODSs are likely to be developed and used by larger user groups. Limiting how and when these can be updated is important to ensure ROs that use them remain consistent. For this reason the most generic RODSs should probably be maintained by the teams responsible for myExperiment and other systems that manage ROs. Management/maintenance policies for more specific RODSs will need to be considered on a case by case basis.

Fairly few changes have been required to implement a basic version of the ROUM and support for RODSs in the myExperiment ontology. Adding Relationships and Relationship Entries to the Packs module of the ontology has been a fairly simple addition. There is no specification for the ontologies and object properties that are used for representing the RODSs as this is defined by the OWL ontology and RDFS.

The interaction between myExperiment and ROs is important for the continued success of the former and the uptake of the latter. myExperiment has begun to evolve beyond a system that just allows grouping of research entities to provide researchers with a means of sharing their research processes. This evolution needs to continue otherwise myExperiment will not be able to continue to communicate accurately and with sufficient detail users' ever more complex research. Without a means and a semantic platform for the e-Research society on which ROs can be shared, discovered and built collaboratively, there is limited motivation for researchers to take up the RO model and start using them.

4.2 Scientific Discourse

Scientific Discourse (also called Scholarly discourse) is the collaborative argumentation of different claims and hypotheses, with the support of evidence from experiments, to construct, confirm or disprove theories. Argumentation is the cornerstone of scientific research and has been present in some form since the times of Ancient Greece and the advent of dialectics (Adler, 1927). In more recent history, consideration has been given to the components of argumentation (Toulmin, 1969). Toulmin proposed that there are six interrelated elements for "analyzing arguments" that are used in scientific discourse:

- Claim
- Evidence (Data)
- Warrant
- Backing
- Rebuttal
- Qualifier

A *Warrant* is a statement that supports some *Evidence* being ascribed to a *Claim*. *Backing* is often required to ensure that a *Warrant* is credible. *Rebuttal* is an addition (often a caveat) to a *Warrant* that may lead to additional *Evidence* being required to support a *Claim*. Argumentation is rarely unambiguous, *Qualifiers* are therefore required, e.g. certainly, probably, possibly, to specify the certainty of a *Claim* and the burden of *Evidence* required to support it.

With Toulmin's and other specifications of argumentation, it has been possible to consider the design of argumentation systems as a structural computing problem. Structural computing problems concern themselves with the linking together of objects; this is the key attribute of hypertext. Argumentation support however is a special case of this, due to need for semantic constraints to restrict which objects can be linked together (Nürnberg et al., 1997). Certain aspects of argumentation complicate the design of a hypertext system for argumentation support, e.g. disambiguating personal associations and formal relationships. The development of Semantic Web technologies provides some of the solutions to these complications.

4.2.1 Scientific Discourse and the Semantic Web

ScholOnto was originally designed as an ontology-based hypertext argumentation system (Buckingham Shum et al., 2000). However, as the ScholOnto project began before many

Semantic Web technologies had been standardised, it used the Operational Conceptual Model Language (OCML) and the WebOnto application to build this ontology. Listing 4.1 is an example of OCML syntax. OCML is designed for specifying ontologies so therefore shares similar attributes to OWL and RDFS.

```
(def-class skc-theory-model (scholarly-contribution)
  ;;; non-argumentation relationships
   (addresses :type skc-problem)
   (analyses :type (or skc-data, skc-idea))
   (uses-applies :type (or skc-analysis, skc-approach,
    skc-data, skc-idea, skc-language, skc-methodology,
    skc-phenomenon, skc-software, skc-theory-model))
   (modifies-extends :type skc-theory-model)
   (is-an-example-of :type (or skc-theory-model, skc-data,
    skc-idea, skc-phenomenon))
   (encapsulates :type skc-theory-model)
   (envisages :type scholarly-contribution)
   (predicts :type scholarly-contribution)

  ;;; argumentation relationships
   (confirms :type scholarly-contribution)
   (is-consistent-with :type scholarly-contribution)
   (is-inconsistent-with :type scholarly-contribution)
   (takes-issue-with :type scholarly-contribution)
   (raises-problem :type problem)
   (refutes :type scholarly-contribution)))
```

Listing 4.1: Extract of OCML Markup for ScholOnto

The central entity of the ScholOnto ontology like Toulmin's specification is the *Claim*. This *Claim* asserts a *Concept*, which can take one of many different forms, namely an analysis, approach, data, idea, language, methodology, phenomenon, problem, software or theory. Each *Claim* must identify the *Agent* who submitted it, whether they be a human or an automated piece of software. A *Claim* must also have a *Justification*; this is equivalent to Toulmin's *Backing* element. This *Justification* can be just free text, a formal document or even a semantic structure that logically describes how the *Claim* is justified. A network of *Concepts* and *Claims* can be built up using *Relationships*. Importantly these *Relationships* are clearly disambiguated as either argumentative or non-argumentative, as can be seen in listing 4.1.

With the ScholOnto ontology already defined in OCML, once the appropriate Semantic Web technologies had been standardised it was possible to transcribe it to an RDF schema[3] to become the earliest Semantic Web specification for scientific discourse. Since then, several other projects have set about producing such specifications, one example is the Semantic Web Applications in Neuromedicine (SWAN) project (Ciccarese et al., 2009).

[3] `http://projects.kmi.open.ac.uk/scholonto/resources/Scholonto2.rdfs`

As its name suggests the SWAN project's domain is neuromedicine but the ontology it has produced[4] is intended to be reusable in other domains. Like the myExperiment ontology (see section 3.2.1) it has a modular structure so that different components can be added dependent on the functionality required and the domain covered. Figure 4.2 shows the entire architecture for SWAN Commons ontology[5]. Unlike the myExperi-

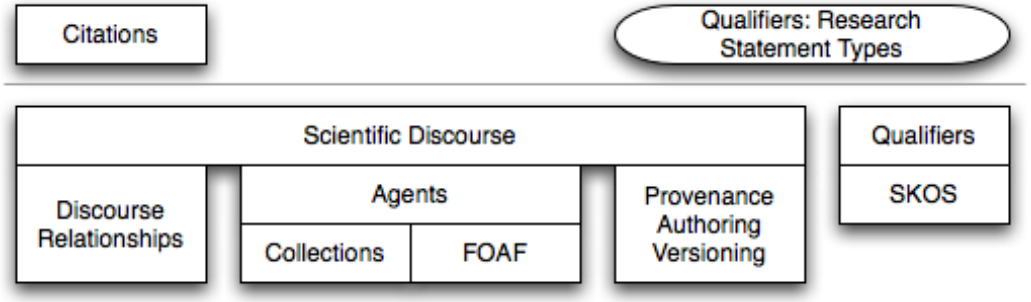


FIGURE 4.2: SWAN Commons Ontology Architecture (Due to `http://swan.mindinformatics.org/ontology.html`)

ment ontology there are a number of modules that provide the specification for basic components:

FOAF Essential specifies a subset of the FOAF vocabulary, which is OWL DL compliant unlike the full FOAF specification[6]. This allows agents, i.e. people, groups and organizations, projects, documents and user accounts to be defined.

Provenance, authoring and versioning provides properties for specifying the status of documents comparable with *Concepts* in the ScholOnto ontology. Where and how a document came about and who is responsible for it is essential information for scientific discourse.

Collections allows a number items to be grouped together as an unordered set, a bag or an ordered list.

Discourse Relationships defines a set of relationship much like those described in listing 4.1 for ScholOnto.

Reification allows a *BinaryRelationship*, i.e. a property, to be instantiated as an object so that additional information can be assigned. Much like how *Memberships* in the myExperiment ontology is a reification of SIOC's *has_member* property.

These modules are designed to be stand-alone so can be individually reused by other projects for the expressivity they provide.

[4] `http://swan.mindinformatics.org/ontology.html`
[5] `http://swan.mindinformatics.org/ontologies/1.2/swan.owl`
[6] `http://xmlns.com/foaf/spec/index.rdf`

As data used in SWAN is pulled in from external resources, over which it has no direct control, it is inevitable that it will contain references to things that are not uniquely identified. An example of this is that a journal database will have a collection of articles, each with its own set of authors, often just specified in plain text. As different journals may format the same person's name differently and because it is possible for a name to belong to more than one person, SWAN provides the *Agents* module that imports the specifications of most of SWAN's basic modules to handle this ambiguity. It does this by instantiating *PersonName* objects for each author it finds for an article. If the person who that *PersonName* represents can be determined then an *aka* relationship can be defined that has the FOAF *Person* representation as its object. If required this *aka* property can be reified so provenance data can be assigned to this relationship. It is common in scientific discourse that not just a single person is responsible for a particular discourse element. Therefore the *Agents* module also provides homogeneous and heterogeneous lists of classes of agent that can be assigned to a discourse element.

To build the complete SWAN scientific discourse ontology a further module is required that imports the *Discourse Relationships* and the *Agents* modules. It specifies a *DiscourseElement* that allows *ResearchQuestions* and *ResearchStatements* (comparable with Toulmin's *Claim* element) to be defined. It also provides reified classes for the discourse relationships making them comparable with Toulmin's *Warrant* element and facilitating the association of entities comparable with Toulmin's *Backing*, *Rebuttal* and *Qualifier* elements.

Qualifiers are not part of the SWAN scientific discourse ontology but are added to SWAN Commons as a separate set of modules. This is because most qualifiers are quite often domain specific. To allow qualifiers to be instantiated, SWAN defines its own version of SKOS as a module, thus ensuring the ontology as a whole remains OWL DL compliant. A second module defines a *Qualifier* class. A *Qualifier* may be assigned *Meanings* using a similar vocabulary to SKOS, i.e. the associated *Meaning* can be broader, narrower or equivalent to the *Qualifier*. A *QualifierConcept* class that is a subclass of both *Qualifier* and SKOS's *Concept* can then be defined. This allows SKOS controlled vocabularies of *Qualifiers* to be built up for the appropriate domain. Finally, SWAN Commons incorporates a basic set of *ResearchStatement QualifierConcepts* that are comparable with ScholOnto's *Concept* types.

As already discussed, SWAN imports data from external resources, such as journal databases. To be able to fully specify this data, another module for called *Citations* is required. As well as providing the *Citation* class it provides classes for different types of contribution that exist in the world of scientific publication, i.e. books, book chapters, conference proceedings and articles, news and comments in journals, newspapers and on the web. Properties are also required so that roles such as author, editor or publisher can be assigned to these contributions. The allows *ResearchStatements* to use *Citations* like Toulmin's *Backing* element.

SWAN Commons is the standard distribution of the SWAN ontology. However, additional SKOS-based modules have been specified to define pathogenic narrative and mechanisms taxonomy qualifiers and a Nature[7] stem cells cheat sheet to produce SWAN Alzheimer[8]. This is a more specific distribution designed for scientific discourse about Alzheimer's disease. Because SWAN Commons contains a *Life Sciences Entities* module (containing representations for genes, organisms, proteins, etc.) and a connector to the Gene Ontology[9] (GO), it is still somewhat domain specific. However, SWAN's modular design allows for a more generic distribution to be created without these. Additional modules could then be added to this more generic distribution to support different domains.

4.2.2 Alignment of Scientific Discourse Related Ontologies

What makes the SWAN project particularly relevant is that through the design of the ontology it has been able to outreach to similar projects through the Healthcare and Life Sciences (HCLS) Interest Group[10] of the W3C to begin a process of alignment across several different aspects of the ontology:

- Online Communities
- Discourse Models
- Bibliographic Citations
- Curation
- Experimental Data

Aligning with the models and ontologies from these areas gives the potential to create new modules for the SWAN ontology to increase its expressive power and capture scientific discourse in greater detail. The ultimate goal of this alignment is to provide an ontology for capturing the SCientific ARticle of The Future (SCARF).

The outreach process also gives the potential to work with integrating tools such as the aTag generator[11] (Samwald and Stenzhorn, 2009). This tool allows a web page to create links so that concepts can be bookmarked with the users own choice of tags. These aTags can then be embedded into a web application such as Wordpress[12] and be discovered using RDF-enabled search engines. RDF visualization tools can also be used to analyse a set of aTags, e.g. those generated by a particular user, group or community.

[7] http://www.nature.com/
[8] http://swan.mindinformatics.org/ontologies/1.2/swan-alzheimer.owl
[9] http://www.geneontology.org
[10] http://www.w3.org/blog/hcls
[11] http://hcls.deri.org/atag/generator/
[12] http://wordpress.org/

4.2.2.1 Online Communities

SWAN already incorporates part of the FOAF specification so it can assign discourse elements and other entities to uniquely identified agents. As already demonstrated in section 3.1.6.4, SIOC provides a semantic representation for community discussion, which can be considered as a more general form of scientific discourse. Figure 4.3 highlights

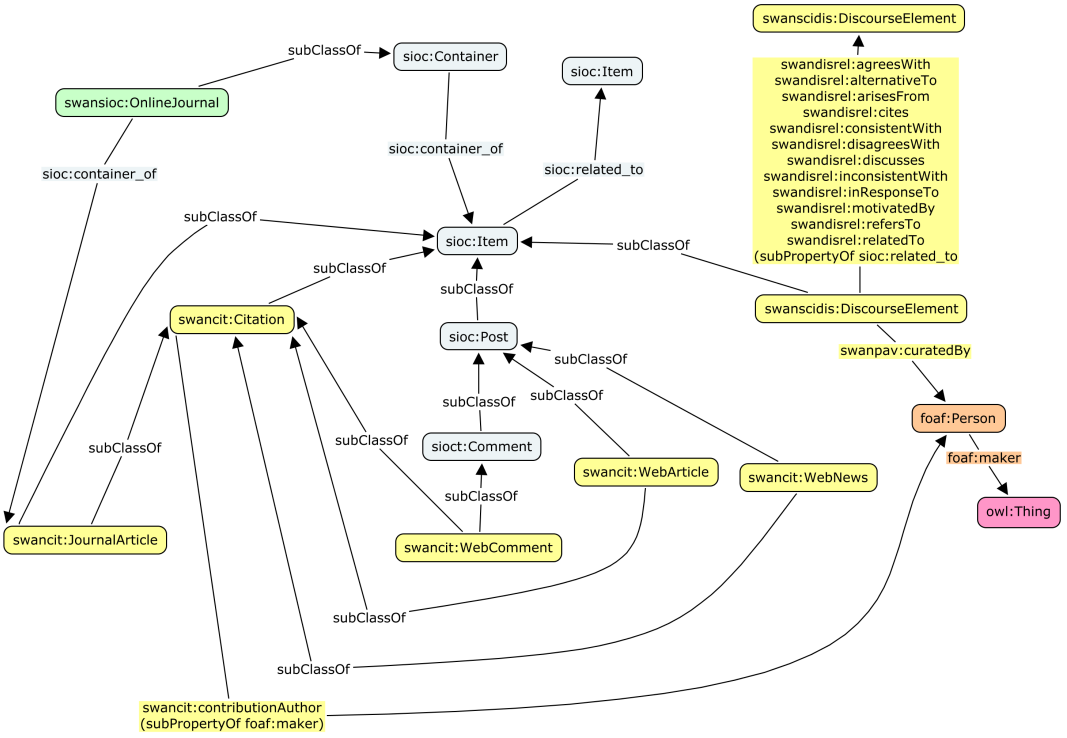


FIGURE 4.3: SWAN SIOC Alignment
(Due to `http://www.w3.org/TR/hcls-swansioc/`)

this generalization and shows where the two ontologies can be aligned (Passant et al., 2009). All nodes in the discourse network along with all types of SWAN *Citation* can be considered subclasses of SIOC's *Item* class or one of its subclasses. SWAN's discourse relationships can also be assigned as sub-properties of SIOC's *related_to* property. These associations are specified in the SWAN/SIOC ontology module[13]. The main motivation for aligning SWAN with SIOC was to ease the integration/alignment of other aspects of the ontology, i.e. discourse models, bibliographic ontologies and experimental data.

4.2.2.2 Discourse Models

SWAN provides a basic discourse model through the properties it provides in its *Scientific Discourse Relationships* module. There exist many other models for expressing discourse relationships beyond those already discussed (Harmsze, 2000), (Teufel and Moens, 2002), (Mizuka et al., 2006), (Lisacek et al., 2005). There are several aspects

[13] `http://rdfs.org/sioc/swan`

that differentiate these models (Groza et al., 2009). These include whether they support coarse-grained rhetorical structure, (e.g. abstract, motivation, background, conclusion, etc.) that may be used to summarize a network of finer-grained discourse. The type of discourse relationships they support, such as argumentative (e.g. *disagrees*, *agrees*), cognitive coherent (e.g. *consistent-with*, *alternative-to*) or rhetorical (e.g. *antithesis*, *circumstance*, *concession*, *purpose*) and whether these relations have implicit or explicit polarity or weighting. Models also vary on the degree of metadata and provenance support they provide.

Through the HCLS interest group's scientific discourse subgroup, a process of alignment has begun between two models:

- Annotations, Background, Contribution, Discussion and Entities (ABCDE) (de Waard and Tel, 2006).
- Semantically Annotated LaTeX (SALT) (Groza et al., 2007).

Both the ABCDE and SALT models have their basis in annotating the original document, which in this case is written in the LaTeX format[14]. The ABCDE model has its own LaTeX style-sheet that has macros to allow Dublin Core metadata annotations to be assigned. The model requires the author to consider whether each section is *Background*, *Contribution* or *Discussion* by assigning them with a B, C or D respectively, creating coarse-grained rhetorical blocks. The user is then encouraged to annotate key sentences of these sections that can then be extracted to generate a structured abstract rather than producing one by hand. Papers contain entities. Some of these such as references, figures, tables etc., are already extracted by LaTeX to produce listings, whilst others, such as project websites, names of people and places are not. Marking these up significantly enriches metadata that can be extracted, which is the main purpose of the ABCDE format.

SALT like ABCDE also has its own LaTeX macros to allow scientific discourse to be annotated within a document. However, it is intended for SALT to be usable in other document formats. SALT's whole discourse model is very detailed and tries to extend the representation provided by ABCDE. SALT has three separate ontologies. The *Document* ontology captures the linear structure and various components (sections, paragraphs, sentences, figures, tables, etc.) of a document. The *Rhetorical* ontology externalizes the rhetoric and argumentation encapsulated within the document and its components. The *Annotation* Ontology allows the latter to be linked to the former.

SALT's Rhetorical ontology first considers *Rhetorical Relations*; these provide a reification of the relationship between a *Claim* and an *Explanation*. This has a basis in the

[14] `http://www.latex-project.org/`

Rhetoric Structure of Text (RST) theory[15] and reuses some of the relation types it defines, e.g. *Antithesis*, *Concession* and *Means* (Taboada and Mann, 2006). The relation type explains the reason for the existence of the *Claim-Explanation* pair in a document. Each *Rhetorical Relation* must specify the *Rhetorical Structure*, e.g. abstract, conclusion, motivation, etc., to which it belongs. *Explanations* and *Claims* are considered as *Rhetorical Elements* that may have associated *Arguments* to discuss their validity.

ABCDE and SALT share the same goal of adding semantics to link the physical structure of the paper to the scientific discourse it contains. This goal is slightly different from that of SWAN, which is to create a knowledge base. Therefore aligning SWAN with these models will allow it to map the knowledge it represents more specifically to parts of the document rather than just the document as a whole. The first part of this alignment is described as the Ontology or Rhetorical Blocks (ORB), which has an OWL representation[16] (HCLS Scientific Discourse Sub Task Group, 2010). This ontology module aligns the different ontologies coarse-grained structure, into five different sections:

1. Introduction
2. Methods
3. Results
4. Discussion
5. References

With these 'sections', it is now possible to start aligning the less coarse-grained elements of these models, known as middle-grained. These can be considered as the sub-elements of these coarse-grained 'sections', where the introduction may contain positioning, hypotheses, etc. or the discussion may contain related work, conclusions, etc.

Another ontology that captures the underlying components of a publication is the Document Component Ontology (DoCO) (Shotton and Peroni, 2010). It describes itself as "an ontology for describing the component parts of a bibliographic document" and is part of the Semantic Publishing And Referencing (SPAR) ontologies suite. DoCO in part is an amalgamation of the rhetorical block elements provided by SALT and the structural components described by the Document Structural Patterns ontology (Di lorio et al., 2010). It is not yet clear whether DoCO could provide extra coverage beyond that provided by the existing alignment between SWAN, SALT and ABCDE. Many of the DoCO's structural components already exist within the alignment, mainly in SALT's Document ontology. However, adding DoCO to the list of aligned models provides a

[15] `http://www.sfu.ca/rst/`
[16] `http://esw.w3.org/images/d/d2/Orb-0_1.owl`

means of testing the existing coverage. It also gives those marking up their publications the option of choosing the model best suited to them, with confidence that it can still be interpreted in systems that use alternate models.

4.2.2.3 Bibliographic Citations

SWAN has its own basic modelling for bibliographic data using its *Citations* module. Slightly more sophisticated models for representing bibliographic data such as the Bibliographic ontology (BIBO) exist (Giasson and D'Arcus, 2009). The SWAN ontology's modular structure means that additional modules could be defined to enhance or replace the existing *Citations* module. Through outreach to the HCLS interest group, there is an intention to align SWAN with the Citation Typing Ontology (CiTO) and the Publishing Requirements for Industry Standard Metadata (PRISM) specification (Shotton, 2009), (IDEAlliance, 2009).

Unlike discourse models in section 4.2.2.2, CiTO's focus on the provision of a semantic annotation for the reference lists found at the end of a paper. CiTO imports the document types defined by DCMI[17], providing significantly more options than SWAN provides in its *Citation* model. This makes it possible to be more specific about what a publication is as well as supporting items that are not specified in SWAN, e.g. letters, spreadsheets, ontologies, presentations, images, etc. CiTO, unlike SWAN and BIBO, recognises the Functional Requirements for Bibliographic Records's (FRBR) distinction between a piece of *Work*, the *Expression* of the work and its *Manifestation*. For example something is written as a research paper, is published as a journal article and is available as a web page (International Federation of Library Associations and Institutions, 1998).

To build its citation network, CiTO provides various properties to interlink cited works. This makes it more sophisticated than a standard citation network because it can specify why and how many times something cites something else. These properties also allow scientific discourse at the even coarser-grained level of a publication, compared with the granularity described in section 4.2.2.2. Therefore an alignment with SWAN would allow this additional level of scientific discourse to be encapsulated.

SWAN and BIBO capture a basic level of metadata about a publication. PRISM specifies a much more detailed level of metadata, including a subset of Dublin Core elements, that it extends to provide a wide range of metadata properties for content publication, licensing, and reuse situations (IDEAlliance, 2009). Building an ontology module that implements the PRISM specification would allow a much wider range of metadata to be assigned to a citation.

[17] `http://dublincore.org/documents/dcmi-type-vocabulary/`

4.2.2.4 Curation

The Annotation Ontology[18] (AO) is an ontology for annotating scientific documents on the web, i.e. performing curation (Ciccarese et al., 2010). Like the OAC ontology (see section 3.1.6.7) the ontology is based on the Annotea model. It was originally designed for use annotating entities in the SWAN scientific discourse model but can be used more generically for annotating any document, although it has specific features for scientific documents.

One feature of AO is the ability to indicate that an *Annotation* is made by a piece of software invoked by a person with both the software and the person captured as provenance for the *Annotation*. Like the OAC ontology, an AO *Annotation* links to the entity being annotated (like OAC's *Target*) and a separate entity for the content of the *Annotation* itself (like OAC's *Body*). One main difference to OAC is that the *Annotation* can be defined as belonging to set, i.e. a *Curation* process. Figure 4.4 shows an example of an AO *Annotation*, which represents part of a *Curation*, where a piece of software was run by a user over a SWAN *Claim* to mark up proteins that can be found within it.

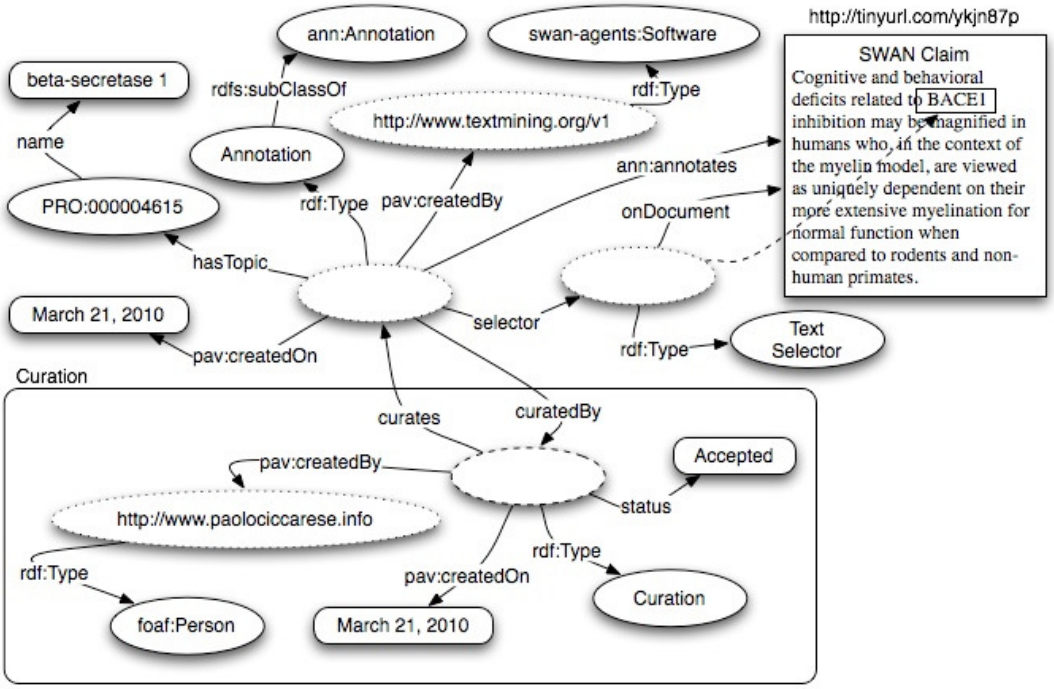


FIGURE 4.4: Example AO Annotation
(Due to (Ciccarese et al., 2010))

One feature of both *Curations* and *Annotations* is their status. Both a human and a piece of software may create erroneous annotations; having a status allows unchecked entities to be marked as pending and afterwards as either accepted or rejected. Steps have

[18] `http://annotation-ontology.googlecode.com/svn/trunk/annotation.owl`

already been taken to align AO with SIOC and the support it provides for community annotation. An AO *Annotation* is a sub-class of SIOC *Item* and an AO *Curation* is a sub-class of SIOC *Annotation Set*.

Curations are needed to be able to record the mark up that they produced through the application of rhetorical models and determining bibliographic citations described in the previous two sections. These tasks will have been undertaken by a person possibly with the assistance of a software application. Capturing the provenance of these *Curations* is essential in building up an accurate knowledge base for documents. Inconsistencies may occur in these knowledge bases, especially when it is contributed to by multiple disparate users using different pieces of software. Maintaining provenance is the only way of resolving these inconsistencies.

4.2.2.5 Experimental Data

Representation of the experiments that provide the evidence component of scientific discourse has up to now not been a central focus of the SWAN ontology. There have been several projects that have set out to define a representation for experiments so it can be referenced in scientific discourse. NeuroScholar is an early example of this (Burns, 2001). NeuroScholar's model describes flow charts, data containers, data mappings and reusable data structures for measurements, as well as more neuroscience specific concepts, to capture the experimental process.

ISA-Tab is a project that provides various tools to allow standard templates for experiment metadata to be defined and instantiated to capture information about individual experiments and then convert them to different formats or upload their data to a database (Field et al., 2009). ISA-Tab is based in the biological / biomedical domain and incorporates similar domain-specific projects such as MAGE-TAB[19], FuGE[20] and PSI[21]. It uses the Minimum Information for Biological and Biomedical Investigations[22] (MIBBI) and ontologies from the Open Biological and Biomedical Ontologies (OBO) Foundry[23] as standards in aiding the design of these experimental templates.

Both projects have focussed on specific domains. SWAN's goal is to provide support for representing experiments across multiple domains but has been initially focussed in the same domains as NeuroScholar and ISA-Tab. The OBO Foundry made use of by ISA-Tab is also used to help build the Ontology for Biomedical Investigations (OBI[24]). It is intended to provide 'universal' terms "applicable across various biological and technological domains" which can be used to describe biological and clinical investigations

[19] http://www.mged.org/mage-tab
[20] http://fuge.sourceforge.net/
[21] http://www.psidev.info/
[22] http://www.mibbi.org/index.php/Main_Page
[23] http://www.obofoundry.org/
[24] http://purl.obolibrary.org/obo/obi.owl

(Courtot et al., 2008). OBI achieves this by importing key terms from ontologies hosted in the Foundry and assembling them following the Basic Formal Ontology (BFO) (Smith, 1998).

Two ontologies that are subsumed by OBI are the Microarray Gene Expression Data (MGED)[25] ontology and the Experimental Factor Ontology (EFO). MGED defines "Concepts, definitions, terms, and resources for standardized description of a microarray experiment"[26]. EFO is specified in both OWL and OBO format and is designed to model the experimental factors in ArrayExpress[27], an archive for genomic experiments (Malone et al., 2008).

Aligning SWAN with OBI, focusing on the aspects defined in the MGED ontology and EFO, will allow SWAN to better represent the experiments involved in scientific discourse model. The myExperiment ontology has its own module for representing experiments that provides a different focus. Primarily, the intention of myExperiment's *Experiment* class, as discussed in section 3.2.1, is to be able to represent an *in silico* experiment, where one or more *Workflows* are enacted one or more times, processing some data and producing an output. To be able to align SWAN, aspects of OBI and the myExperiment ontology has required some consideration. Figure 4.5 shows a proposed alignment of SWAN's *Discourse Elements* with myExperiment's *Experiment* objects. At present it proposes using terms from the MGED ontology but as previously discussed, these may become superseded by terms from OBI as this is a more generic ontology for biological/biomedical investigation.

To be able to align these three ontologies an additional 'glue' module is required to express their synergy. Central in the 'glue' module is the concept of a *Study* motivated by a previously described *ResearchQuestion* in the SWAN knowledge base. This *Study* entity would be equivalent to an MGED *Experiment* entity and have *ExperimentalFactors* associated with it. It would allow *Data Acquisitions* from experimental *Assays*, as well as *Computations* from enacted *Workflows* to be assigned to it. This would allow scientific discourse statements to be made about the study in the form of *Hypotheses* made up of one or more *Claims* backed by the evidence provided by the *Data* produced by the experimental *Assays* and/or enacted *Workflows*.

In addition to the alignment proposed in Figure 4.5, the planned integration between myExperiment and BioCatalogue, as described in section 3.2.3, is also relevant to defining the provenance of analysis generated by enacted *Workflows*. This further alignment with a BioCatalogue ontology would allow social metadata captured by the BioCatalogue website to be incorporated so that the services within a workflow can be annotated as well as the workflow itself. It is essential that every element, even those that are subelements can be annotated within the alignment of scientific discourse ontologies model.

[25] `http://mged.sourceforge.net/ontologies/MGEDOntology.1.3.0.1.owl`
[26] `http://mged.sourceforge.net/ontologies/MGEDontology.php`
[27] `http://www.ebi.ac.uk/arrayexpress/`

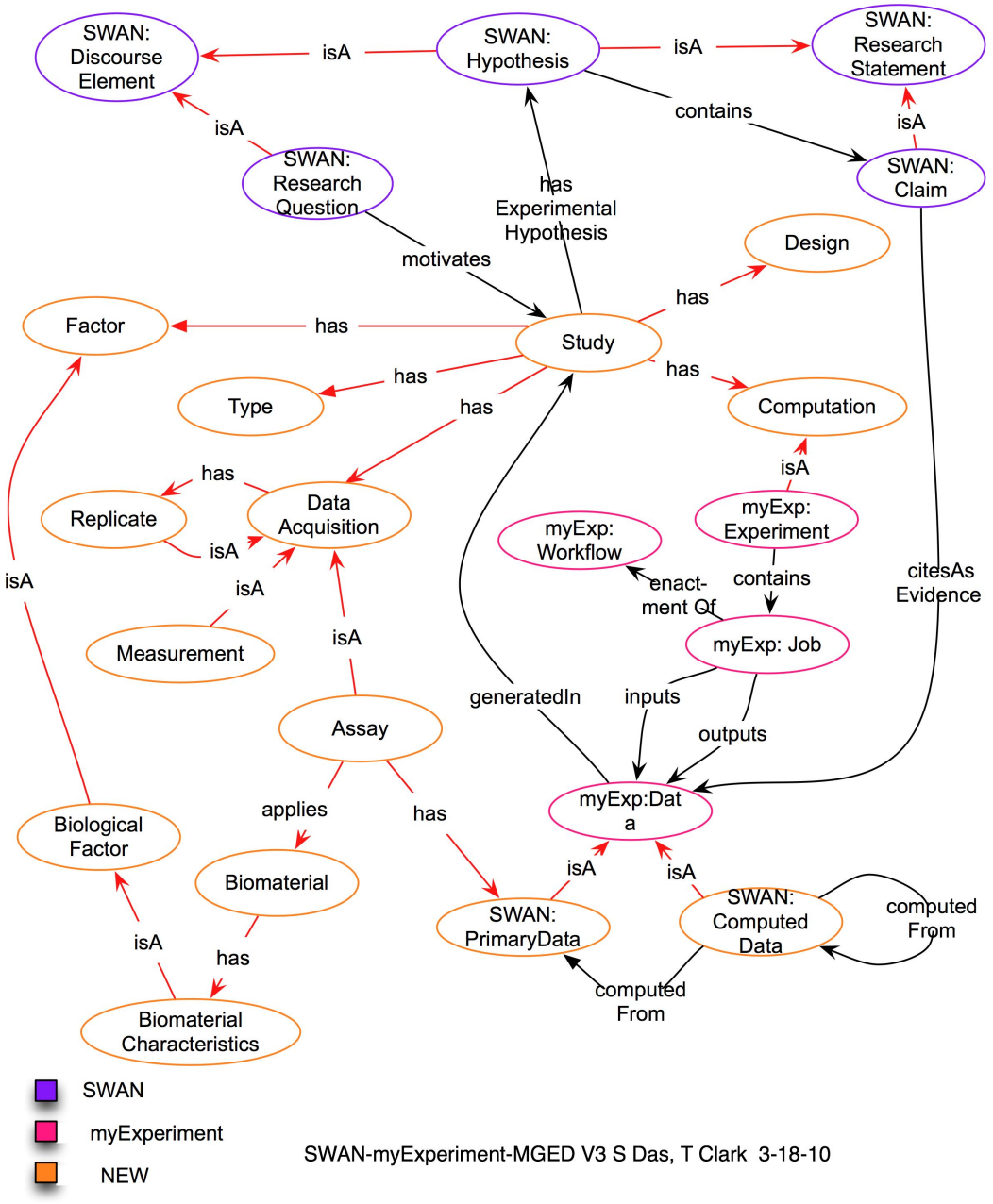


FIGURE 4.5: SWAN Alignment with Experimental Ontologies (Due to `http://esw.w3.org/HCLSIG/SWANSIOC/Actions/SWANmyExpArray`)

Different elements will have different properties and by including a BioCatalogue ontology with the model, it would be possible to incorporate the unique properties it has for workflow services. These could then be used within the AO model described in section 4.2.2.4 for curation of this aspect of a *Study*.

One area that the alignment of experimental ontologies that does not have great coverage is the design and instantiation of experimental plans. Currently, a *Study* can have *Hypotheses* and *Claims* that can be supported by *Evidence*. How this *Evidence* is collated

is represented by *Data Acquisition*, *Computation* and *Analysis* steps. However, how experimenters go from their *Hypotheses* and *Claims* to determining the steps they are going to carry out to obtain their *Evidence*, i.e. their plan, is unclear. There exist a number of projects that have a closer focus on the concept of an experimental plan.

oreChem is a relatively new project for providing semantic representations of research, as well as developing and deploying infrastructure, services, and applications to support these representations (Lagoze, 2009). Like ROs it is is based on OAI-ORE, allowing a number of resources to be aggregated together. Also like ROs it intends to follow Linked Data principles to support the delivery of its representations of research and link with appropriate repositories, databases and web services.

As the name may suggest the project's initial focus is Chemistry but is intended to be generic enough so that it can be used in any area of research that uses experimentation. The types of resources that make up oreChem's representations of research are quite similar to those proposed in the alignments of ontologies with SWAN across the four different aspects previously described, i.e. online communities, discourse models, bibliographic citations and experimental data. Both include experimental measurements, documents, data and scientists (people).

In part due to the chemistry focus, oreChem has a model of experimental plans that can be instantiated as an experiment one or more times. Once instantiated the experiment can be augmented with data, observations or measurements captured during the experiment, the plan that the experiment follows can also be amended to handle unexpected occurrences. Once an experiment is complete, the outputs can be analysed and used to produce publications and other research outputs, thus completing the experimental lifecycle.

Knowledge Engineering for Experimental Design (KEfED) is a model that allows the representation of the pieces of data that make up a scientific experiment (Burns and Russ, 2009). These pieces of data can take the form of one of three different variable types:

Independent variables Such as parameters set by the experimenter.

Dependent variables Such as measurements observed by the experimenter.

Derived variables Such as calculations from existing variables.

These variables are defined in the context of how they relate with each other as well as with other components within the experiment, namely experimental objects and activities. Primarily, this provides a means of curating information gathered from literature to provide a knowledge base of experiment plans and instantiations of these plans with different sets of independent and dependent variables. However, there is no reason why the

model could not be used by experimenters to help build their own experiment plans and capture their instantiations thereof. To accurately capture the interrelations between dependent and independent variables, each dependent variable is treated as a multi-dimensional space, where each dimension represents one of the independent variables. By adding taxonomies and other structures from external ontologies that describe the organisation of the parameter and measurement space, it becomes possible to perform first-order logical reasoning over a KEfED model. This reasoning may help determine potential relationships between dependent and independent variables, allowing the user to determine new experiments to test these.

This existing alignment model will allow SWAN to not only represent scientific discourse that is extracted from publications but also discourse as it evolves over the course of a Study. Adding some of the concepts proposed by oreChem and KEfED will further enhance the capability of SWAN to capture scientific discourse over the entire experimental life-cycle in an integrated manner. This will assist in the process of producing publications from experiments and using publications as well as the discourse or experimental results to inspire new experiments.

4.2.2.6 SCientific ARticle of The Future (SCARF)

As already set out, the purpose of aligning ontologies across the five areas discussed in sections 4.2.2.1 to 4.2.2.5 is to facilitate the design of a specification that can capture a machine-readable representation of a research publication in the most semantically-rich way possible, namely a SCientific ARticle of The Future (SCARF). The different parties involved in the alignment process have varying motivations for trying to achieve this. SWAN and CiTO have primarily wanted to be able to take existing publications and be able to build a knowledge base from them. ABCDE and SALT have come from the opposite direction of wanting to allow scientists to be able to produce semantically-enhanced publications to submit to journals and conferences. All of these projects believe that the research paper is still the "unit of currency" for scientific publication.

Some projects have a more radical opinion of what will be the future "unit of currency" for scientific publication. The Concept Web Alliance's (CWA) idea of Nano-publication proposes that rather than having many papers that essentially include the same assertion because of the ambiguity of natural language, each assertion should be published individually (Mons and Velterop, 2009). Detecting replication of assertions across huge numbers of natural language papers would be very difficult to automate and would be extremely time-consuming to do by hand. However, where this is done republishing as nano-publications, would be extremely useful. The motivation for Nano-publication's streamlined form of publication is to support reproducible research.

However research may be published in the future, the key goal of all the projects involved with the HCLS Scientific Discourse subtask group is to support reproducible research in a world where there are ever more complex research processes and large datasets. This corresponds with the primary motivation of Research Objects (ROs).

4.2.3 Research Objects for Scientific Discourse

Section 4.1.2 described a number of different types of RO to meet various use cases. Modelling scientific discourse is just another use case for an RO. There are clear comparisons between the *Study* entity as shown in Figure 4.5 and a RO. The *Publication* RO is probably most synonymous with the *Study* entity. Both entities have the goal of allowing those that build them to present their research so that it can be digested by human and machine. However, their models for providing this vary in certain ways.

One major factor that has affected each models' design, is the type of experiment that is its primary inspiration. In the case of SCARF, experiments would typically involve one or more in vivo or in vitro steps, whereas ROs are primarily focussed on experiments that are wholly *in silico*. For this reason much of the focus for SCARF is being able to provide digital representations for these *in vivo* and *in vitro* steps.

The focus of the two models is also somewhat different, for SCARF this focus is scientific discourse. This means that a significant amount of the model concerns itself with this and is the driving force of the experimental lifecycle (from research question to hypothesis and then a study to produce evidence to support research statements). Whereas ROs were originally devised to provide a means for interchange of users' research processes as workflows, whether they be automated or manually driven, ROs puts this at the centre of their model. Therefore the experimental lifecycle is based on the steps required to reproduce the research or research process captured. This could potentially entail the same entities as SCARF, i.e. research, question, hypothesis, etc. but often these steps are more practical.

The rhetorical structure of a scientific paper is central to the architecture of SCARF from coarse through medium to fine-grained rhetorical blocks. ROs architecture is based on a network of associated items. How items can be associated and what properties can be used to associate certain items is described in the ROUM and RODSs. As already described in section 4.1.4, the ROUM provides generic properties to build networks that represent research and research processes. RODSs define more specific properties. These specific properties may be for building a type of RO such as a Publication or Method object. A RODS can be defined for scientific discourse so that a user can capture properties like *motivates* and *hasExperimentalHypothesis*. It is also theoretically possible to write a RODS for capturing rhetorical structure. However, as SCARF has

already spent significant time implementing this, there seems little point in ROs trying to reinvent the wheel.

Despite these differences, being able to perform transformations from one model to another would be advantageous because of their synergy. For ROs being able to extract the research and research processes described within a SCARF would be useful to allow users to take advantage of the features provided by ROs, as discussed in 4.1.1 and the tools that will be provided to manipulate, visualize and enact them. In particular, reuse and re-purposing of this extracted RO could make up part of a new publication, where this augmented RO could be dropped back into a SCARF and used as a resource to build a semantically-rich scientific article. It is not yet clear how to perform such transformations. Both models have a graceful degradation of understanding. At a basic level, ROs have *Interrelations* with predicates taken from hierarchical SKOS *Concepts Schemes*. SCARF uses AO to build sets of annotations for capturing the various aspects, such as rhetorical structure, discourse elements and experimental data. Being able to map between RO *Interrelations* and AO *Annotations* may provide a means for performing these transformations. However, significant work will be needed to determine whether this is possible in a way that does not lose or incorrectly transpose information in the process.

4.3 Question-Answering Systems for e-Research Societies

The myExperiment project allows significant amounts of data to be captured about e-Research contributions, the society that shares them and how they are shared and annotated. This data is stored in a relational database and is queried using the Structured Query Language (SQL). There are a number of reasons why it would be inappropriate to provide myExperiment users a means of querying its data using SQL, in particular requiring the users to learn how to use the query language.

To a novice user there are several problems with using such a language. First, they have to understand the terms, syntax and grammar of the language; even though this is fairly simple it will be unfamiliar to many users who do not come from a Computer Science background. Second, to be able to use this language the user must have intricate knowledge of the structure of the database, i.e. the precise names of all the entities it contains. From scratch this understanding will take significant time to learn and this one reason why it is unusual for any project to provide universal read-only access to its database. Another major reason, as highlighted in section 3.2.1, is because much of the data needs to be kept private.

Chapter 3 has described a number of ways that myExperiment makes it data available whilst ensuring private elements can only be accessed if the user has appropriate permissions:

1. The website.
2. REST API.
3. RDF API and SPARQL endpoint.

The website is the only one of these interfaces designed specifically for users to access the data stored and is an example of both a Graphical User Interface (GUI) and a form-based interface. Four significant types of user interface that can be designed to allow users access to stored data are:

1. Command line
2. Form-based
3. Graphical
4. Natural Language (NL)

Command line interfaces have already been discussed as being both difficult and time-consuming for novices to use. Form-based interfaces are commonly used as a quick and

easy means of providing a user interface to a machine-readable querying language. PHP-MyAdmin[28] is an example of a form-based interface that generates SQL to construct and manipulate a MySQL database. By using this interface, a user does not need to know the terms, syntax or grammar of SQL to complete most tasks. myExperiment uses form-based interfaces for search using the Solr search engine, calls to the REST API[29] and to build SPARQL queries[30]. Higher level, form-based systems also exist providing a greater abstraction over the underlying database making it easier for novice users to access the data they require (Florescu et al., 1998).

GUIs can also simplify the accessing of data held within a database. Like form-based interfaces they allow a lot of information about the structure of the database to be displayed. Unlike form-based interfaces, GUIs require little if any typed input. mSpace is an example of a GUI for a database (m. c. schraefel et al., 2005). One example of mSpace allows users to find pieces of classical music using a graphical, hierarchical, search technique. This allows the music to be broken into categories and subcategories based on *period*, *composer*, *form* and *arrangement*. The user can also view additional information about the categories, such as pictures or composers and descriptions of periods, as well as a sample of music from each category.

Natural Language (NL) and pseudo-NL interfaces to query databases are often referred to as Question-Answering (QA) systems. A pseudo-NL interface is one that is only capable of processing a restricted subset of NL phrases. These are often a useful compromise between a full NL system and a formal query language. They allow users to build queries in a form that is intuitive to them, i.e. as questions, but with a less complex mechanism required for transcribing them into a machine-readable queries. However, the disadvantage is that without training more questions are liable to fail to return a result. Fortunately, this training usually occurs incidentally by users reformulating their questions until they get a result. As long as there is sufficient flexibility in the questions that can be answered successfully, the user may well adapt to this pseudo-NL specification rather than lose patience with the interface.

QA systems are quite different from both form-based interfaces and GUIs. Considerably more effort is required to process the user's input to determine what results should be returned. The first part of this is how the natural language is parsed to produce a machine-readable representation. To achieve this set of rules, known as a grammar, is required to define how natural language is parsed.

[28] `http://www.phpmyadmin.net/`
[29] `http://www.myexperiment.org/mashup/api`
[30] `http://rdf.myexperiment.org`

4.3.1 Natural Language Processing

Natural Language Processing (NLP) is the process of parsing and evaluating natural language to produce machine-readable representation of grammatical structure and semantic content of the text. Chomsky showed that a finite-state grammar that uses chaining Markov models is not sufficient to map all English NL sentences (Chomsky, 1956). He stated that a context-free, phrase-structure grammar can represent a lot more sentences. He also considered that many NL sentences share basically the same meaning, e.g.

- The sandwich was eaten by the man.
- The man ate the sandwich.

In this example both sentences describe the same concept, i.e. they have the same action, actor and thing being acted upon. The difference between these sentences is that one is written in the passive voice and the other is written in the active voice. For this reason, Chomsky proposed the idea of transformational rules. One of these transformational rules can convert the first sentence into the second. Using Transformational rules means that phrase structure rules are only needed to define basic sentences, from which transformational rules can the generate the rest. A further advantage of transformational rules is that it provides two levels of representation for a sentence:

Deep Structure This is the most basic form of the sentence.

Surface Structure Derived from a deep structure through the application of transformational rules.

Having this additional level of representation, known as the deep structure, which effectively stores the semantic meaning of the sentence, is useful within the context of a QA system as it can make it easier to process and determine the correct response.

The addition of transformational rules to a phrase structure grammar has come to be known as a transformational grammar. Since Chomsky's seminal paper many grammatical theories have been developed to try and provide better ways to represent a transformational grammar with the ultimate goal of producing a universal grammar theory applicable across all natural languages (Chomsky, 1956). These grammars include:

- Augmented Transition Network (ATN) (Woods, 1970).
- Generalized Phrase Structure Grammar (GPSG) (Gazdar et al., 1985).
- Head-driven Phrase Structure Grammar (HPSG).

- Principle-based Grammars.
- Lexical Functional Grammar (LFG) (Kaplan and Bresnan, 1982).
- Unification-based grammar.
- Tree Adjoining Grammar (TAG) (Joshi et al., 1997).

Each of these grammars can be implemented as part of a NL parser with the same outcome, a machine-readable representation of some natural language. The interpretation of this natural language by applying these grammars may differ but the arbiter of which interpretation is correct is ultimately the person who provided the original natural language. Although much effort has gone into modelling natural language, their are certain quirks that are difficult to encompass within logical rules. Statistical analysis is a means of handling some of these quirks.

Corpora are collections of natural language text and can be used for statistical modelling natural language and assessing the validity of a particular grammar or set of rules. Parsing these corpora produces annotated treebanks, such as the Penn Treebank[31] or Susanne Corpus[32] that have Parts Of Speech (POS) tags for each word and a grammatical tree structure for each sentence (Marcus et al., 1994), (Sampson, 1995). These treebanks allow probabilistic models to be generated for help determine a sentence's structure. Combining these probabilistic models with a transformational grammar provide a hybrid approach to NL processing.

The original concept for the hybrid approach to NLP was proposed by the theory of Probabilistic Context-Free Grammars (PCFG). This technique allows several parse trees of a sentence to be generated that have varying final probabilities (Booth, 1969). However, it was not until the early 1990s, when sufficiently large annotated treebanks could be produced that it was possible to implement such an approach. Many modern day NL parsers, such as SPATTER, Collins Parser, Stanford Parser and Minipar use similar approaches to parse natural language (Magerman, 1995), (Collins, 1996), (Klein and Manning, 2003), (Lin, 1998).

NL parsers are only the first step to converting an NL question into a machine-readable query. There are many other tools required to interpret the semantic meaning of the question and deal with the inherent issues with natural language such as domain opacity, co-reference resolution, anaphora, entity recognition, etc.

4.3.2 Inherent Issues with QA Systems

The first problem that a user might encounter when using a QA system is the domain opacity. Even if the QA system is for a restricted domain, such as e-Research in a

[31] http://www.cis.upenn.edu/~{}treebank/
[32] http://www.grsampson.net/RSue.html

specific field, the user is unlikely to know precisely how the domain is represented and therefore how to express their questions. Form-based interfaces and GUIs have less of a problem with this as options and sometimes even the structure of domain are inherently displayed within the interface. To minimise domain opacity as a problem, QA systems need to interpret as many different constructions of questions as is feasible. They must also be flexible with the vocabulary that can be used as different users may use different synonyms.

Inevitably, some questions will not be interpretable and in this case it is important that a user receive an appropriate error messages so they can understand why no answer could be determined and more accurately assess how their question needs to be modified to get the answer they require:

1. The question submitted had either grammatical or spelling defects.
2. The question does not have any specific spelling or grammatical defects but it could not be mapped to a syntactical representation.
3. A syntactical representation for the question was successfully produced but no semantic meaning could be extracted from it. Chomsky gives the now famous example of "Colorless green ideas sleep furiously" as a grammatical sentence with no semantic meaning.
4. The question's semantic meaning can be interpreted but the system is incapable of transcribing the question into a machine-readable query.
5. The database to which the machine-readable query is applied does not contain any data relevant to that query. If the QA system subscribes to an Open World Assumption (OWA), it cannot say that no results exists, just that it is not aware of any.

For the first three of these, it is possible for the user to reformulate the question and potentially get the answer they require. For the last two it is unlikely or impossible that any reworking of the question would produce a suitable answer. This may well be because the type of question asked is outside the scope of the system. It is therefore important that error messages for these make the user aware of the scope of the system, so they do not pointlessly make repeated attempts, which will invariably lead to frustration and disillusionment with the system (Shneiderman, 1980).

Co-reference resolution has already been discussed as a problem in the world of Linked Data, see section 3.1.7. It is equally as big a problem in NL processing. Determining that two entities in a piece of text are in fact the same thing can be a difficult problem to resolve, especially when they occur in different sentences or even different paragraphs. A similar approach as Linked Data is often used to resolve this problem. Once a text is

parsed and a semantic meaning has been determined, the semantic relationships for each of the entities can be compared. If there is a sufficient number of shared relationships and the two relationship sets do not show any inconsistency, then the two entities can be considered to be the same. However, this often requires some conceptual representation of the domain, e.g. an ontology, to be able to check for inconsistency. Fortunately, questions are usually only made of a single sentence and are therefore unlikely to contain two different entities with the same name. If they do, the grammar of the sentence will most likely be able to determine this. However, pronouns add complication to co-reference resolution that may not be resolved by the question's grammar.

Anaphora is where pronouns, both personal and impersonal, are used to refer to people or things that have been mentioned previously, e.g. "John has a dog. It was given to him by his dad." There are three pronouns in this example, *it*, *him* and *his*. Most people would probably assume that John is male, with this assumption the pronouns *him* and *his* can be interpreted as referring to John. As no reference is made to the gender of the dog, in general it would be assumed that *it* refers to the dog. This demonstrates a semantic analysis of the sentence. *John* is not just a proper noun but a male name, *a dog* is a creature that has a gender but can be referred to as *it* if no gender is specified. Even with this semantic analysis it is still sometimes impossible to resolve anaphora, e.g. "The page gave the knight his sword." "Whose sword does the knight have?" is an impossible question to answer with certainty. It is possible to determine that both the page and knight are male, therefore the possessive pronoun *his* could refer to either of them. In this case, a QA system would have to return an error message asking the user to rephrase the question to disambiguate this *his*. For example, "The page gave the knight's sword to him." "Whose sword does the knight have?"

Sometimes it easy to determine the entities of a sentence as they are single words like the previous examples concerning anaphora. However, entities may not just be two or three words long but whole phrases or even sentences. myExperiment's workflows often have long titles such as "KEGG pathways common to both QTL and microarray based investigations". Such a title could make it difficult to parse a sentence because the phrase is not relevant to the semantic meaning of the sentence beyond it being an entity that represents a workflow. Therefore, it is often necessary to determine that this is an entity before parsing the sentence to generate POS tags and a grammatical tree.

Often a user's recall of a workflow may be imperfect, making the task of recognizing and labelling the entity more difficult. To overcome this problem, entity recognition algorithms need to be a bit more sophisticated. One way of achieving this is through n-grams. N-grams work by breaking an entity down into its individual words and then searching for occurrences of two or more of these words appearing in close proximity to each other. The number, closeness and ordering of these words determine a probability of whether there is an entity match. A threshold can then be applied to specify how high the probability of a match needs to be for it to be accepted. The sophistication of

the entity recognition can be further increased by determining the stems and synonyms of words so that even if the word is not an exact match it can still add weight to the probability of an entity match. Entity recognition is a key component for determining the semantic meaning of a sentence or question.

The ability to answer NL questions may give the impression to the user that a system is intelligent, with common sense and human reasoning abilities (Androutsopoulos et al., 1995). This may lead to an exaggerated sense of disappointment with a system when it does not live up to these expectations by not returning an answer or by returning one that is unexpected, due to any of the issues previously discussed. For all of these issues, QA systems provide the user with an interface where they do not have to follow very strict grammatical, and often non-intuitive rules, that languages such as SQL and SPARQL require. Instead they allow a question to be expressed much more flexibly, in a language the user understands. This increased flexibility allows the user to express questions that a more structured form-based interface may not allow. Form-based interfaces and GUIs also often struggle when a user query involves negation, quantification or temporal relations (Cohen, 1992).

4.3.3 A Brief History of QA Systems

QA Systems in part have their origins in the conversational systems developed in late 1950s and 60s in an effort to design a computer system that could pass the Turing test. In basic terms, the Turing test is a test to see whether a computing system can convince a human that they are talking to another human. It is an attempt to frame a real-world problem to the question "Can machines think?" (Turing, 1950). To test this empirically, Turing proposed a scenario called "The Imitation Game".

The Imitation Game involves three players, a man, a woman and a human interrogator of unspecified gender. The interrogator can not see the man or the woman but can ask them questions and receive back type-written answers. The goal for the interrogator is to determine, which of the other two players is the woman. The goal of the woman and the man is to convince the interrogator they are the woman whether they are or not. Through repeating this experiment, a control percentage for how often an interrogator is deceived can be determined. If the man is now replaced by a computer system and the game is replayed, if the interrogator is deceived as often, then the computer system can be considered to have passed the Turing test.

One of the most well-known of these earlier conversational systems was called ELIZA (Weizenbaum, 1966). It was designed to attempt to pass the Turing test by trying to convince a human that it was a psychiatrist. Written in Lisp, ELIZA would examine a sentence input by a user to find keywords. Taking the set of keywords found and ranking them, ELIZA could decompose the sentence, perform the correct transformation,

e.g. first person pronouns to second person pronouns and vice-versa, before using a reassembly rule to build a reply. By using this technique ELIZA could come across as a very convincing psychiatrist. However because it uses non-keyword parts of the sentence to make its response seem as though it has understood the sentence, if the human user inputs garbage for these parts, ELIZA will actually respond using this garbage, making it easy for the user to detect they are not conversing with a human.

The first QA systems such as *The Oracle*, the *General Enquirer*, *Baseball*, the *Specific Question Answerer*, used similar although slightly less shallow approaches to ELIZA to try to determine what results they should return to the user (Phillips, 1960), (Stone et al., 1962), (Green Jr. et al., 1963), (Black, 1964). These systems could perform basic syntactic analysis using a minimal set of grammatical rules over simple and usually domain-specific questions. In general, this syntactic analysis would then be used to crudely build machine-readable queries to produce a set of results that could be returned to the user as an answer. What was queried varied between, logical statements, previously entered sentences and actual databases storing information about the domain.

As grammatical theories have developed to support the parsing of many more English sentences, it has been possible to develop QA Systems that use parsers that make use of these grammars. Lunar was one of the first systems to use a transformational grammar parser, in its case one based on ATNs (Woods et al., 1972). It queried a database about moon rocks with queries generated from the parser's analysis. Ladder also used a parser, which it called Lifer, that was capable of performing spelling corrections and providing error messages on failure (Hendrix et al., 1978). Ladder was also capable of a technique called ellipsis, that could keep track of at least the previous question, allowing users to ask a stream of questions without having to redefine what the question was in reference to each time.

Chat-80 is a QA system that queries over a Prolog database of logical statements. Rather than using a transformational grammar parser, it used one based on extraposition grammar, a specialised definite clause grammar, a grammar for representing first-order logic, capable of left extraposition, one of the features of transformational rules. Janus also used a parser using a logic-based grammar known as Montague grammar (Hinrichs, 1988). Janus was one of the first examples to show how several applications, in this case parsers and translators, are needed to produce an output that could be mapped to a sufficiently accurate and detailed machine-readable query.

FALCON is a more modern QA system and an early user of a statistical rather than transformational or logic-based grammar (Harabagiu et al., 2000). Like Janus and many of its peers, FALCON realised that converting an NL question into a machine-readable query was a complex process that required multiple tools to complete the process, due to the number of inherent issues described in 4.3.2.

4.3.4 Tools for NLP

Beyond NL parsers there are still many tools needed to take an NL question and convert it to a machine-readable query that can generate results that will satisfy the user. Many of these tools are designed to tackle the inherent issues discussed previously. An example of such a tool is a thesaurus; this allows links to be constructed between synonymous words. Such a tool is essential for a QA system as it allows different terminology to be used, but when it comes down to distilling the question, the same machine-readable query can be determined. Thesauruses were originally written as books where users could look up words and find synonyms along with a brief description of what the word meant. This meaning was essential because a single word could have many different meanings, e.g. minute, meaning both sixty seconds and very small, and the user would want to ensure they had found a synonym for the appropriate meaning. Representing a thesaurus electronically, in a relational database or similar, is considerably more efficient with only one word meaning needed for each group of synonyms.

WordNet[33] is an example of an online lexical database (Fellbaum et al., 1998). Such a system replicates the functionality of the thesaurus, where a form is used to look up synonyms for a particular word. By having a primary entity known as a *lexical concept*, WordNet can however go further than this. Each lexical concept has a type, namely a noun, verb, adjective, adjective satellite or adverb, it also has one or more words and commonly one but sometimes more definitions. Each *lexical concept* can be associated with other *lexical concepts* through a number of different properties, namely *antonym*, *hyponym*, *hypernym*, *meronym*, *holonym*, *caused by*, *groups with*, *participle of*, *pertains to*, *see also*. This allows, in the case of *antonym* for users to find words that mean the opposite of the word they searched for. *hyponym* and *hypernym*, and to a lesser extent *meronym* and *holonym* are equivalent to the *narrower* and *broader* properties used in SKOS, see section 3.1.6.2. They allow hierarchies of lexical concepts to be built. For a QA system this is especially useful, as a question may use a very specific term that is not represented within the domain but, if the thesaurus can resolve it to a more general term, a query can still be generated.

It is not uncommon that a term may not occur in a standard thesaurus or even one constructed for a specific domain. OpenThesaurus is a project, albeit designed for German rather than English, that provides users with a system to collaboratively construct their own thesaurus (Naber, 2004). A user of the system can create, discover and edit synonym sets. The editing history of these sets of synonyms is recorded allowing administrators to rollback errors and resolve any inconsistencies created by users. This is similar to the Wiktionary project[34], that allows users to collaboratively build dictionary entries, including references to synonyms, within a wiki. Probably due in part to the success of

[33] `http://wordnet.princeton.edu/`
[34] `http://en.wiktionary.org/wiki/Wiktionary:Main_Page`

its sister Wikipedia project, Wiktionary has had a considerably higher number of users and contributors. The DBPedia project has also been taking information captured by Wiktionary to produce a formal representation for it. However, in some ways neither of these projects meet the goal of either a specific community of users building a thesaurus for their terminology or a single user being able to build a thesaurus so they can map terms they use to the terms that will be recognised by the QA system.

WordNet, OpenThesaurus and Wiktionary only deal with a single issue of natural language. There are many other issues to deal with and different applications and approaches for tackling them. For this reason the General Architecture for Text Engineering (GATE) has built an architecture to allow developers to piece together NL processing components, using a graphical IDE, to build a workflow that can be run over some natural language input (Cunningham, 1999). For a QA system, this can theoretically be used to produce a workflow to go from a NL question to a machine-readable query. These components support tasks such as named entity recognition, co-reference (e.g. pronoun) resolution, template element construction, relation construction and scenario template production. They are provided through the Collection of REusable Objects for Language Engineering (CREOLE), which is essentially a library of modules that encapsulate pre-existing or user-defined tools and resources. These can take one of three forms:

Language Resources (LRs) such as lexicon, corpora, annotated treebanks, gazetteers and ontologies.

Processing Resources (PRs) such as parsers, generators and n-gram modelers.

Visual Resources (VRs) for visualizing and editing the LR and PR components.

A Nearly-New Information Extraction system (ANNIE) is an example of a tool developed using GATE, by combining CREOLE modules (Cunningham et al., 2007). ANNIE can perform all the tasks previously described for components. One particular task it can be used for is to build more complete entities. To do this, it first takes a NL sentence and tokenizes it. There are five different types of token, namely, word, number, symbol, punctuation and space. Each of these types have further subtypes. Next this tokenized sentence is searched for named entities taken from a gazetteer. Finally, based on the named entity found, it selects an appropriate grammatical rule in an attempt to build the complete entity, e.g. if the search using the gazetteer found the term “US Dollars”, it would invoke the associated grammatical rule that checks preceding tokens to see if they are a number or a word that represents a number. If they are this token is added to the existing entity to make it more complete.

Java Annotation Patterns Engine (JAPE) is a tool for building rules for annotating documents with tags. It can be used by a tool like ANNIE to perform information

extraction from text (Thakker et al., 2009). JAPE will have first run a tokenizer, sentence splitter, POS tagger and gazetteer over the text. This will generate tags for entities, phrases and words and assign basic annotations such as parts-of-speech to words or attributes of the identified entities. Using these tags and associated annotations, JAPE rules can be applied using a transducer to generate further annotations for these tags. Multiple transducers can be applied one after another to build up ever more detailed information about the text.

One of the most interesting things about GATE is that when a system, i.e. a workflow of assembled components, is executed all the communication between the modules goes through the GATE document manager to facilitate a common API for communication. This allows the document manager to act as a repository and store all the information that is collected about processed text in a standard format, specifically the TIPSTER annotation format (Grishman, 1997). From the perspective of a QA system having richly annotated NL questions is useful to evaluate the type of questions being asked, so the system can be tuned to ensure it has a high successful response rate.

4.3.5 Web and Semantic Web based QA Systems

Since the advent of the Web, the number of interfaces to databases has increased significantly. Also the numbers using these interfaces are a lot greater and made of a wide spectrum of users, some of whom may have very little understanding of the domain to which they are interfacing or even how to go about using such an interface. For QA systems, this lack of understanding is particularly important. It makes it critical that the user is informed about the capabilities of the system before they start using it and when / if they fail to get the responses they expected, they are presented with informative error messages.

The START system according to its website[35] was the first web-based QA system and has operated continuously since December 1993. The current incarnation of START uses a system called Omnibase (Katz et al., 2002). It is a data model designed to store the heterogeneous data that can be found on the web. Omnibase's data model stores records of the form "object-property-value", to represent relationships, much like the RDF model of triples. Many questions that might be submitted to a QA system can have an answer record encoded in this way. Figure 4.1 shows how the question "What is the capital of Spain?" might be encoded. Make the assumption that "what" can

	Object	Property	Value
Question	What	is the capital of	Spain
Answer	Madrid	is the capital of	Spain

TABLE 4.1: START "Object-Property-Value" Encoded Question and Answer

[35] `http://start.csail.mit.edu/`

be determined as an unknown. Any answer records that include Spain as the value and have a property that represents the former being the capital of the latter can be returned. To be able to achieve this it is essential to be able to determine when two different pieces of text represent the same thing. At a basic level this may just mean being able to determine synonymous words such as 'tallest' and 'highest', however often more sophisticated evaluation will be required. Further to this, answer records may be inverted, e.g. 'Spain' 'has the capital' 'Madrid'. This requires the database to be able to associate these inverse properties so that the system can perform searches where the unknown is the value not the object. Both of these techniques should increase the number of times the system can return a response but unless these associations of analogous and inverted properties are highly accurate, it is likely that the percentage of erroneous responses will also increase.

Web-based QA systems can be employed to work within fairly structured domains, such as the book, film and music industry. Although this structure tends to limit the number of properties that need to be handled they do contain large numbers of named entities. In particular these named entities are often difficult to isolate from the question as they are a phrase if not a sentence in their own right. Taking the music industry as an example, the question "Who recorded standing on the shoulder of giants?" is a grammatical sentence but to those unable to identify the named entity it would appear slightly odd semantically. For a system like START it would first have to identify the likely domain of the question. The property 'recorded' suggests that the domain could be films but more likely music. The START system would then go to the web source(s) it has for that domain and search for a matching named entity. First it will look for the whole phrase it has initially identified as the value, i.e. "standing on the shoulder of giants". Usually if this is put unquoted into a search engine, it will return both exact matches and those that match one or more key words. With this data START can evaluate whether there is a sufficiently accurate match and if so what is the complete named entity. The web page representing this resource can then be parsed to extract information about the entity. This will hopefully provide an "object-property-value" that can be used to determine the value of 'who'.

Omnibase is not the only system to try to integrate heterogeneous web resources so they can be used by a common interface: Araneus (Atzeni et al., 1997) and Ariadne (Knoblock et al., 2001) use an SQL database to store the data from different web sources in separate tables and then use tools to integrate these tables to produce one integrated web resource, where data from several sources can be pulled together to answer a query. The main difference between these systems and Omnibase is its relational data model, that makes data integration between user questions and online content more intuitive. However, it does not support questions where multiple sources are needed.

AskMSR is another web-based system that uses search engines to try to find answers to the questions (Brill et al., 2002). It focusses on being able to determine analogous

sentence templates, i.e. those that have the same predicate or property. The system achieves this by taking the NL question submitted and rewriting it using all suitable declarative answer templates, that have an unknown, which is the answer to the question. These rewrites are weighted using heuristics based on their resemblance to the original question. These rewrites are then submitted to a search engine that looks over the annotated corpora. From the results it returns, n-grams are extracted that might be the answer to the question. These n-grams are then weighted based on the rewrites weighting and summed together to produce totals for each different n-gram. A further re-weighting is performed dependent on whether the answer matches the type expected, e.g. 'who' in the question would expect an n-gram for a person, group or organisation. Finally, overlapping n-grams are also merged and their totals summed. The list of n-grams are then given percentage probabilities and returned to the user. The Web QASystem uses a similar technique called Query Formulation to try to find all the possible forms of answer phrase (Yousefi and Kosseim, 2006).

Some web search engines claim to allow the user to submit an NL question and it will find an answer, such as Answers.com[36] and Ask Jeeves[37]. The former of these relies on crowd sourcing where users can both ask and answer questions. This can potentially be very effective in answering complex NL questions, as humans have an innate ability to understand them. However, it does rely on the answerer having the knowledge to answer correctly and them not deliberately answering the question incorrectly. For this Answers.com provides a trust ratings for answerers[38]. However, this rating could potentially be increased by a user by using dummy accounts to assign "Trust Points" to the primary account.

Ask Jeeves does not actually fully evaluate the question asked. Like many search engine it searches for keywords in the query and uses n-grams to weight those results where keywords appear closer to each other. Other techniques such a word-stemming, e.g. allowing the keyword 'corpora' to also match 'corpus', are also used. However, because the search engine is only returning Web pages that may contain the answer, determining a full semantic representation of the question would be excessive. This would only be useful if every currently indexed Web page had all its text similarly evaluated. However, in specialist cases, search engines do parse queries more carefully, such as maths, dictionary, weather, time and conversion questions.

Wolfram Alpha[39] is an example of a search engine or as it calls itself a "computational knowledge engine" that has many macros for determining and evaluating specialist case questions. These macros have been written for 29 different topic areas, including mathematics, most scientific, social science, engineering and humanities fields, dates,

[36] `http://www.answers.com`
[37] `http://uk.ask.com/`
[38] `http://wiki.answers.com/help/trust_points`
[39] `http://www.wolframalpha.com/`

colours, weather, transportation, organizations, etc. The Wolfram Alpha framework allows macros to be continually added to increase the number of questions it is capable of answering. When a question is submitted the input is interpreted. Sometimes this is just determining a keyword, other times it might determine a phrase that can be put into a template, e.g. "What is the largest country in Europe?" will use the "largest country" template with the parameter that it is "in Europe". This keyword or template will cause one or more macros to run. In the case of the previous example, three macros run, returning lists of the five largest countries by total area, Gross Domestic Product (GDP) and population respectively. If the user had chosen to ask the more specific question "What is the largest country in Europe by GDP?", this would have created a template with two parameters, "in Europe" and "by GDP" and would have caused only one macro to fire.

Much attention has been paid to the Wolfram Alpha user interface. In the previous example of the list of countries, each list can be expanded to give the rankings of smaller countries. Also depending on the macro, different formats of results will be returned, such as maps, graphs, diagrams, tables or paragraphs of text. Unlike some QA systems, Wolfram Alpha does not directly answer the question posed but provides concise information that can be used by the user to determine the answer for themselves. From a user's perspective this is often more useful than providing a single word answer, as it provides additional information to verify the answer is correct. A further feature of Wolfram Alpha is that it provides the sources from which the macros found their information, so that if the users so wishes, they can research the answer in further detail.

A drawback of Wolfram Alpha is that the coverage of its macros is not complete. This can lead to what may seem like fairly simple questions failing to return a satisfactory response. An example of this is the question, "Who has been the tallest president of the US?" Figure 4.6 shows how Wolfram Alpha will return information about the president of the US for this question but none of this can be disseminated to determine the answer. Alternatively, if the same question is submitted to Google, the first result is a Wikipedia page containing a table ranking the US presidents in order of height, see Figure 4.7. At present, Wolfram Alpha is also quite limited on the complexity of questions it can answer. Composite queries such as "What is the largest country in Europe or Africa?" or those that use negation such as "Which countries are not members of the European Union?" fail to return correct answers. Some temporal questions return correct answers, when a specific defined chronology exists, such as "Who was the US president in 1974?" but others such as "Which countries were members of the European Union in 1974?" do not. Although Wolfram Alpha can struggle with more complex questions, it can be a very useful resource if a user has a simple specific question.

One of the critical requirements for being able to tackle complex questions is having a detailed model of the domain being queried. As described in section 3.1, SW technologies and OWL in particular are a well developed means of producing specification

FIGURE 4.6: Screen shot of Wolfram Alpha answering the question "Who has been the tallest president of the US?" (As of 15/12/2010)

for describing a domain. A number of modern QA systems have built upon these technologies. NaLiX is a QA system that uses XQuery queries over a database of XML

Heights of Presidents of the United States and presidential candidates

From Wikipedia, the free encyclopedia

A record of the heights of the Presidents of the United States and presidential candidates is useful for evaluating what role, if any, height plays in presidential elections. Some observers have noted that the taller of the two major-party candidates tends to prevail, due to the public's apparent preference for taller candidates.[1]

The tallest U.S. Presidents were Abraham Lincoln and Lyndon B. Johnson at 6 ft 4 in (193 cm), while the shortest was James Madison at 5 ft 4 in (163 cm).

Barack Obama, the current President, is 6 ft 1 in (185 cm),[2][3] and Joe Biden, the current Vice-President, is 6 ft 0 in (183 cm).[4]

Presidents, like the U.S. population, have grown taller over time.

Contents [hide]

1 U.S. Presidents by height order
2 Trends
3 The taller candidate wins?
4 Comparative table of heights of United States presidential candidates
5 Statistical breakdown
6 Extremes
7 See also
8 Notes
9 References
10 External links

U.S. Presidents by height order [edit]

Rank	President	Height	
1	Abraham Lincoln[5][6] Lyndon B. Johnson[7][8]	6 ft 4 in	193 cm
3	Thomas Jefferson[6][9]	6 ft 2½ in	189 cm
4	Franklin D. Roosevelt[6] George H. W. Bush[6][10] Bill Clinton[6][11][10] Andrew Jackson[6][12]	6 ft 2 in	188 cm

Abraham Lincoln and....

FIGURE 4.7: Screen shot of height of US presidents Wikipedia page (As of 15/12/2010)

documents to find its answers (Boag et al., 2007), (Li et al., 2005). Although XML is not specifically a SW technology, it is a standard format for encoding SW data. NaLiX can support questions that include comparisons, simple negation, quantifications, nesting, aggregations and sorting. NaLiX uses the statistical parser Minipar to generate a syntactic parse tree for a submitted question. The system then has three steps to take this parse tree and convert it into a valid XQuery expression:

Classification Each word or phrase is classified as a token or marker depending on whether it is a XQuery component or not.

Validation Checks to see whether the classified parse tree can map to an XQuery. It also check whether the elements, attributes and values in the user's question can be found in the database. If either of these tasks cannot be completed an appropriate error message is returned to the user.

Translation The NL constructions of the parse tree are used to generate an appropriate XQuery expression. The concepts of token attachment, token relationship and core tokens are used to achieve this mapping.

Later QA systems that have used specific SW technologies include GINSENG and TeLQAS (Bernstein et al., 2006), (Hejazi et al., 2003). GINSENG (Guided Input Natural language Search ENGine) is one of the first systems that attempted to provide a NL / pseudo-NL search engine using SW technologies. It guided the user when they entered words for their query to try to ensure it got an NL question that used known words and structure. This helped overcome one of the greatest problems with QA systems, namely domain opacity. Having a guided input meant the NL processing and conversion techniques required to produce a machine-readable query could be made much simpler.

The GINSENG architecture was made up of three main parts. The first was a multi-level grammar, which contained about 120 static domain-independent rules and a dynamic grammar rule for every entity contained within the loaded ontologies. That provided the pop-up options for the guided input interface. The second part was the incremental parser, as well as specifying all the parsable sentences it also provided the information required to generate the machine-readable (SPARQL) queries. The final part was Jena's SPARQL interpreter that executed the query and returned the result.

GINSENG had a few limitations and issues. First, any queries that could not be transcribed into SPARQL could not be handled, such as "How many cities named Austin are there in the USA?" Second, the grammar of GINSENG was limited, meaning that a perfectly acceptable NL question may not be understood. Grammatical rules were static, therefore if any common structures were missing they had to be added manually. A further problem was that a user had to use certain terms, if they tried to use a synonym the query would generally fail.

TeLQAS (Telecommunication Literature Question-Answering System) was a QA system developed for answering simple English questions in the telecommunication domain. The system had three main subsystems, off-line, ontology and online. The off-line system used a focussed crawler to gather documents relevant to the domain, which were automatically categorized and indexed. TeLQAS incorporated a domain ontology. In the most part this was constructed by experts but parts could be auto-generated by the off-line system as it categorised and indexed documents. The online system took a user's NL question and parsed it to produce a 'plausible' question, that it sent to the inference engine. If an answer could be found in the knowledge base without any inference then it was returned, otherwise new 'plausible' questions were inferred using various transforms in a recursive process. If answers were found in the knowledge base for these new plausible questions, then the values for plausibility of the question and the support for the answer were combined to determine the certainty of whether the answer was correct. TeLQAS realised that the domain ontology and knowledge base may not contain enough information to generate a plausible question that can find an answer. In this case the system would just perform a keyword search over the documents indexed in the system.

One thing common across all these web and SW-based systems is the consideration given to the user interface. Providing guided input, producing structured and visual results, intermediate output, links to sources, fail over to more basic query methods and appropriate error messages, are all techniques to assist the user in asking a question that will give an answer or allowing the user to assess whether that answer is correct. As search engines have become commonplace, the average user has become used to tailoring their queries to improve the chances of finding the particular page they want. A similar tailoring process can occur when users become accustomed to how a particular QA system works. Users will use grammatical structures they know to work and endeavour to use the simplest sentence possible to express their question. Although many of the QA systems described in this section have striven to be able to answer complex questions, many questions are often quite basic. As observed with Wolfram Alpha, extra complexity is often added to questions through logical operators, e.g. 'and', 'or' and 'not' and temporal clauses. As long as these modifiers can be isolated it is possible to build the basic machine-readable queries and then combine or refine them to represent more complex composite queries.

Another commonality across these Web and SW-based QA systems as well as some of the QA systems described in section 4.3.3 is the need for a well-defined model of the domain(s) so that once the NL question is parsed it is possible to generate a machine-readable query that will succeed in retrieving some useful results. As knowledge representation technologies, in particular those for the Semantic Web, have improved, being able to build these well-defined models and associated knowledge bases has become easier.

4.3.6 A Potential Question-Answering (QA) System for myExperiment

As already described myExperiment has a number of ways that a user can query its data or discover entities. There are simple shallow ways for the user to try to find what they are looking for, such as entering keywords in to the Solr search engine or by scrolling through lists of items, ranked by the date created or how popular they are. There are more sophisticated ways of searching, such as using the SPARQL endpoint or the query syntax provided by the Solr search engine. Solr's query syntax is based on the Lucene query parser syntax[40] to allow users to add logical operators to find items that meet a number of criteria, e.g. listing 4.2 is the query that could be used in myExperiment to find all Taverna 1 and 2 workflows tagged with BLAST or bio2rdf.

```
(kind:"Taverna 1" OR kind:"Taverna 2") AND (tag_list:BLAST OR tag_list:bio2rdf)
\label{lst:solrsearch}
```

Listing 4.2: Example Solr query using logical operators

[40] http://lucene.apache.org/java/2_3_2/queryparsersyntax.html

However, both SPARQL and Solr's query syntax are very specific and not particularly error tolerant. As already discussed in section 3.2.2, for a project of which the user base is not made up of computer scientists, expecting users to learn both myExperiment's specification of the domain and the syntax for the query language, is not a sensible approach and may lead to users shying away from using myExperiment altogether. The burden has to be on the developer to provide the user with an interface they can get to grips with quickly. As part of the Repository Enhancement programme funding, myExperiment has been working on developing a faceted browsing tool. This allows users to filter the workflows they are looking for, specifying values for one or more facets. Figure 4.8 is a screen shot of faceted browsing tool performing the same query as described in listing 4.2. The left hand menu allows users to choose on which facet they want to filter. When a user checks a box, that value is filtered in as can be seen in the screen shot, where the first result is a Taverna 1 workflow tagged with BLAST, which is consistent with those two boxes being checked. Much consideration was given to how different facets and values should be combined. It was decided that, for the initial implementation, to keep this simple by or-ing values and and-ing facets, meaning that it is not possible to get the interface to find all workflows that are either Taverna 1 or tagged with the word BLAST. As users become accustomed to the system and provide feedback it will be possible to evolve the interface to allow different types of query like the one previously described or that have negation or temporal clauses. However, care needs to be taken to ensure that the interface does not become too complicated, as the whole purpose of providing a simple means for filtering down workflows will be lost.

QA systems do not suffer the same problem as form-based, faceted browsing, as adding functionality to perform additional and more complex queries does not appear in the user interface. Therefore, in cases where myExperiment users want to perform either complex faceted browsing queries or queries which require more sophisticated semantics to be understood, they only have to deal with a single text input box.

myExperiment already has a number of the components required to build a QA system. It has a well-defined model in the form of the myExperiment ontology. This is accompanied by a structured knowledge base, the RDF API, that can be queried using a SPARQL endpoint. This leaves only a few more components needing to be built. The most major component required is an application that can accurately parse NL questions that myExperiment users may submit to a QA system and produce a machine-readable semantic representation of the sentence. A tool such as GATE could be used to build such an application. However, it would need a number of language resources, to be able to perform this task.

One language resource required is gazetteers, so that the system can perform named entity recognition. The SPARQL endpoint provides a means of obtaining lists of tags, user and group names, workflow, file or pack titles, etc., all of which may be named entities that a user might include in a question.

FIGURE 4.8: Screen shot of faceted browsing on myExperiment (As of 17/12/2010)

Another language resource required is a thesaurus. This is needed to be able to map nouns and verbs used in a user's question to classes and properties used in myExperiment ontology. The SPARQL endpoint can be queried for the names of classes and properties in the ontology but mapping to potential terms that a user might use requires an external resource. WordNet would be a good choice for this, as it is already a Linked Data project and therefore myExperiment could generate a linkset between its domain concepts and WordNet's lexical concepts. When a term in a user's question does not map directly to a domain concept, appropriate links to WordNet lexical concepts should be followed to

see if they contain the term used in the question. The process of generating the linkset should be fairly automatic, as the domain concept names found in myExperiment could be used to generate a list of SPARQL queries that could be submitted to the WordNet SPARQL endpoint[41]. The results returned from these queries would have to be evaluated by a human to determine which lexical concepts should be mapped to. It would also indicate which domain concepts within the ontology do not have mappings, for which the QA system developer would have to go and find these lexical concepts manually or write their own.

With these language resources it is possible to map much of a user's question to semantic machine-readable representation. Table 4.2 shows an example question with semantic machine-readable mappings. There are still a number of things that need to be done

How many	workflows mecontrib:Workflow	are	owned by sioc:has_owner	Paul Fisher myexp:users/43	and	tagged with meannot:has-tag	BLAST myexp:tags/49

TABLE 4.2: Example question mapped to myExperiment entities and concepts

to produce a SPARQL query that can retrieve the information needed to answer the question. One of the most critical is determining how to combine the machine readable entities into a single query. The parse tree generated by the NL parser in the GATE generated parsing application should be able to provide some assistance with this. Figure 4.9 is the parse tree for the question shown in table 4.2 just down to the level of phrases rather than POS tags. Using this parse tree it is possible to see that

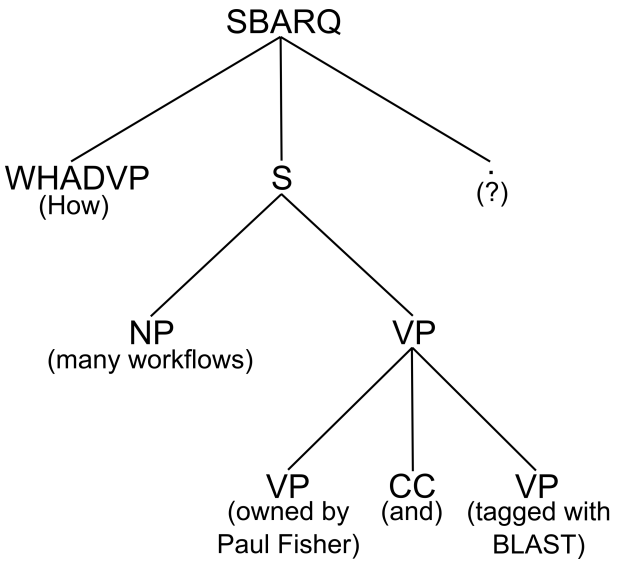


FIGURE 4.9: Parse tree for myExperiment example question

sioc:has_owner should be joined with *myexp:users/43* and *meannot:has-tag* should be joined with *myexp:tags/49*. As the type of entities and the range and domain of the properties can be determined from the ontology, it is possible to determine whether

[41] `http://api.talis.com/stores/wordnet/services/sparql`

these entities are the subject or object of a triple. Therefore the following two SPARQL like triple can be generated:

1. ?s1 sioc:has_owner myexp:users/43
2. ?s2 meannot:has-tag myexp:tags/49

The conjunction 'and' means these two triples can be merged. Depending on the conjunction, the rule for merging terms needs to be different. Table 4.3 shows rules for merging triples with 'and' and 'or' conjunctions. Using the rule for 'and' in table 4.3 for

Conjunction	First Triple	Second Triple	Merged Triples
and	?s1 ?p1 ?o1	?s2 ?p2 ?o2	?s ?p1 ?o1 . ?s ?p2 ?o2
or	?s1 ?p1 ?o1	?s2 ?p2 ?o2	?s ?p1 ?o1 UNION ?s ?p2 ?o2

TABLE 4.3: Example of merging SPARQL-like triples for different conjunctions

the example question provides the following merging:

?s sioc:has_owner myexp:users/43 . ?s meannot:has-tag myexp:tags/49

Next the remaining entity from the question has to be merged. Having mapped workflows to a class in the ontology this generates an RDF *type* triple, i.e. *?s3 rdf:type mecontrib:Workflow*. Because the parse tree in Figure 4.9 shows that this is a noun phrase (NP) for which the previously merged triples are an associated verb phrase (VP), then the two subjects can be taken to be same producing the following merger:

?s rdf:type mecontrib:Workflow .
?s sioc:has_owner myexp:users/43 .
?s meannot:has-tag myexp:tags/49

This merger can then be wrapped with appropriate SPARQL query attributes to build an executable query, as show in listing 4.3. Unfortunately, this query is not currently executable because SWRL rule has not been implemented to generate *meannot:has-tag* relationships from *Tagging*'s *mebase:annotates* and *meannot:uses-tag*. To do this first the rule would need to be written up using the appropriate syntax and secondly a script would need to be written to evaluate all the triples for appropriate matches and then generate the additional triples that this SWRL rule implies.

```
BASE <http://www.myexperiment.org/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX mecontrib: <http://rdf.myexperiment.org/ontologies/contributions/>
PREFIX sioc: <http://rdfs.org/sioc/ns#>
PREFIX meannot: <http://rdf.myexperiment.org/ontologies/annotations/>

SELECT *
WHERE {
  ?s1 rdf:type mecontrib:Workflow .
  ?s1 sioc:has_owner <users/43> .
```

```
  ?s1 meannot:has-tag <tags/49>
}
```

LISTING 4.3: SPARQL Query for myExperiment example question

The results returned by the query could be returned to the user with just some basic formatting, as a listing of URLs. However, this does not fully answer the question, it does not tell the user 'how many' workflows. SPARQL 1.1[42] implements a *COUNT* function and this is now implemented in myExperiment's instance of 4Store. However, it does not yet fully support more complicated mathematical operations, so post-processing functions would need be written to take the results of the SPARQL query and perform the appropriate mathematical operation.

The example question in table 4.2 is a very basic one. More complex questions with negation or temporal clauses or with more complex grammatical structures will be more difficult to answer. Modifying the example question to find workflows that do not have the owner Paul Fisher, requires only a fairly simple modification to the query performing a filter for the FOAF *name* of a *User* to check it is not "Paul Fisher", as shown in listing 4.4.

```
BASE <http://www.myexperiment.org/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX mecontrib: <http://rdf.myexperiment.org/ontologies/contributions/>
PREFIX sioc: <http://rdfs.org/sioc/ns#>
PREFIX meannot: <http://rdf.myexperiment.org/ontologies/annotations/>

SELECT *
WHERE {
  ?s1 rdf:type mecontrib:Workflow .
  ?s1 sioc:has_owner ?user .
  ?user foaf:name ?user_name .
  ?s1 meannot:has-tag <tags/49>
  FILTER (!REGEX('Paul Fisher',STR(?user)))
}
```

LISTING 4.4: Modified SPARQL Query for myExperiment example question

Performing negation on tags is more difficult because there are potentially multiple *has-tag* properties. One way to resolve this is to perform two queries, the original query (listing 4.3) and then the same query without the *has-tag* triple. This would require a post-processing step to subtract the first list of results from the second. However, the issue is not unique to SPARQL as SQL would have the same problem.

There are two main types of temporal clause, explicit and relative. Explicit clauses reference a certain date whereas relative clauses use terms like yesterday, next week, the last three months, a year ago, etc. An information extraction tool like ANNIE could be used to determine entities for both explicit and relative clauses. A separate algorithm

[42] http://www.w3.org/TR/2009/WD-sparql11-query-20091022/

would then need to be implemented to convert relative entities into explicit ones. These explicit entities would then need to be mapped to a SPARQL *FILTER* statement. Taking the phrase "in the last three months" as an example, first this is mapped to a date, if the current date is 18th December 2010, this date will be 18th September 2010. Second this date is mapped to a filter statement $?date >= xsd:dateTime('2010-09-18T00:00:00Z')$. The "greater than or equal to" is determine by the "in the last" part of the phrase.

To map more complex grammatical structures to SPARQL queries might require individual attention. However, as the tasks the user might use the myExperiment QA system for are limited, the number of structures to deal with should also be limited. Also, using an approach used by other QA systems, failing over to executing partial query / queries that could be determined from the question may provide the information the user required. Because myExperiment is a Linked Data project, when URIs are returned as results to the user in a web browser, they can click on these to go to the human-readable web pages for these entities that may provide the answer wanted.

4.3.7 Research Objects for Questions

As described at the end of section 4.3.4, a system built using GATE stores all the inputs and outputs from components in the system. This could be extended to incorporate all the steps to produce the SPARQL query, the SPARQL queries themselves and any post-processing steps used to generate the final results and format them for the user.

Using ROs is one way of capturing the details from the answering of a question in a myExperiment QA system. To be able to build a Question RO, a RO Domain Schema (RODS) will need to be defined. This schema will need to contain classes to represent all the components, inputs and outputs throughout the system. The myExperiment ontology Components module, (see section 3.2.1.4), could be extended to provide a representation for this. A question RO will also capture provenance information such as the user asking the question, the time the question was executed and the state of the gazetteers, thesauruses, RDF datasets at this time. This may require taking a snapshot of the VoID specification at the time of the question.

ROs for questions can be used to provide useful provenance information to the user, thus producing an audit trail allowing assessments of reliability to be made. Capturing the SPARQL query / queries executed as part of the RO allows the user to extract this and use it in their own workflow, making it reusable.

Storing question ROs as part of the QA system, may save having to re-evaluate a question if it has been asked before. The query generated for the question RO will need to be re-executed, along with any time-dependent components such as that for converting relative

temporal clauses into explicit ones, to ensure the up to date results are returned. This shows a repeatable use of a question RO.

Storing question ROs also allows users to review previous questions to see what types of questions will successfully return results, maybe copying and altering them slightly, an example of a re-purposable use case. The system would also be able to capture provenance information referencing the original question for this re-purposed question.

To make a question RO truly reproducible, all the resources that could change over time need to be encapsulated within the RO. This includes full representations of gazetteers, thesauruses and RDF datasets. An example use case for this is needing to prove in a publication that myExperiment had a certain number of workflows on a particular date.

Storing a question as an RO allows the information about its execution to be imported into tools that can evaluate this information and aggregate analysis across multiple questions. Another tool might be one that can demonstrate how the system goes step by step from question to answer. This could help give further confidence to user of the accuracy of the answer, making a question RO re-playable.

As demonstrated, there are clearly use cases for ROs as a means for representing the execution of a questions in a QA system. Having a stable, concrete syntax for question executions gives the QA system the opportunity to provide a number of additional useful features.

Chapter 5

Conclusions

e-Research is intended to support researchers with electronic infrastructure to aid their research. This infrastructure may be high-performance computing systems that allow huge amounts of data to be processed or very complex simulations to be run. It may be tools to allow processes to be automated by a machine saving the time and effort required to do them by hand. It could be applications for electronically logging data as experiments are carried out in the lab. The consequence of all of these is that they output data that may need to be further analysed. This analysis of the data is often a collaborative process, so a key part of this electronic infrastructure is being able to share the data it produces.

Tools such as email, newsgroups, bulletin boards, the Web, IRC and MUDs are all means of communicating that have been used by researchers to share data often in a very free-form way. However, their lack of structure means that often data sharing is not as efficient as it could be, e.g. emailing colleagues with a number of attached data files, where data is replicated and sent to several people. Some of the recipients may have no interest in the data, so for them this replication process is wasted. The sender may have forgotten to send the email to a particular person or somebody new suddenly needs to be able to access the data. This requires additional effort by the sender or one of the recipients to forward it. In an email the sender may provide some metadata about the data files. The distinct separation between the email and its attachments make it very easy for the metadata for these data files to be lost. If the email is forwarded, extra metadata may be added that is only available to the new recipients. Ensuring any email discussion that arises includes all the participants is difficult and aggregating this discussion is a further problem. A lot of these problems arise because email is a distributed means of sharing data. The web is a more centralized way of sharing data but it needs to be tailored so that it can meet the requirements of collaborators who want to share and discuss their research.

Social networking sites are an example of a tailored web application. Their purpose is to allow users to find each other and create links between them in the form of friendships. These friendships allow users to communicate specifically with their friends online. However, unlike Instant Messaging (IM) clients, this communication is designed to be persistent so that both those involved (and maybe other permitted parties) can look back at this in the future, without requiring somebody to capture and distribute a log. Communication is essentially just a specific form of data sharing, where users exchange natural language messages. Social Networking sites allow friends to share other forms of data such as photos, videos, documents, links, etc. and then to discuss these pieces of data. This functionality would also be incredibly useful to the researcher emailing out a set of data files discussed in the earlier example. They would be able to:

1. Make a file available for distributing to a specific group of people but only actually distributing it to those who are interested in the file.
2. Easily allow additional people to access the file without having to resend it.
3. Allow someone to get hold of a new copy, if they lose the original.
4. Provide a central location for a file so that if the associated metadata gets separated from the file it can easily be recovered.
5. Allow additional metadata and discussion about the file to be added and accessible to all parties concerned at any time, now or in the future.

Social Networking websites such as Facebook, MySpace, etc. have been designed for a specific user base who want to share aspects of their social life such as pictures and videos. The requirements of researchers is slightly different from this user base and therefore the social networking model needs to be refined to meet these requirements. The myExperiment project is an example of where this refinement process has been performed by seeking the input of researchers across various fields to ensure that the features provided best support their requirements of data sharing. These requirements included things such as being able to assign structured metadata to files and control over which friends / groups could view, download or edit the metadata for an individual file. This process also identified that additional features beyond that of data sharing were required to support researchers collaborating using social networking applications. These were combined with existing requirements to define the four capabilities of Social VREs (De Roure et al., 2008):

1. Facilitate management of *Research Objects (ROs)*.
2. Support a *social model*.
3. Provide an *open extensible environment*.

4. Provide a platform to *action* research.

Having applications that fulfil these capabilities makes it possible to build an e-Research society that can collaborate and carry out their research online. However, just building these applications alone, may lead to the creation of separate silos of research. Exchanging research between different applications is as important as users being able to share it within an application. REST APIs are one means of sharing information between applications. One thing they lack is a strong semantic representation of the things they exchange. This leads to significant effort being required to transform the data so entities in one application can be interpreted in another. Semantic Web (SW) technologies allow entities to be defined based on the open specification that is an OWL ontology. These ontologies can be used by multiple applications so no transformation is required during exchange. Otherwise, these ontologies describe the entities to assist the transformation process or help other applications interpret these entities so they can make use of them.

One of the main contributions of this thesis has been to take the e-Research society myExperiment and provide a semantic representation and platform for accessing it. To achieve this first the underlying model of myExperiment had to be understood. This required decomposing the model into its three main areas of functionality:

1. Content Management
2. Social Networking
3. Object Annotation

These three areas of functionality were assembled as the *Base* module of an ontology for myExperiment, that also imported SNARM, a bespoke ontology for managing myExperiment's sharing model. Extension modules for specific contributions and annotations, along with functionality for credit and attribution, usage statistics, packs, experiments and workflow components were also defined. To support extensibility in this, *Interface* classes were defined to describe particular behaviours. Using OWL's support for multiple inheritance, non-interface classes were defined as being subclasses of these *Interface* classes so they could inherit their behaviours, e.g. making a *Workflow* *Taggable* and *Rateable*. All these modules were then imported by a final module, named 'Specific', that included entities specific to the main instance of myExperiment. This modularisation was intended to make the ontology reusable. Allowing anyone who reuses the ontology to choose which modules to use and then potentially write their own modules to provide additional functionality where needed.

Reusabilty has been one of the main focusses in the design of the myExperiment ontology. To be able to achieve this it has been essential to reuse existing classes and properties from well established ontologies / schemas. myExperiment has reused classes

and properties from Dublin Core, FOAF, SIOC, OAI-ORE, Creative Commons, SKOS and DBPedia. This makes the task for anyone reusing the myExperiment ontology easier because it clarifies the purpose of classes and properties used. Also it makes the myExperiment ontology easier to align with existing ontologies, as is currently being undertaken with the SWAN ontology.

The myExperiment ontology although having the specific purpose of supporting the myExperiment model, is more generally defined to support the three areas of functionality described earlier. For this it is not the only specification that exists. Drupal a website design framework, original designed to support content management, has defined a RDF schema for its own model. Rather than defining its own classes and properties it has tried to solely reuse existing properties from well established ontologies / schemas. Following this approach, has allowed it to avoid inconsistencies between its models and the those used by the ontologies it reuses such as FOAF and SIOC. By comparison with Drupal's RDF schema it has been possible to identify where inconsistencies lie in the myExperiment ontology, to know where modifications may be required to align with other ontologies. A particular example is the myExperiment's ontology's lack of separation between the user and their account.

Defining an ontology for myExperiment is only one part to building a semantic platform for an e-Research society, another critical part is generating the machine-readable representation of myExperiment entities based on the ontology. At the time this task was undertaken there were no sufficiently advanced tools to support the intricacies of the mappings between myExperiment's relational database, or even its Ruby-on-Rails model, and the myExperiment ontology's model. This led to a bespoke script being written to perform these mappings.

Once the RDF has been generated, it has to be delivered to the user or more particularly the tools they want to use with it. Originally it was decided to keep the machine-readable representations of myExperiment entities separate from the web pages that describe them. This was a good decision for the initial stages of development, as it saved having to make compromises in the design of the ontology, so that it would fit into the existing myExperiment model, allowing the full abstraction of concepts to be achieved. However, as the project progressed it became apparent that for myExperiment to make full use of these machine-readable representations, it would be necessary to integrate them with the myExperiment website. The principles of Linked Data provided guidance in this task. Primarily, the requirement to provide all formats of the same entity from a single URI and then use content negotiation to deliver the correct format.

It was clear that myExperiment differed from many of the other projects that have implemented the principles of Linked Data. myExperiment already had three well developed formats, HTML, XML and RDF, that up to now had been kept quite separate, with their own URI schemes. Careful consideration was needed to bring these different

formats together under a single URI scheme whilst maintaining existing functionality. This led to a few decisions that differ slightly from the model encapsulated by Linked Data principles. One decision was to add an additional redirect to the old XML URI scheme for requests for XML, i.e. URIs ending '.xml'. This differed from choice taken for the RDF API, which redirects requests for the old URI scheme to the new one. Making this decision ensured that applications already using the REST API would not be affected and users would still be able to take the intuitive step of adding '.xml' to the NIR URI to get the XML format. Another deviation from Linked Data principles was to allow requests for ".html" URIs to redirect via the NIR to RDF or XML if that was the format specified in the request. As most myExperiment users are not computer scientists and are probably unaware of how Linked Data works, if they wanted to take a myExperiment entity and use it in a SW-based tool, they may not know that the URI in their web browser is that of the HTML format not the NIR URI. This mechanism saves this being an issue, whether the use of '.html' URI is due to a lack of understanding or by accident. Both these deviations and the choice of a URI scheme that uses file extensions for different formats were made to ensure that the Linked Data provided by myExperiment was as user friendly as possible.

Another interface to the machine-readable representations of myExperiment entities is the SPARQL endpoint. Like the design decisions taken in implementing Linked Data principles, consideration of the user was central. One way this is reflected is in the choice of triplestore. A user wants to be able to receive prompt responses to their queries even as they become more complex, which is one of the features of 4Store. It also provides the user with a useful error and warning message when there is a problem with the query and the option to return results in different formats. To enhance these features, the myExperiment SPARQL endpoint has it own bespoke user interface. This has allowed the user to view information about the status of the SPARQL endpoint, choose additional formats to render there results in, such as an HTML table or a CSV file, quickly add useful prefixes to their query or have them inferred from prefix labels used, clearly see error or warning messages generated by the triplestore and generate their query as a service so they can use it in a separate application such as a Taverna workflow. However, one of the most important features has been the documentation for how to use SPARQL for writing queries to myExperiment, giving extensive explanations of the purpose of various clauses used in SPARQL and demonstrating how they work through example queries that can be executed through myExperiment's SPARQL endpoint. This has helped to bridge the knowledge gap for those with very little or no previous experience of using SPARQL.

To be able to be placed on the Linked Data Cloud, myExperiment has to link to projects already on this cloud. myExperiment has already managed to link to three different projects, EPrints, DBLP and DBPedia, the first two through resources aggregated in myExperiment *Packs* and the last by DBPedia's *residence* property for the location of

myExperiment users. As the resources that packs aggregate are user defined the number of these links can continue to grow, potentially linking to other projects on the Linked Data Cloud. Writing a VoID specification is a way of listing details of the RDF dataset a project publishes along with 'Linksets' to other projects and is a requirement for being placed on the Linked Data Cloud. It became apparent that any VoID specification for myExperiment would need regular updating because of the dynamism of its data. As myExperiment's triplestore is updated every morning, it made sense to also update the published public dataset along with the VoID specification and Linksets to other projects using an automated script. As more projects with data as dynamic as myExperiment are added to the Linked Data Cloud it will become more apparent how important it is to keep these Linksets up to date. Tools are being developed that can discover links between projects and could potentially automatically generate Linksets but until that is fully implemented the onus is on the Linked Data provider to produce up to date VoID specifications and Linksets.

With an ontology, a RDF API built within a Linked Data architecture, a SPARQL endpoint and the publication of an RDF dataset, Linksets and a VoID specification myExperiment can now truly say it has comprehensive semantic platform for an e-Research society. The process of taking this step has provided much insight into the issues of adding such an infrastructure to an existing e-Research society and the tools required to achieve this goal. This insight has helped inform other e-Laboratories projects when it comes to the design of ontologies and implementing Linked Data principles. It has also informed the design of Research Objects (ROs) that have been conceived to allow the transfer of research and research processes between primarily different e-Laboratories projects but ultimately any e-Research projects. myExperiment's representation of packs has guided the ROs in the choice of high level architecture to use, namely OAI-ORE. The myExperiment ontology's use of base and extension modules has informed the choice to define a basic upper model (ROUM) and then extend it with domain specific modules (RODSs). myExperiment's involvement in the ROs effort has allowed it to place itself as a suitable repository for ROs. This has allowed consideration for how it might support the additions required to packs to make them into ROs, in particular how to represent the interrelations between packs items, so that new types of interrelations can be both user-defined but also associated with existing properties and allow provenance about these interrelations, such as its time of creation, to be captured.

Providing the capability to align the myExperiment ontology with existing ontologies, has been one of the focusses of its design. SWAN, an ontology for modelling scientific discourse has been leading an effort as part of the HCLS scientific discourse subtask group at the W3C, to align ontologies in the scientific discourse field. Although the myExperiment ontology has not defined any scientific discourse concepts, it does define a representation for *in silico* experiments, where workflows are enacted and the inputs and outputs are captured along with the provenance of this enactment, such as the time

it was run and the application used to run it. This part of the myExperiment ontology is an element that is not currently defined within SWAN. However, by combining this with SWAN's scientific discourse elements and OBIs specification for *in vivo / in vitro* experiments, it makes it possible to specify the provenance of data and analysis produced by a user to support a claim made in the scientific discourse model, to either support or refute a previous hypothesis. The proposed alignment between myExperiment and a future BioCatalogue ontology would allow further provenance to be stored because BioCatalogue annotates services used in myExperiment workflows and these annotations could be added to the model.

The overall goal of the HCLS scientific discourse subtask group has been to produce a specification for what it sees as the SCientific ARticle of the Future (SCARF), a semantically-rich version of the current conference paper or journal publication. This specification encompasses bibliographic data, rhetorical models, experiment data and social curation. The purpose of this is to support reproducible research where a user can take this SCARF and replicate the research it entails without having to converse with the person who carried it out. ROs share the goal of being able to produce reproducible research. There are benefits for both projects in being able to convert their information from one model to the other. In the case of ROs it would allow the tools within SCARF to be used to help build a semantically rich publication from the research or research process captured in the RO. For SCARF it would allow it to use RO tools for replaying, reusing or re-purposing the research that it captured. Both SCARF and ROs have an overarching model for how they interconnect themselves, in the case of SCARF it is through the use of *Annotation Sets* described using the Annotation Ontology (AO). For ROs it this use of *Interrelation* objects to describe relationships between items.

Many Linked Data projects exist that produce Linked Data but there are far fewer that consume it. It has been shown that myExperiment's semantic representation and Linked Data has been useful in a number of ways, such as allowing collaborators to build workflow recommender systems or for evaluating statistics about users and content on myExperiment. However, it does not "eat its own dog food"[1], by consuming both its own and other projects Linked Data. There are various scenarios where this might be useful. One example of this is in building a QA system for myExperiment. The difficulties of getting to grips with SPARQL has already been discussed and even with extensive effort to support users, there is still a steep learning curve for the user that may put them off. A QA system built using the semantic representation provide by myExperiment ontology, along with data available from other Linked Data projects such as WordNet, is a way of allowing users to take advantage of this representation without the steep learning curve. As section 4.3 described, people have been designing QA systems for over forty years with mixed success. However, up until recently, a lot of the tools required to make this possible have not been available. Annotated treebanks,

[1] `http://en.wikipedia.org/wiki/Eating_your_own_dog_food`

hybrid NL parsers, extensive machine-readable thesauruses and information extraction tools are all necessary in generating a machine-readable representation of a natural language question. However, the critical application is that which allows the user to assemble all these components together, for which GATE is probably the best example. Combining these tools with the detailed semantic representation, that can be provided with an ontology such as myExperiment's, provides the real possibility of producing a QA system that can have both a high rate of success but also be capable of answering complex questions that would be difficult to obtain using any other interface. Linked Data is one of the final pieces of the jigsaw, as for the first time it allows the association of existing resources so that as each evolves, the whole can take advantage of these changes. Such as using new lexical concepts defined by WordNet.

A QA system does not just give the opportunity for myExperiment to consume its own Linked Data, it is also another suitable use case for a RO. All the information captured in the process of answering a question may be useful to whoever asked the question or those that submit questions in the future. These uses coincide with the features of ROs:

- Reliability through user evaluation of the process information.
- Reusability through extraction of SPARQL queries for use elsewhere.
- Repeatability by allowing users to use existing questions rather than writing their own.
- Re-purposing through the adaptation of previous questions with slightly different parameters.
- Reproducibility and re-playability through the re-execution of the answering of a question to prove that a statement made in a publication is true.

The focus in this thesis has been to consider e-Research from a user perspective. Designing tools and applications that allow researchers to take advantage of electronic infrastructure is important but it is key that the users can make full use of these without needing to expend significant effort learning how they work. Beyond this, these tools should not force a user to significantly alter their current research practices or create additional work for them, which does not give them immediate benefit or the ability to see how this additional work will benefit them in the future. This has always been the ethos of the myExperiment project, with continuous user engagement and new features driven by what the user needs at the time rather than a longer development cycle that might lead to the building of extensive functionality not needed by the user. Every step involved in the process of making myExperiment a Linked Data project has been focussed on how to provide this functionality to users in such a way that they can make greatest use of it. Whether they be myExperiment users or those that wish to make use of representation that myExperiment provides through its ontology. To ensure

myExperiment continues to meet the ever more sophisticated requirements of the user community, it needs to be capable of making use of this semantic representation that has been defined for it. This includes being able to support detailed representations of users' research in the form of ROs and sophisticated methods for interrogating this semantic representation, such as through a QA system.

5.1 Research Outcomes

Chapter 1 described the novel research contributions of this thesis. The main contribution cited was the extensive insight provided towards the design on Research Objects that was obtained through the development of a semantic platform for the emergent e-Research society of myExperiment. This process has made myExperiment a suitable repository for storing and developing Research Objects and this thesis has discussed the steps that myExperiment should take to support full Research Objects so that users can share and manage their research and research processes to produce truly reproducible research.

As stated in chapter 1, before insight could even be contributed to the design of Research Objects a number of prerequisite contributions were required. The first of these contributions was to review the myExperiment data model. Doing this it was possible to abstract the most basic features and entities and consider how they combine to provide more sophisticated applications. This made it possible to determine what is required to build a basic platform for a generic e-Research society that supports users in uploading, sharing and annotating research output within a social networking framework.

Using the abstraction of the myExperiment data model, the next contribution was to design an OWL ontology for myExperiment. This has allowed it to provide much richer data to its users. Allowing users to export or query over this data has given them the opportunity to build new applications on top of myExperiment, such as workflow recommender systems. A significant part of this contribution was to give consideration on how the OWL specification could be used to provide an ontology that was both extensible and reusable. Allowing the ontology to continue to evolve and grow, as the myExperiment data model changes and provide the opportunity to align with related ontologies.

Another prerequisite contribution was the augmentation of the existing myExperiment website to support Linked Data. Doing this allowed myExperiment to take advantage of the data captured by other Linked Data projects such as the metadata captured about publications in EPrints and DBLP and the geographical information captured in DBPedia. This process also provided a node for other projects such a BioCatalogue to associate with, so if it implements Linked Data it can link up with myExperiment. One of the most significant outcomes of this contribution was the insight gained of

the difficulties of bringing together several different existing delivery mechanisms for the various formats of myExperiment data under the umbrella of Linked Data. In particular how to achieve this without causing significant disruption to existing users.

As a result of the careful design the myExperiment ontology, making it both reusable and extensible, it was possible to collaborate with similar projects to begin the initial alignment with other e-Research ontologies for use in scientific discourse. This undertaking has made it possible to gain insight into myExperiment's place in the world of e-Research. It has also allowed a comparison between the specification being generated by this alignment and the architecture of Research Objects. In particular how these might complement each other and how they may develop symbiotically so each can potentially make use of the tools and applications developed for the other.

After building a semantic platform for an e-Research society and using it to provide insight into the design of Research Objects, the final contribution of this thesis was to consider how both of these could be used to provide a more sophisticated and hopefully intuitive interface for querying the data of an e-Research society, in particular myExperiment. This took the form of a theoretical exercise of how existing Natural Language Processing (NLP) tools and the Linked Data and semantic platform provided by myExperiment could be combined to produce a Question-Answering (QA) system for interrogating myExperiment data. This exercise has allowed an assessment of how myExperiment can further link with other Linked Data projects to provide an even richer representation of its data. It has also allowed the conceptualisation of a QA system execution as a new and different type of Research Object, where the answering of a question is the particular use case.

The central outcome to all of the contributions in this thesis has been towards the ultimate goal of producing reproducible research in a world where both research and research processes are growing ever more complex and difficult to understand.

5.2 Further Work

As this chapter has made clear, there are several areas in which the work undertaken by this thesis could be extended. There are still aspects of the myExperiment ontology and the RDF API that may need to be evolved to support alignment with other ontologies and to provide users with the highest quality of RDF data. These include the separation of the user and account concepts, the adaptation of *Annotations* so they can be aligned with the Open Annotation ontology and the definition and application of SWRL rules so that reified objects like *Taggings*, *Ratings*, *Creditations* can be inferred to produce simple relationships between a *Contribution* and the value for an associated *Annotation*.

ROs are still very much at a theoretical stage and considerable work is still needed for them to be realised. Support for ROs in myExperiment is a means for moving towards this realisation stage. At present only a very basic assessment has been made to how myExperiment and its ontology can support ROs. If time was spent on an initial implementation based on this, a more detailed assessment will be possible.

Like ROs, the process of ontology alignment to provide a specification for SCARF is still at a theoretical stage. Dedicated time to assess how to build the experimental data component is needed. Once both RO and SCARF reach the realisation stage, it should be easier to determine if and how research and research processes stored in each can be exchanged. However, this is probably not a piece of further work that could be undertaken immediately.

The design and implementation of any Question-Answering (QA) system is a major task. Even with significant consideration on how a QA system for myExperiment can take advantage of modern NL processing tools and be assisted by Semantic Web technologies and the power of Linked Data, there is still a lot of detail to be determined. However, as e-Research becomes more complex, providing novel user interfaces, which can handle this complexity and expose it to users in a way that they can digest, will become more and more important.

Appendix A

myExperiment Data and Specifications

A.1 myExperiment REST API Responses

A.1.1 Example Workflow REST API Response

http://www.myexperiment.org/workflow.xml?id=16 as of 16/10/2011.

```
<?xml version="1.0"?>
<workflow uri="http://www.myexperiment.org/workflow.xml?id=16" resource="http://www.
    myexperiment.org/workflows/16" version="7">
  <id>16</id>
  <title>Pathways and Gene annotations for QTL region</title>
  <description>&lt;p&gt;This workflow searches for genes which reside in a QTL (
    Quantitative Trait Loci) region in the mouse, Mus musculus. The workflow requires
    an input of: a chromosome name or number; a QTL start base pair position; QTL end
    base pair position. Data is then extracted from BioMart to annotate each of the
    genes found in this region. The Entrez and UniProt identifiers are then sent to
    KEGG to obtain KEGG gene identifiers. The KEGG gene identifiers are then used to
    searcg for pathways in the KEGG pathway database.&lt;/p&gt;</description>
  <type uri="http://www.myexperiment.org/type.xml?id=2" resource="http://www.
    myexperiment.org/content_types/2">Taverna 2</type>
  <uploader resource="http://www.myexperiment.org/users/43" uri="http://www.
    myexperiment.org/user.xml?id=43">Paul Fisher</uploader>
  <created-at>Fri Sep 02 11:43:00 +0100 2011</created-at>
  <preview>http://www.myexperiment.org/workflows/16/previews/full</preview>
  <svg>http://www.myexperiment.org/workflows/16/previews/svg</svg>
  <license-type uri="http://www.myexperiment.org/license.xml?id=2" resource="http://
    www.myexperiment.org/licenses/2">by-sa</license-type>
  <content-uri>http://www.myexperiment.org/workflows/16/download/
    pathways_and_gene_annotations_forqtl_region_290738.t2flow</content-uri>
  <content-type>application/vnd.taverna.t2flow+xml</content-type>
  <tags>
    <tag uri="http://www.myexperiment.org/tag.xml?id=462" resource="http://www.
    myexperiment.org/tags/462">shim</tag>
    <tag uri="http://www.myexperiment.org/tag.xml?id=366" resource="http://www.
    myexperiment.org/tags/366">pathway</tag>
```

```
    <tag uri="http://www.myexperiment.org/tag.xml?id=412" resource="http://www.
    myexperiment.org/tags/412">qtl</tag>
    <tag uri="http://www.myexperiment.org/tag.xml?id=275" resource="http://www.
    myexperiment.org/tags/275">kegg</tag>
    <tag uri="http://www.myexperiment.org/tag.xml?id=214" resource="http://www.
    myexperiment.org/tags/214">genotype</tag>
    <tag uri="http://www.myexperiment.org/tag.xml?id=377" resource="http://www.
    myexperiment.org/tags/377">phenotype</tag>
    <tag uri="http://www.myexperiment.org/tag.xml?id=574" resource="http://www.
    myexperiment.org/tags/574">data-driven</tag>
    <tag uri="http://www.myexperiment.org/tag.xml?id=573" resource="http://www.
    myexperiment.org/tags/573">pathway-driven</tag>
    <tag uri="http://www.myexperiment.org/tag.xml?id=368" resource="http://www.
    myexperiment.org/tags/368">pathways</tag>
    <tag uri="http://www.myexperiment.org/tag.xml?id=119" resource="http://www.
    myexperiment.org/tags/119">disease</tag>
    <tag uri="http://www.myexperiment.org/tag.xml?id=694" resource="http://www.
    myexperiment.org/tags/694">nbiconworkflows</tag>
    <tag uri="http://www.myexperiment.org/tag.xml?id=580" resource="http://www.
    myexperiment.org/tags/580">mouse</tag>
    <tag uri="http://www.myexperiment.org/tag.xml?id=1307" resource="http://www.
    myexperiment.org/tags/1307">subworkflow</tag>
    <tag uri="http://www.myexperiment.org/tag.xml?id=69" resource="http://www.
    myexperiment.org/tags/69">chromosome</tag>
    <tag uri="http://www.myexperiment.org/tag.xml?id=144" resource="http://www.
    myexperiment.org/tags/144">ensembl</tag>
    <tag uri="http://www.myexperiment.org/tag.xml?id=145" resource="http://www.
    myexperiment.org/tags/145">entrez</tag>
    <tag uri="http://www.myexperiment.org/tag.xml?id=178" resource="http://www.
    myexperiment.org/tags/178">gene</tag>
    <tag uri="http://www.myexperiment.org/tag.xml?id=1190" resource="http://www.
    myexperiment.org/tags/1190">genes</tag>
    <tag uri="http://www.myexperiment.org/tag.xml?id=516" resource="http://www.
    myexperiment.org/tags/516">uniprot</tag>
  </tags>
</workflow>
```

Listing A.1: Example Workflow REST Response

A.2 myExperiment Ontology Modules

A.2.1 Simple Network Access Rights Management (SNARM) Ontology Module

http://rdf.myexperiment.org/ontologies/snarm/ as of 16/10/2011.

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE rdf:RDF [
 <!ENTITY snarm 'http://rdf.myexperiment.org/ontologies/snarm/'>
 <!ENTITY rdf  'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
 <!ENTITY rdfs 'http://www.w3.org/2000/01/rdf-schema#'>
 <!ENTITY owl 'http://www.w3.org/2002/07/owl#'>
 <!ENTITY xsd 'http://www.w3.org/2001/XMLSchema#'>
 <!ENTITY dc 'http://purl.org/dc/elements/1.1/'>
```

```
 <!ENTITY dcterms 'http://purl.org/dc/terms/'>
]>

<rdf:RDF xml:base       ="&snarm;"
         xmlns          ="&snarm;"
         xmlns:rdf      ="&rdf;"
         xmlns:rdfs     ="&rdfs;"
         xmlns:owl      ="&owl;"
         xmlns:dc       ="&dc;"
         xmlns:dcterms  ="&dcterms;"
         xmlns:xsd      ="&xsd;"
>

  <!-- =============== Description ==================== -->

  <owl:Ontology rdf:about="&snarm;">
    <owl:versionInfo></owl:versionInfo>
    <rdfs:label>SNARM Ontology v1.1</rdfs:label>
    <rdfs:comment> This ontology is designed for representing access rights within a
    simple network of associated users/groups.</rdfs:comment>
    <dc:language>en</dc:language>
    <dc:title xml:lang="en">Simple Network Access Rights Management (SNARM) Ontology</
    dc:title>
    <dc:creator rdf:resource="http://rdf.ecs.soton.ac.uk/person/9421"/>
    <dc:contributor rdf:datatype="http://www.w3.org/2001/XMLSchema#string">David R
    Newman</dc:contributor>
    <dc:publisher rdf:resource="http://rdf.myexperiment.org"/>
    <dc:date rdf:datatype="http://www.w3.org/2001/XMLSchema#date">2009-01-28</dc:date>
    <owl:versionInfo>$Date: 2011/09/02 $</owl:versionInfo>
    <dc:format rdf:datatype="http://www.w3.org/2001/XMLSchema#string">rdf/xml</
    dc:format>
  </owl:Ontology>

  <!-- ============= Annotation Properties =========== -->

  <rdf:Description rdf:about="&dc;language">
    <rdf:type rdf:resource="&owl;AnnotationProperty"/>
  </rdf:Description>

  <rdf:Description rdf:about="&dc;title">
    <rdf:type rdf:resource="&owl;AnnotationProperty"/>
  </rdf:Description>

  <rdf:Description rdf:about="&dc;creator">
    <rdf:type rdf:resource="&owl;AnnotationProperty"/>
  </rdf:Description>

  <rdf:Description rdf:about="&dc;contributor">
    <rdf:type rdf:resource="&owl;AnnotationProperty"/>
  </rdf:Description>

  <rdf:Description rdf:about="&dc;publisher">
    <rdf:type rdf:resource="&owl;AnnotationProperty"/>
  </rdf:Description>

  <rdf:Description rdf:about="&dc;date">
    <rdf:type rdf:resource="&owl;AnnotationProperty"/>
  </rdf:Description>
```

```
<rdf:Description rdf:about="&dc;format">
  <rdf:type rdf:resource="&owl;AnnotationProperty"/>
</rdf:Description>

<!-- ========= OWL-DL Compliance statements ======== -->

<rdf:Description rdf:about="&dcterms;RightsStatement">
  <rdf:type rdf:resource="&owl;Class"/>
</rdf:Description>


<!-- ================== CLASSES =================== -->

<owl:Class rdf:about="Access">
  <rdfs:label>Access</rdfs:label>
  <rdfs:comment>The Unrestricted Access to an AccessType</rdfs:comment>
  <owl:disjointWith rdf:resource="Accesser"/>
  <owl:disjointWith rdf:resource="AccessType"/>
  <owl:disjointWith rdf:resource="Policy"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="has-access-type" />
      <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:isDefinedBy rdf:resource="&snarm;"/>
</owl:Class>

<owl:Class rdf:about="Accesser">
  <rdfs:label>Accesser</rdfs:label>
  <rdfs:comment>The Accesser that is getting access</rdfs:comment>
  <owl:disjointWith rdf:resource="Access"/>
  <owl:disjointWith rdf:resource="AccessType"/>
  <owl:disjointWith rdf:resource="Policy"/>
  <rdfs:isDefinedBy rdf:resource="&snarm;"/>
</owl:Class>

<owl:Class rdf:about="AccessType">
  <rdfs:label>AccessType</rdfs:label>
  <rdfs:comment>The AccessType that is being giving, e.g. view, edit, download, etc.
  </rdfs:comment>
  <owl:disjointWith rdf:resource="Access"/>
  <owl:disjointWith rdf:resource="Accesser"/>
  <owl:disjointWith rdf:resource="Policy"/>
  <rdfs:isDefinedBy rdf:resource="&snarm;"/>
</owl:Class>

<owl:Class rdf:about="Policy">
  <rdfs:label>Policy</rdfs:label>
  <rdfs:comment>A Policy for the access rights to an object for users in the social
  network</rdfs:comment>
  <owl:disjointWith rdf:resource="Access"/>
  <owl:disjointWith rdf:resource="Accesser"/>
  <owl:disjointWith rdf:resource="AccessType"/>
  <rdfs:subClassOf rdf:resource="&dcterms;RightsStatement"/>
  <rdfs:isDefinedBy rdf:resource="&snarm;"/>
</owl:Class>

<owl:Class rdf:about="RestrictedAccess">
```

```
    <rdfs:label>RestrictedAccess</rdfs:label>
    <rdfs:comment>The restricted Access to an AccessType</rdfs:comment>
    <rdfs:subClassOf rdf:resource="Access"/>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="has-accesser" />
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:isDefinedBy rdf:resource="&snarm;"/>
  </owl:Class>


<!-- ============= Object Properties ============== -->
  <owl:ObjectProperty rdf:about="has-access">
    <rdfs:label>has-access</rdfs:label>
    <rdfs:comment>An Access that a Policy provides</rdfs:comment>
    <rdfs:domain rdf:resource="Policy"/>
    <rdfs:range rdf:resource="Access"/>
    <rdfs:isDefinedBy rdf:resource="&snarm;"/>
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:about="has-accesser">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:label>has-accesser</rdfs:label>
    <rdfs:comment>An Accesser that a Mode provides access to</rdfs:comment>
    <rdfs:domain rdf:resource="Access"/>
    <rdfs:range rdf:resource="Accesser"/>
    <rdfs:isDefinedBy rdf:resource="&snarm;"/>
  </owl:ObjectProperty>

   <owl:ObjectProperty rdf:about="has-access-type">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:label>has-access-type</rdfs:label>
    <rdfs:comment>The AccessType an Access provides</rdfs:comment>
    <rdfs:range rdf:resource="AccessType"/>
    <rdfs:isDefinedBy rdf:resource="&snarm;"/>
  </owl:ObjectProperty>

</rdf:RDF>
```

Listing A.2: Simple Network Access Rights Management (SNARM) Ontology Module

A.2.2 Base Ontology Module

`http://rdf.myexperiment.org/ontologies/base/` as of 16/10/2011.

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE rdf:RDF [
 <!ENTITY rdf  'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
 <!ENTITY rdfs 'http://www.w3.org/2000/01/rdf-schema#'>
 <!ENTITY owl 'http://www.w3.org/2002/07/owl#'>
 <!ENTITY xsd 'http://www.w3.org/2001/XMLSchema#'>
 <!ENTITY cc 'http://web.resource.org/cc/'>
 <!ENTITY dc 'http://purl.org/dc/elements/1.1/'>
 <!ENTITY dcterms 'http://purl.org/dc/terms/'>
```

```
 <!ENTITY foaf 'http://xmlns.com/foaf/0.1/'>
 <!ENTITY sioc 'http://rdfs.org/sioc/ns#'>
 <!ENTITY snarm 'http://rdf.myexperiment.org/ontologies/snarm/'>
 <!ENTITY ore 'http://www.openarchives.org/ore/terms/'>
 <!ENTITY dbpedia 'http://dbpedia.org/ontology/'>
 <!ENTITY mebase 'http://rdf.myexperiment.org/ontologies/base/'>
]>

<rdf:RDF xml:base                ="&mebase;"
         xmlns                          ="&mebase;"
         xmlns:rdf        ="&rdf;"
         xmlns:rdfs       ="&rdfs;"
         xmlns:owl        ="&owl;"
         xmlns:cc     ="&cc;"
         xmlns:dc                 ="&dc;"
               xmlns:dcterms="&dcterms;"
         xmlns:foaf       ="&foaf;"
               xmlns:sioc   ="&sioc;"
               xmlns:snarm  ="&snarm;"
         xmlns:ore              ="&ore;"
         xmlns:dbpedia="&dbpedia;"
         xmlns:xsd              ="&xsd;"
>
  <!-- =============== Description ==================== -->

  <owl:Ontology rdf:about="&mebase;">
    <rdfs:label>myExperiment Base v1.0</rdfs:label>
    <rdfs:comment>This provides the base elements required by myExperiment for content
     management, social networking and object annotation.</rdfs:comment>
    <dc:language>en</dc:language>
    <dc:title xml:lang="en">The myExperiment Base Ontology</dc:title>
    <dc:creator rdf:resource="http://rdf.ecs.soton.ac.uk/person/9421"/>
    <dc:contributor rdf:datatype="http://www.w3.org/2001/XMLSchema#string">David R
    Newman</dc:contributor>
    <dc:publisher rdf:resource="http://rdf.myexperiment.org"/>
    <dc:date rdf:datatype="http://www.w3.org/2001/XMLSchema#date">2009-01-28</dc:date>
    <owl:versionInfo>$Date: 2010/11/25 $</owl:versionInfo>
    <dc:format rdf:datatype="http://www.w3.org/2001/XMLSchema#string">rdf/xml</
    dc:format>
    <owl:imports rdf:resource="&snarm;"/>
  </owl:Ontology>

<!-- =============== Annotation Properties ========= -->

  <rdf:Description rdf:about="&dc;language">
    <rdf:type rdf:resource="&owl;AnnotationProperty"/>
  </rdf:Description>

  <rdf:Description rdf:about="&dc;title">
    <rdf:type rdf:resource="&owl;AnnotationProperty"/>
  </rdf:Description>

  <rdf:Description rdf:about="&dc;creator">
    <rdf:type rdf:resource="&owl;AnnotationProperty"/>
  </rdf:Description>

  <rdf:Description rdf:about="&dc;contributor">
    <rdf:type rdf:resource="&owl;AnnotationProperty"/>
  </rdf:Description>
```

```
<rdf:Description rdf:about="&dc;publisher">
  <rdf:type rdf:resource="&owl;AnnotationProperty"/>
</rdf:Description>

<rdf:Description rdf:about="&dc;date">
  <rdf:type rdf:resource="&owl;AnnotationProperty"/>
</rdf:Description>

<rdf:Description rdf:about="&dc;format">
  <rdf:type rdf:resource="&owl;AnnotationProperty"/>
</rdf:Description>


<!-- ========= OWL-DL Compliance statements ======== -->

<rdf:Description rdf:about="&dcterms;created">
  <rdf:type rdf:resource="&owl;DatatypeProperty"/>
  <rdfs:range rdf:resource="&xsd;dateTime"/>
</rdf:Description>

<rdf:Description rdf:about="&dcterms;description">
  <rdf:type rdf:resource="&owl;DatatypeProperty"/>
  <rdfs:range rdf:resource="&xsd;string"/>
</rdf:Description>

<rdf:Description rdf:about="&dcterms;modified">
  <rdf:type rdf:resource="&owl;DatatypeProperty"/>
  <rdfs:range rdf:resource="&xsd;dateTime"/>
</rdf:Description>

<rdf:Description rdf:about="&dcterms;title">
  <rdf:type rdf:resource="&owl;DatatypeProperty"/>
  <rdfs:range rdf:resource="&xsd;string"/>
</rdf:Description>

<rdf:Description rdf:about="&dcterms;type">
  <rdf:type rdf:resource="&owl;DatatypeProperty"/>
  <rdfs:range rdf:resource="&xsd;string"/>
</rdf:Description>

<rdf:Description rdf:about="&dcterms;identifier">
  <rdf:type rdf:resource="&owl;DatatypeProperty"/>
  <rdfs:range rdf:resource="&xsd;string"/>
</rdf:Description>

<rdf:Description rdf:about="&dcterms;Agent">
  <rdf:type rdf:resource="&owl;Class"/>
</rdf:Description>

<rdf:Description rdf:about="&dcterms;hasVersion">
  <rdf:type rdf:resource="&owl;ObjectProperty"/>
  <owl:inverseOf rdf:resource="&dcterms;isVersionOf"/>
</rdf:Description>

<rdf:Description rdf:about="&dcterms;isVersionOf">
  <rdf:type rdf:resource="&owl;ObjectProperty"/>
  <owl:inverseOf rdf:resource="&dcterms;hasVersion"/>
</rdf:Description>
```

```
<rdf:Description rdf:about="&foaf;based_near">
  <rdf:type rdf:resource="&owl;DatatypeProperty"/>
  <rdfs:range rdf:resource="&xsd;string"/>
</rdf:Description>

<rdf:Description rdf:about="&foaf;homepage">
  <rdf:type rdf:resource="&owl;ObjectProperty"/>
</rdf:Description>

<rdf:Description rdf:about="&foaf;knows">
  <rdf:type rdf:resource="&owl;ObjectProperty"/>
</rdf:Description>

<rdf:Description rdf:about="&foaf;mbox">
  <rdf:type rdf:resource="&owl;ObjectProperty"/>
</rdf:Description>

<rdf:Description rdf:about="&foaf;mbox_sha1sum">
  <rdf:type rdf:resource="&owl;DatatypeProperty"/>
  <rdfs:range rdf:resource="&xsd;string"/>
</rdf:Description>

<rdf:Description rdf:about="&foaf;name">
  <rdf:type rdf:resource="&owl;DatatypeProperty"/>
  <rdfs:range rdf:resource="&xsd;string"/>
</rdf:Description>

<rdf:Description rdf:about="&sioc;avatar">
  <rdf:type rdf:resource="&owl;ObjectProperty"/>
</rdf:Description>

<rdf:Description rdf:about="&sioc;Item">
  <rdf:type rdf:resource="&owl;Class"/>
</rdf:Description>

<rdf:Description rdf:about="&sioc;name">
  <rdf:type rdf:resource="&owl;DatatypeProperty"/>
  <rdfs:range rdf:resource="&xsd;string"/>
</rdf:Description>

<rdf:Description rdf:about="&sioc;has_member">
  <rdf:type rdf:resource="&owl;ObjectProperty"/>
  <owl:inverseOf rdf:resource="&sioc;member_of"/>
</rdf:Description>

<rdf:Description rdf:about="&sioc;member_of">
  <rdf:type rdf:resource="&owl;ObjectProperty"/>
  <owl:inverseOf rdf:resource="&sioc;has_member"/>
</rdf:Description>

<rdf:Description rdf:about="&sioc;has_owner">
  <rdf:type rdf:resource="&owl;ObjectProperty"/>
  <owl:inverseOf rdf:resource="&sioc;owner_of"/>
</rdf:Description>

<rdf:Description rdf:about="&sioc;owner_of">
  <rdf:type rdf:resource="&owl;ObjectProperty"/>
  <owl:inverseOf rdf:resource="&sioc;has_owner"/>
```

```
</rdf:Description>

<rdf:Description rdf:about="&sioc;UserAccount">
  <rdf:type rdf:resource="&owl;Class"/>
</rdf:Description>

<rdf:Description rdf:about="&sioc;UserGroup">
  <rdf:type rdf:resource="&owl;Class"/>
</rdf:Description>

<rdf:Description rdf:about="&snarm;Policy">
  <rdf:type rdf:resource="&owl;Class"/>
</rdf:Description>

<rdf:Description rdf:about="&cc;license">
  <rdf:type rdf:resource="&owl;ObjectProperty"/>
</rdf:Description>

<rdf:Description rdf:about="&cc;License">
  <rdf:type rdf:resource="&owl;Class"/>
</rdf:Description>

<rdf:Description rdf:about="&cc;requires">
  <rdf:type rdf:resource="&owl;ObjectProperty"/>
</rdf:Description>

 <rdf:Description rdf:about="&cc;permits">
  <rdf:type rdf:resource="&owl;ObjectProperty"/>
</rdf:Description>

<rdf:Description rdf:about="&cc;prohibits">
  <rdf:type rdf:resource="&owl;ObjectProperty"/>
</rdf:Description>

<rdf:Description rdf:about="&dbpedia;residence">
  <rdf:type rdf:resource="&owl;ObjectProperty"/>
</rdf:Description>



<!-- ====================== Interfaces ====================== -->

<owl:Class rdf:about="Interface">
  <rdfs:label>Interface</rdfs:label>
  <rdfs:comment>Superclass for all Interface classes</rdfs:comment>
  <rdfs:isDefinedBy rdf:resource="&mebase;"/>
</owl:Class>

<owl:Class rdf:about="Annotatable">
  <rdfs:label>Annotatable</rdfs:label>
  <rdfs:comment>An object that can be annotated with Annotations</rdfs:comment>
  <rdfs:subClassOf rdf:resource="Interface"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="has-annotation" />
      <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">0</
  owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
```

```
  <rdfs:isDefinedBy rdf:resource="&mebase;"/>
</owl:Class>

<owl:Class rdf:about="Versionable">
  <rdfs:label>Versionable</rdfs:label>
  <rdfs:comment>A Contribution that can be a Version</rdfs:comment>
  <rdfs:subClassOf rdf:resource="Interface"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="has-current-version" />
      <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="has-version" />
      <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">1</
  owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:isDefinedBy rdf:resource="&mebase;"/>
</owl:Class>

<owl:Class rdf:about="Version">
  <rdfs:label>Version</rdfs:label>
  <rdfs:comment>A Contribution may be a Version of another Contribution</
  rdfs:comment>
  <rdfs:subClassOf rdf:resource="Interface"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="version-number" />
      <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&dcterms;isVersionOf" />
      <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:isDefinedBy rdf:resource="&mebase;"/>
</owl:Class>


<!-- ================== Abstract Classes ================== -->

<owl:Class rdf:about="Actor">
  <rdfs:label>Actor</rdfs:label>
  <rdfs:comment>An object that can perform an action</rdfs:comment>
  <owl:disjointWith rdf:resource="Submission"/>
  <rdfs:subClassOf rdf:resource="&dcterms;Agent"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&dcterms;created" />
      <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
```

```
        <owl:onProperty rdf:resource="&dcterms;modified" />
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="&sioc;name" />
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="&foaf;name" />
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="&dcterms;description" />
        <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1</
    owl:maxCardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:isDefinedBy rdf:resource="&mebase;"/>
  </owl:Class>

  <owl:Class rdf:about="Submission">
    <rdfs:label>Submission</rdfs:label>
    <rdfs:comment>An object that has been submitted.  This might be a Contribution,
    Annotation, Request or an Attibution/Creditation of an Upload</rdfs:comment>
    <owl:disjointWith rdf:resource="Actor"/>
    <owl:equivalentClass rdf:resource="&sioc;Item"/>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="&dcterms;created" />
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:isDefinedBy rdf:resource="&mebase;"/>
  </owl:Class>

  <owl:Class rdf:about="Annotation">
    <rdfs:label>Annotation</rdfs:label>
    <rdfs:comment>An Annotation of a Annotatable object</rdfs:comment>
    <owl:disjointWith rdf:resource="Request"/>
    <owl:disjointWith rdf:resource="Announcement"/>
    <owl:disjointWith rdf:resource="Contribution"/>
    <owl:disjointWith rdf:resource="Message"/>
    <rdfs:subClassOf rdf:resource="Submission" />
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="has-annotator" />
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="annotates" />
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
```

```
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:isDefinedBy rdf:resource="&mebase;"/>
 </owl:Class>

 <owl:Class rdf:about="Contribution">
    <rdfs:label>Contribution</rdfs:label>
    <rdfs:comment>An object that is contributed by a User</rdfs:comment>
    <owl:disjointWith rdf:resource="Annotation"/>
    <owl:disjointWith rdf:resource="Announcement"/>
    <owl:disjointWith rdf:resource="Request"/>
    <owl:disjointWith rdf:resource="Message"/>
    <rdfs:subClassOf rdf:resource="Submission" />
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="&dcterms;modified" />
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="&sioc;has_owner" />
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
     <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="has-policy" />
        <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1</
    owl:maxCardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="&dcterms;description" />
        <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1</
    owl:maxCardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:isDefinedBy rdf:resource="&mebase;"/>
  </owl:Class>

  <owl:Class rdf:about="Request">
    <rdfs:label>Request</rdfs:label>
    <rdfs:comment>A Request can be made by an Actor to another Actor</rdfs:comment>
    <owl:disjointWith rdf:resource="Annotation"/>
    <owl:disjointWith rdf:resource="Announcement"/>
    <owl:disjointWith rdf:resource="Contribution"/>
    <owl:disjointWith rdf:resource="Message"/>
    <rdfs:subClassOf rdf:resource="Submission"/>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="has-requester" />
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="has-accepter" />
```

```
      <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1</
  owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="accepted-at" />
      <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1</
  owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&sioc;has_owner" />
      <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1</
  owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:isDefinedBy rdf:resource="&mebase;"/>
</owl:Class>

<owl:Class rdf:about="Invitation">
  <rdfs:label>Invitation</rdfs:label>
  <rdfs:comment>A Request could be an external Invitation</rdfs:comment>
  <owl:disjointWith rdf:resource="Friendship"/>
  <owl:disjointWith rdf:resource="Membership"/>
  <rdfs:subClassOf rdf:resource="Request"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&sioc;has_owner" />
      <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&foaf;mbox_sha1sum" />
      <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:isDefinedBy rdf:resource="&mebase;"/>
</owl:Class>

<owl:Class rdf:about="Upload">
  <rdfs:label>Upload</rdfs:label>
  <rdfs:comment>An object that can be contributed by a User that requires uploading<
  /rdfs:comment>
  <rdfs:subClassOf rdf:resource="Contribution"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="content-url" />
      <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1</
  owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="has-content-type" />
      <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1</
  owl:maxCardinality>
```

```
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&cc;license" />
      <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&dcterms;title" />
      <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:isDefinedBy rdf:resource="&mebase;"/>
</owl:Class>

<!-- =============== myExperiment Entity Classes ================= -->

<owl:Class rdf:about="Announcement">
  <rdfs:label>Announcement</rdfs:label>
  <rdfs:comment>A public Announcement</rdfs:comment>
  <owl:disjointWith rdf:resource="Annotation"/>
  <owl:disjointWith rdf:resource="Contribution"/>
  <owl:disjointWith rdf:resource="Message"/>
  <owl:disjointWith rdf:resource="Request"/>
  <rdfs:subClassOf rdf:resource="Submission"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="has-announcer" />
      <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&dcterms;title" />
      <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="text" />
      <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&dcterms;modified" />
      <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:isDefinedBy rdf:resource="&mebase;"/>
</owl:Class>

<owl:Class rdf:about="ContentType">
  <rdfs:label>ContentType</rdfs:label>
  <rdfs:comment>The type of content for an Upload</rdfs:comment>
  <rdfs:subClassOf rdf:resource="&mebase;Contribution"/>
  <rdfs:isDefinedBy rdf:resource="&mebase;"/>
```

```
</owl:Class>

<owl:Class rdf:about="Friendship">
  <rdfs:label>Friendship</rdfs:label>
  <rdfs:comment>A Friendship between two Users</rdfs:comment>
  <owl:disjointWith rdf:resource="Membership"/>
  <owl:disjointWith rdf:resource="Invitation"/>
  <rdfs:subClassOf rdf:resource="Request"/>
  <rdfs:isDefinedBy rdf:resource="&mebase;"/>
</owl:Class>

<owl:Class rdf:about="FriendshipInvitation">
  <rdfs:label>FriendshipInvitation</rdfs:label>
  <rdfs:comment>A FriendshipInvitation to an external email address</rdfs:comment>
  <owl:disjointWith rdf:resource="MembershipInvitation"/>
  <rdfs:subClassOf rdf:resource="Invitation"/>
  <rdfs:isDefinedBy rdf:resource="&mebase;"/>
</owl:Class>

<owl:Class rdf:about="Group">
  <rdfs:label>Group</rdfs:label>
  <rdfs:comment>A Group of Users</rdfs:comment>
  <owl:equivalentClass rdf:resource="&sioc;UserGroup"/>
  <owl:disjointWith rdf:resource="User"/>
  <rdfs:subClassOf rdf:resource="Actor" />
   <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&sioc;has_owner" />
      <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&dcterms;description" />
      <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1</
  owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&sioc;has_member" />
      <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">0</
  owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:isDefinedBy rdf:resource="&mebase;"/>
</owl:Class>

<owl:Class rdf:about="GroupAnnouncement">
  <rdfs:label>GroupAnnouncement</rdfs:label>
  <rdfs:comment>An Announcement to a Group</rdfs:comment>
  <rdfs:subClassOf rdf:resource="Announcement" />
  <rdfs:subClassOf>
    <owl:Restriction>
       <owl:onProperty rdf:resource="public-announcement" />
       <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
```

```
    <owl:Restriction>
      <owl:onProperty rdf:resource="announced-to" />
      <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:isDefinedBy rdf:resource="&mebase;"/>
</owl:Class>

<owl:Class rdf:about="License">
  <rdfs:label>License</rdfs:label>
  <rdfs:comment>A License under which an Upload is licensed under</rdfs:comment>
  <rdfs:subClassOf rdf:resource="Contribution" />
  <rdfs:subClassOf rdf:resource="&cc;License" />
  <rdfs:subClassOf>
    <owl:Restriction>
       <owl:onProperty rdf:resource="&dcterms;identifier" />
       <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&cc;requires" />
      <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">0</
  owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
     <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&cc;permits" />
      <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">0</
  owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&cc;prohibits" />
      <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">0</
  owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:isDefinedBy rdf:resource="&mebase;"/>
</owl:Class>

<owl:Class rdf:about="Membership">
  <rdfs:label>Membership</rdfs:label>
  <rdfs:comment>A Membership of a User to a Group</rdfs:comment>
  <owl:disjointWith rdf:resource="Friendship"/>
  <owl:disjointWith rdf:resource="Invitation"/>
  <rdfs:subClassOf rdf:resource="Request"/>
  <rdfs:isDefinedBy rdf:resource="&mebase;"/>
</owl:Class>

<owl:Class rdf:about="MembershipInvitation">
  <rdfs:label>MembershipInvitation</rdfs:label>
  <rdfs:comment>A MembershipInvitation to an external email address</rdfs:comment>
  <owl:disjointWith rdf:resource="FriendshipInvitation"/>
  <rdfs:subClassOf rdf:resource="Invitation"/>
  <rdfs:isDefinedBy rdf:resource="&mebase;"/>
</owl:Class>
```

```
<owl:Class rdf:about="Message">
  <rdfs:label>Message</rdfs:label>
  <rdfs:comment>A Message sent between two Users</rdfs:comment>
  <owl:disjointWith rdf:resource="Annotation"/>
  <owl:disjointWith rdf:resource="Announcement"/>
  <owl:disjointWith rdf:resource="Contribution"/>
  <owl:disjointWith rdf:resource="Request"/>
  <rdfs:subClassOf rdf:resource="Submission"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="text" />
      <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:isDefinedBy rdf:resource="&mebase;"/>
</owl:Class>

<owl:Class rdf:about="User">
  <rdfs:label>User</rdfs:label>
  <rdfs:comment>A User</rdfs:comment>
  <owl:equivalentClass rdf:resource="&sioc;User"/>
  <owl:disjointWith rdf:resource="Group"/>
  <rdfs:subClassOf rdf:resource="Actor" />
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="username" />
      <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&sioc;avatar" />
      <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1</
  owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&foaf;based_near" />
      <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1</
  owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&foaf;mbox" />
      <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1</
  owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&foaf;mbox_sha1sum" />
      <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1</
  owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
```

```
    <owl:Restriction>
      <owl:onProperty rdf:resource="&foaf;homepage" />
      <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1</
  owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&dbpedia;residence" />
      <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">0</
  owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&sioc;member_of" />
      <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">0</
  owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:isDefinedBy rdf:resource="&mebase;"/>
</owl:Class>

<!-- =============== Object Properties ============== -->

<owl:ObjectProperty rdf:about="has-annotator">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:label>has-annotator</rdfs:label>
  <rdfs:comment>An Annotation has User as annotator</rdfs:comment>
  <rdfs:subPropertyOf rdf:resource="&sioc;has_owner"/>
  <rdfs:domain rdf:resource="Annotation"/>
  <rdfs:range rdf:resource="User"/>
  <owl:inverseOf rdf:resource="annotator-of"/>
  <rdfs:isDefinedBy rdf:resource="&mebase;"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="annotator-of">
  <rdfs:label>annotator</rdfs:label>
  <rdfs:comment>A User is an annotator-of an Annotation</rdfs:comment>
  <rdfs:subPropertyOf rdf:resource="&sioc;owner_of"/>
  <rdfs:domain rdf:resource="User"/>
  <rdfs:range rdf:resource="Annotation"/>
  <owl:inverseOf rdf:resource="has-annotator"/>
  <rdfs:isDefinedBy rdf:resource="&mebase;"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="annotates">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:label>annotates</rdfs:label>
  <rdfs:comment>An Annotation is associated with a particular Contribution</
  rdfs:comment>
  <rdfs:domain rdf:resource="Annotation"/>
  <rdfs:range rdf:resource="Annotatable"/>
  <owl:inverseOf rdf:resource="has-annotation"/>
  <rdfs:isDefinedBy rdf:resource="&mebase;"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="has-annotation">
  <rdf:type rdf:resource="&owl;InverseFunctionalProperty"/>
```

```
  <rdfs:label>has-annotation</rdfs:label>
  <rdfs:comment>An Annotable may have a Annotation</rdfs:comment>
  <rdfs:domain rdf:resource="Annotatable"/>
  <rdfs:range rdf:resource="Annotation"/>
  <owl:inverseOf rdf:resource="annotates"/>
  <rdfs:isDefinedBy rdf:resource="&mebase;"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="has-announcer">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:label>has-announcer</rdfs:label>
  <rdfs:comment>An Announcement has an annoucer that is a User</rdfs:comment>
  <rdfs:subPropertyOf rdf:resource="&sioc;has_owner"/>
  <rdfs:domain rdf:resource="Announcement"/>
  <rdfs:range rdf:resource="User"/>
  <rdfs:isDefinedBy rdf:resource="&mebase;"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="announced-to">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:label>announced-to</rdfs:label>
  <rdfs:comment>The Group a GroupAnnouncement has been announced to</rdfs:comment>
  <rdfs:domain rdf:resource="GroupAnnouncement"/>
  <rdfs:range rdf:resource="Group"/>
  <rdfs:isDefinedBy rdf:resource="&mebase;"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="has-version">
  <rdfs:subPropertyOf rdf:resource="&dcterms;hasVersion"/>
  <rdfs:label>has-version</rdfs:label>
  <rdfs:comment>A Versionable object has at least one version</rdfs:comment>
  <rdfs:domain rdf:resource="Versionable"/>
  <rdfs:range rdf:resource="Version"/>
  <rdfs:isDefinedBy rdf:resource="&mebase;"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="content-url">
  <rdfs:label>content-url</rdfs:label>
  <rdfs:comment>An Upload has content at a URL</rdfs:comment>
  <rdfs:isDefinedBy rdf:resource="&mebase;"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="has-current-version">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:subPropertyOf rdf:resource="has-version"/>
  <rdfs:label>has-current-version</rdfs:label>
  <rdfs:comment>A Versionable object has a current version</rdfs:comment>
  <rdfs:isDefinedBy rdf:resource="&mebase;"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="has-announcement">
  <rdfs:label>has-announcement</rdfs:label>
  <rdfs:comment>Groups may have GroupAnnouncements</rdfs:comment>
  <rdfs:domain rdf:resource="Group"/>
  <rdfs:range rdf:resource="GroupAnnouncement"/>
  <owl:inverseOf rdf:resource="announced-to"/>
  <rdfs:isDefinedBy rdf:resource="&mebase;"/>
</owl:ObjectProperty>
```

```
<owl:ObjectProperty rdf:about="email">
  <rdfs:label>email</rdfs:label>
  <rdfs:comment>A User has an email address</rdfs:comment>
  <rdfs:subPropertyOf rdf:resource="&foaf;mbox"/>
  <rdfs:domain rdf:resource="User"/>
  <rdfs:isDefinedBy rdf:resource="&mebase;"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="is-friends-with">
  <rdfs:subPropertyOf rdf:resource="&foaf;knows" />
  <rdf:type rdf:resource="&owl;SymmetricProperty"/>
  <rdfs:label>is-friends-with</rdfs:label>
  <rdfs:comment>A User may be friends with another User</rdfs:comment>
  <rdfs:domain rdf:resource="User"/>
  <rdfs:range rdf:resource="User"/>
  <rdfs:isDefinedBy rdf:resource="&mebase;"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="from">
  <rdfs:label>from</rdfs:label>
  <rdfs:comment>A Message is sent from a User</rdfs:comment>
  <rdfs:domain rdf:resource="Message"/>
  <rdfs:range rdf:resource="User"/>
  <rdfs:isDefinedBy rdf:resource="&mebase;"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="has-content-type">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:label>has-content-type</rdfs:label>
  <rdfs:comment>An Upload has a ContentType</rdfs:comment>
  <rdfs:domain rdf:resource="Upload"/>
  <rdfs:range rdf:resource="ContentType"/>
  <rdfs:isDefinedBy rdf:resource="&mebase;"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="openid-url">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:label>openid-url</rdfs:label>
  <rdfs:comment>A User may have an openid-url represented as a uri</rdfs:comment>
  <rdfs:domain rdf:resource="User"/>
  <rdfs:isDefinedBy rdf:resource="&mebase;"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="has-policy">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:label>has-policy</rdfs:label>
  <rdfs:comment>A Contribution has a Policy for access rights management</
  rdfs:comment>
  <rdfs:domain rdf:resource="Contribution"/>
  <rdfs:range rdf:resource="&snarm;Policy"/>
  <rdfs:isDefinedBy rdf:resource="&mebase;"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="has-accepter">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:label>has-accepter</rdfs:label>
  <rdfs:comment>A Request must have an accepter that can accept</rdfs:comment>
  <rdfs:domain rdf:resource="Request"/>
  <rdfs:range rdf:resource="Actor"/>
```

```
  <rdfs:isDefinedBy rdf:resource="&mebase;"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="reply-to">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:label>reply-to</rdfs:label>
  <rdfs:comment>A Message may have a Message it is a reply-to</rdfs:comment>
  <rdfs:domain rdf:resource="Message"/>
  <rdfs:range rdf:resource="Message"/>
  <rdfs:isDefinedBy rdf:resource="&mebase;"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="has-requester">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:label>has-requester</rdfs:label>
  <rdfs:comment>A Request must have a requester</rdfs:comment>
  <rdfs:domain rdf:resource="Request"/>
  <rdfs:range rdf:resource="Actor"/>
  <rdfs:isDefinedBy rdf:resource="&mebase;"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="has-friendship">
  <rdf:type rdf:resource="&owl;InverseFunctionalProperty"/>
  <rdfs:label>has-friendship</rdfs:label>
  <rdfs:comment>A User has a Friendship (with another User)</rdfs:comment>
  <rdfs:domain rdf:resource="User"/>
  <rdfs:range rdf:resource="Friendship"/>
  <rdfs:isDefinedBy rdf:resource="&mebase;"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="has-membership">
  <rdf:type rdf:resource="&owl;InverseFunctionalProperty"/>
  <rdfs:label>has-membership</rdfs:label>
  <rdfs:comment>A User has a Membership (of a Group)</rdfs:comment>
  <rdfs:domain rdf:resource="User"/>
  <rdfs:range rdf:resource="Membership"/>
  <rdfs:isDefinedBy rdf:resource="&mebase;"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="has-shared-item">
  <rdfs:label>has-shared-item</rdfs:label>
  <rdfs:comment>Contributions that are shared within a Group</rdfs:comment>
  <rdfs:domain rdf:resource="Group"/>
  <rdfs:range rdf:resource="Contribution"/>
  <rdfs:isDefinedBy rdf:resource="&mebase;"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="to">
  <rdfs:label>to</rdfs:label>
  <rdfs:comment>A Message is sent to a User</rdfs:comment>
  <rdfs:domain rdf:resource="Message"/>
  <rdfs:range rdf:resource="User"/>
  <rdfs:isDefinedBy rdf:resource="&mebase;"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="unconfirmed-email">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:label>unconfirmed-email</rdfs:label>
  <rdfs:comment>An email that has yet to be confirmed</rdfs:comment>
```

```
    <rdfs:domain rdf:resource="User"/>
    <rdfs:isDefinedBy rdf:resource="&mebase;"/>
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:about="uri">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:label>uri</rdfs:label>
    <rdfs:comment>The URI for some object</rdfs:comment>
    <rdfs:isDefinedBy rdf:resource="&mebase;"/>
  </owl:ObjectProperty>

<!-- =============== Datatype Properties ============== -->

  <owl:DatatypeProperty rdf:about="accepted-at">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:label>accepted-at</rdfs:label>
    <rdfs:comment>A Request can be accepted-at a certain dateTime</rdfs:comment>
    <rdfs:domain rdf:resource="Request"/>
    <rdfs:range rdf:resource="&xsd;dateTime"/>
    <rdfs:isDefinedBy rdf:resource="&mebase;"/>
  </owl:DatatypeProperty>

  <owl:DatatypeProperty rdf:about="activated-at">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:label>activated-at</rdfs:label>
    <rdfs:comment>A User account is a activated-at certain dateTime</rdfs:comment>
    <rdfs:domain rdf:resource="User"/>
    <rdfs:range rdf:resource="&xsd;dateTime"/>
    <rdfs:isDefinedBy rdf:resource="&mebase;"/>
  </owl:DatatypeProperty>

  <owl:DatatypeProperty rdf:about="auto-accept">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:label>auto-except</rdfs:label>
    <rdfs:comment>A Group either auto-accepts a User or it doesn't</rdfs:comment>
    <rdfs:domain rdf:resource="Group"/>
    <rdfs:range rdf:resource="&xsd;boolean"/>
    <rdfs:isDefinedBy rdf:resource="&mebase;"/>
  </owl:DatatypeProperty>

  <owl:DatatypeProperty rdf:about="country">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:label>country</rdfs:label>
    <rdfs:comment>A User is based in a country</rdfs:comment>
    <rdfs:domain rdf:resource="User"/>
    <rdfs:range rdf:resource="&xsd;string"/>
    <rdfs:isDefinedBy rdf:resource="&mebase;"/>
  </owl:DatatypeProperty>

  <owl:DatatypeProperty rdf:about="count">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:label>count</rdfs:label>
    <rdfs:comment>Certain Annotations may be a count of something</rdfs:comment>
    <rdfs:domain rdf:resource="Annotation"/>
    <rdfs:range rdf:resource="&xsd;nonNegativeInteger"/>
    <rdfs:isDefinedBy rdf:resource="&mebase;"/>
  </owl:DatatypeProperty>

  <owl:DatatypeProperty rdf:about="contact-details">
```

```
  <rdfs:label>contact-details</rdfs:label>
  <rdfs:comment>A User has contact-details</rdfs:comment>
  <rdfs:domain rdf:resource="User"/>
  <rdfs:range rdf:resource="&xsd;string"/>
  <rdfs:isDefinedBy rdf:resource="&mebase;"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:about="deleted-by-sender">
  <rdfs:label>deleted-by-sender</rdfs:label>
  <rdfs:comment>The sender has deleted this Message from their inbox</rdfs:comment>
  <rdfs:domain rdf:resource="Message"/>
  <rdfs:range rdf:resource="&xsd;boolean"/>
  <rdfs:isDefinedBy rdf:resource="&mebase;"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:about="deleted-by-recipient">
  <rdfs:label>deleted-by-recipient</rdfs:label>
  <rdfs:comment>The recipient has deleted this Message from their inbox</
  rdfs:comment>
  <rdfs:domain rdf:resource="Message"/>
  <rdfs:range rdf:resource="&xsd;boolean"/>
  <rdfs:isDefinedBy rdf:resource="&mebase;"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:about="email-confirmed-at">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:label>email-confirmed-at</rdfs:label>
  <rdfs:comment>An email is confirmed at a certain dateTime</rdfs:comment>
  <rdfs:domain rdf:resource="User"/>
  <rdfs:range rdf:resource="&xsd;dateTime"/>
  <rdfs:isDefinedBy rdf:resource="&mebase;"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:about="field">
  <rdfs:label>field</rdfs:label>
  <rdfs:comment>A User works in a field</rdfs:comment>
  <rdfs:domain rdf:resource="User"/>
  <rdfs:range rdf:resource="&xsd;string"/>
  <rdfs:isDefinedBy rdf:resource="&mebase;"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:about="filename">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:label>filename</rdfs:label>
  <rdfs:comment>A File has a filename</rdfs:comment>
  <rdfs:domain rdf:resource="Upload"/>
  <rdfs:range rdf:resource="&xsd;string"/>
  <rdfs:isDefinedBy rdf:resource="&mebase;"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:about="interests">
  <rdfs:label>interests</rdfs:label>
  <rdfs:comment>A User's interests</rdfs:comment>
  <rdfs:domain rdf:resource="User"/>
  <rdfs:range rdf:resource="&xsd;string"/>
  <rdfs:isDefinedBy rdf:resource="&mebase;"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:about="is-current-version">
```

```
  <rdfs:label>is-current-version</rdfs:label>
  <rdfs:comment>A User's interests</rdfs:comment>
  <rdfs:domain rdf:resource="Versionable"/>
  <rdfs:range rdf:resource="&xsd;boolean"/>
  <rdfs:isDefinedBy rdf:resource="&mebase;"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:about="last-seen-at">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:label>last-seen-at</rdfs:label>
  <rdfs:comment>The last time a User was seen</rdfs:comment>
  <rdfs:domain rdf:resource="User"/>
  <rdfs:range rdf:resource="&xsd;dateTime"/>
  <rdfs:isDefinedBy rdf:resource="&mebase;"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:about="request-token">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:label>request-token</rdfs:label>
  <rdfs:comment>Requests may have a request-token as a string</rdfs:comment>
  <rdfs:domain rdf:resource="Request"/>
  <rdfs:range rdf:resource="&xsd;string"/>
  <rdfs:isDefinedBy rdf:resource="&mebase;"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:about="occupation">
  <rdfs:label>occupation</rdfs:label>
  <rdfs:comment>A User has an occupation</rdfs:comment>
  <rdfs:domain rdf:resource="User"/>
  <rdfs:range rdf:resource="&xsd;string"/>
  <rdfs:isDefinedBy rdf:resource="&mebase;"/>
</owl:DatatypeProperty>

<owl:ObjectProperty rdf:about="openid-url">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:label>openid-url</rdfs:label>
  <rdfs:comment>A User may have an openid-url represented as a uri</rdfs:comment>
  <rdfs:domain rdf:resource="User"/>
  <rdfs:isDefinedBy rdf:resource="&mebase;"/>
</owl:ObjectProperty>

<owl:DatatypeProperty rdf:about="organisation">
  <rdfs:label>organisation</rdfs:label>
  <rdfs:comment>A User is part of an organisation</rdfs:comment>
  <rdfs:domain rdf:resource="User"/>
  <rdfs:range rdf:resource="&xsd;string"/>
  <rdfs:isDefinedBy rdf:resource="&mebase;"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:about="public-announcement">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:label>public-announcement</rdfs:label>
  <rdfs:comment>Is the GroupAnnouncement viewable (public) to those outside the
  Group</rdfs:comment>
  <rdfs:domain rdf:resource="GroupAnnouncement"/>
  <rdfs:range rdf:resource="&xsd;boolean"/>
  <rdfs:isDefinedBy rdf:resource="&mebase;"/>
</owl:DatatypeProperty>
```

```
  <owl:DatatypeProperty rdf:about="read-at">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:label>read-at</rdfs:label>
    <rdfs:comment>A Message is a read-at a particular dateTime</rdfs:comment>
    <rdfs:domain rdf:resource="Message"/>
    <rdfs:range rdf:resource="&xsd;dateTime"/>
    <rdfs:isDefinedBy rdf:resource="&mebase;"/>
  </owl:DatatypeProperty>

  <owl:DatatypeProperty rdf:about="receive-notifications">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:label>receive-notifications</rdfs:label>
    <rdfs:comment>Does the user receive-notifications</rdfs:comment>
    <rdfs:domain rdf:resource="User"/>
    <rdfs:range rdf:resource="&xsd;boolean"/>
    <rdfs:isDefinedBy rdf:resource="&mebase;"/>
  </owl:DatatypeProperty>

  <owl:DatatypeProperty rdf:about="subject">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:label>subject</rdfs:label>
    <rdfs:comment>A Message has a subject</rdfs:comment>
    <rdfs:domain rdf:resource="Message"/>
    <rdfs:range rdf:resource="&xsd;string"/>
    <rdfs:isDefinedBy rdf:resource="&mebase;"/>
  </owl:DatatypeProperty>

  <owl:DatatypeProperty rdf:about="text">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:label>text</rdfs:label>
    <rdfs:comment>A Submission has some text associated with it</rdfs:comment>
    <rdfs:domain rdf:resource="Submission"/>
    <rdfs:range rdf:resource="&xsd;string"/>
    <rdfs:isDefinedBy rdf:resource="&mebase;"/>
  </owl:DatatypeProperty>

  <owl:DatatypeProperty rdf:about="username">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:label>username</rdfs:label>
    <rdfs:comment>A User may have a username represented as a string</rdfs:comment>
    <rdfs:range rdf:resource="&xsd;string"/>
    <rdfs:isDefinedBy rdf:resource="&mebase;"/>
  </owl:DatatypeProperty>

  <owl:DatatypeProperty rdf:about="version-number">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:label>version-number</rdfs:label>
    <rdfs:comment>A Version has a version-number</rdfs:comment>
    <rdfs:domain rdf:resource="Version"/>
    <rdfs:range rdf:resource="&xsd;positiveInteger"/>
    <rdfs:isDefinedBy rdf:resource="&mebase;"/>
  </owl:DatatypeProperty>

</rdf:RDF>
```

Listing A.3: myExperiment's Base Ontology Module

A.2.3 Attribution and Credit Ontology Module

http://rdf.myexperiment.org/ontologies/attrib_credit/ as of 16/10/2011.

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE rdf:RDF [
 <!ENTITY rdf  'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
 <!ENTITY rdfs 'http://www.w3.org/2000/01/rdf-schema#'>
 <!ENTITY owl 'http://www.w3.org/2002/07/owl#'>
 <!ENTITY xsd 'http://www.w3.org/2001/XMLSchema#'>
 <!ENTITY dc 'http://purl.org/dc/elements/1.1/'>
 <!ENTITY dcterms 'http://purl.org/dc/terms/'>
 <!ENTITY snarm 'http://rdf.myexperiment.org/snarm#'>
 <!ENTITY mebase 'http://rdf.myexperiment.org/ontologies/base/'>
 <!ENTITY meac 'http://rdf.myexperiment.org/ontologies/attrib_credit/'>
]>

<rdf:RDF xml:base     ="&meac;"
         xmlns                    ="&meac;"
               xmlns:mebase ="&mebase;"
         xmlns:rdf              ="&rdf;"
         xmlns:rdfs             ="&rdfs;"
         xmlns:owl              ="&owl;"
         xmlns:dc                 ="&dc;"
               xmlns:dcterms="&dcterms;"
               xmlns:snarm  ="&snarm;"
         xmlns:xsd              ="&xsd;"
>
  <!-- =============== Description ==================== -->

  <owl:Ontology rdf:about="&meac;">
    <rdfs:label>myExperiment Attribution &amp; Credit v1.0</rdfs:label>
    <rdfs:comment>This allows contributions to give attribution to earlier
    contributions and pay credit to users and groups involved in their creation.</
    rdfs:comment>
    <dc:language>en</dc:language>
    <dc:title xml:lang="en">The myExperiment Attribution &amp; Creditation ontology</
    dc:title>
    <dc:creator rdf:resource="http://rdf.ecs.soton.ac.uk/person/9421"/>
    <dc:contributor rdf:datatype="http://www.w3.org/2001/XMLSchema#string">David R
    Newman</dc:contributor>
    <dc:publisher rdf:resource="http://rdf.myexperiment.org"/>
    <dc:date rdf:datatype="http://www.w3.org/2001/XMLSchema#date">2009-01-28</dc:date>
    <owl:versionInfo>$Date: 2011/05/19 $</owl:versionInfo>
    <dc:format rdf:datatype="http://www.w3.org/2001/XMLSchema#string">rdf/xml</
    dc:format>
  </owl:Ontology>

<!-- ============= Annotation Properties =========== -->

  <rdf:Description rdf:about="&dc;language">
    <rdf:type rdf:resource="&owl;AnnotationProperty"/>
  </rdf:Description>

  <rdf:Description rdf:about="&dc;title">
    <rdf:type rdf:resource="&owl;AnnotationProperty"/>
  </rdf:Description>
```

```
<rdf:Description rdf:about="&dc;creator">
  <rdf:type rdf:resource="&owl;AnnotationProperty"/>
</rdf:Description>

<rdf:Description rdf:about="&dc;contributor">
  <rdf:type rdf:resource="&owl;AnnotationProperty"/>
</rdf:Description>

<rdf:Description rdf:about="&dc;publisher">
  <rdf:type rdf:resource="&owl;AnnotationProperty"/>
</rdf:Description>

<rdf:Description rdf:about="&dc;date">
  <rdf:type rdf:resource="&owl;AnnotationProperty"/>
</rdf:Description>

<rdf:Description rdf:about="&dc;format">
  <rdf:type rdf:resource="&owl;AnnotationProperty"/>
</rdf:Description>


<!-- ========= OWL-DL Compliance statements ======== -->

<rdf:Description rdf:about="&dcterms;modified">
  <rdf:type rdf:resource="&owl;DatatypeProperty"/>
  <rdfs:range rdf:resource="&xsd;dateTime"/>
</rdf:Description>

<rdf:Description rdf:about="&mebase;Contribution">
  <rdf:type rdf:resource="&owl;Class"/>
</rdf:Description>

<rdf:Description rdf:about="&mebase;Interface">
  <rdf:type rdf:resource="&owl;Class"/>
</rdf:Description>

<rdf:Description rdf:about="&mebase;Actor">
  <rdf:type rdf:resource="&owl;Class"/>
</rdf:Description>

<rdf:Description rdf:about="&mebase;Announcement">
  <rdf:type rdf:resource="&owl;Class"/>
</rdf:Description>

<rdf:Description rdf:about="&mebase;Submission">
  <rdf:type rdf:resource="&owl;Class"/>
</rdf:Description>

<rdf:Description rdf:about="&mebase;Message">
  <rdf:type rdf:resource="&owl;Class"/>
</rdf:Description>

<rdf:Description rdf:about="&mebase;Request">
  <rdf:type rdf:resource="&owl;Class"/>
</rdf:Description>


<!-- ====================== Interfaces ====================== -->
```

```
<owl:Class rdf:about="Attributable">
  <rdfs:label>Attributable</rdfs:label>
  <rdfs:comment>An object can be attributed to another object</rdfs:comment>
  <rdfs:subClassOf rdf:resource="&mebase;Interface"/>
  <rdfs:isDefinedBy rdf:resource="&meac;"/>
</owl:Class>

<owl:Class rdf:about="Creditable">
  <rdfs:label>Creditable</rdfs:label>
  <rdfs:comment>An object can be credited to someone</rdfs:comment>
  <rdfs:subClassOf rdf:resource="&mebase;Interface"/>
  <rdfs:isDefinedBy rdf:resource="&meac;"/>
</owl:Class>


<!-- =============== myExperiment Entity Classes ================ -->

 <owl:Class rdf:about="Attribution">
  <rdfs:label>Attribution</rdfs:label>
  <rdfs:comment>An Attribution to a Contribution from another Contribution</
  rdfs:comment>
  <owl:disjointWith rdf:resource="&mebase;Request"/>
  <owl:disjointWith rdf:resource="&mebase;Announcement"/>
  <owl:disjointWith rdf:resource="&mebase;Contribution"/>
  <owl:disjointWith rdf:resource="&mebase;Message"/>
  <owl:disjointWith rdf:resource="Creditation"/>
  <rdfs:subClassOf rdf:resource="&mebase;Submission"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="has-attributable" />
      <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="attributes" />
      <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&dcterms;modified" />
      <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:isDefinedBy rdf:resource="&meac;"/>
</owl:Class>

<owl:Class rdf:about="Creditation">
  <rdfs:label>Creditation</rdfs:label>
  <rdfs:comment>A Creditation from a Contribution to an Actor</rdfs:comment>
  <owl:disjointWith rdf:resource="&mebase;Request"/>
  <owl:disjointWith rdf:resource="&mebase;Announcement"/>
  <owl:disjointWith rdf:resource="&mebase;Contribution"/>
  <owl:disjointWith rdf:resource="&mebase;Message"/>
  <owl:disjointWith rdf:resource="Attribution"/>
  <rdfs:subClassOf rdf:resource="&mebase;Submission"/>
   <rdfs:subClassOf>
    <owl:Restriction>
```

```
      <owl:onProperty rdf:resource="has-creditable" />
      <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="credits" />
      <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&dcterms;modified" />
      <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:isDefinedBy rdf:resource="&meac;"/>
</owl:Class>

<!-- =============== Object Properties ============== -->

<owl:ObjectProperty rdf:about="has-attribution">
  <rdfs:label>has-attribution</rdfs:label>
  <rdfs:comment>An Attributable has an attribution for another Attributable</
  rdfs:comment>
  <rdfs:domain rdf:resource="Attributable"/>
  <rdfs:range rdf:resource="Attributable"/>
  <rdfs:isDefinedBy rdf:resource="&meac;"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="has-attributable">
  <rdfs:label>has-attributable</rdfs:label>
  <rdfs:comment>An Attribution has an Atrributable</rdfs:comment>
  <rdfs:domain rdf:resource="Attribution"/>
  <rdfs:range rdf:resource="Attributable"/>
  <rdfs:isDefinedBy rdf:resource="&meac;"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="attributes">
  <rdfs:label>attributes</rdfs:label>
  <rdfs:comment>An Attribution attributes an Attributable object</rdfs:comment>
  <rdfs:domain rdf:resource="Attribution"/>
  <rdfs:range rdf:resource="Attributable"/>
  <rdfs:isDefinedBy rdf:resource="&meac;"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="gives-credit-to">
  <rdfs:label>gives-credit-to</rdfs:label>
  <rdfs:comment>A Creditable gives credit to an Actor</rdfs:comment>
  <rdfs:domain rdf:resource="Creditable"/>
  <rdfs:range rdf:resource="&mebase;Actor"/>
  <rdfs:isDefinedBy rdf:resource="&meac;"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="credits">
  <rdfs:label>credits</rdfs:label>
  <rdfs:comment>A Creditation credits an Actor</rdfs:comment>
  <rdfs:domain rdf:resource="Creditation"/>
  <rdfs:range rdf:resource="&mebase;Actor"/>
```

```
    <rdfs:isDefinedBy rdf:resource="&meac;"/>
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:about="has-creditable">
    <rdfs:label>has-creditable</rdfs:label>
    <rdfs:comment>A Creditation has-creditable</rdfs:comment>
    <rdfs:domain rdf:resource="Creditation"/>
    <rdfs:range rdf:resource="Creditable"/>
    <rdfs:isDefinedBy rdf:resource="&meac;"/>
  </owl:ObjectProperty>

</rdf:RDF>
```

Listing A.4: myExperiment's Attribution and Credit Module

A.2.4 Annotations Ontology Module

http://rdf.myexperiment.org/ontologies/annotations/ as of 16/10/2011.

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE rdf:RDF [
 <!ENTITY rdf  'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
 <!ENTITY rdfs 'http://www.w3.org/2000/01/rdf-schema#'>
 <!ENTITY owl 'http://www.w3.org/2002/07/owl#'>
 <!ENTITY xsd 'http://www.w3.org/2001/XMLSchema#'>
 <!ENTITY dc 'http://purl.org/dc/elements/1.1/'>
 <!ENTITY dcterms 'http://purl.org/dc/terms/'>
 <!ENTITY foaf 'http://xmlns.com/foaf/0.1/'>
 <!ENTITY skos 'http://www.w3.org/2004/02/skos/core#'>
 <!ENTITY mebase 'http://rdf.myexperiment.org/ontologies/base/'>
 <!ENTITY meannot 'http://rdf.myexperiment.org/ontologies/annotations/'>
]>

<rdf:RDF xml:base                 ="&meannot;"
               xmlns                      ="&meannot;"
         xmlns:mebase ="&mebase;"
         xmlns:rdf        ="&rdf;"
         xmlns:rdfs             ="&rdfs;"
         xmlns:owl              ="&owl;"
         xmlns:dc                 ="&dc;"
               xmlns:dcterms="&dcterms;"
               xmlns:foaf               ="&foaf;"
               xmlns:skos               ="&skos;"
         xmlns:xsd              ="&xsd;"
>
  <!-- =============== Description ==================== -->

  <owl:Ontology rdf:about="&meannot;">
    <rdfs:label>myExperiment Annotations v1.0</rdfs:label>
    <rdfs:comment>This provides the different types of annotations used in
    myExperiment.</rdfs:comment>
    <dc:language>en</dc:language>
    <dc:title xml:lang="en">The myExperiment Annotations Ontology</dc:title>
    <dc:creator rdf:resource="http://rdf.ecs.soton.ac.uk/person/9421"/>
    <dc:contributor rdf:datatype="http://www.w3.org/2001/XMLSchema#string">David R
    Newman</dc:contributor>
```

```
    <dc:publisher rdf:resource="http://rdf.myexperiment.org"/>
    <dc:date rdf:datatype="http://www.w3.org/2001/XMLSchema#date">2009-01-28</dc:date>
    <owl:versionInfo>$Date: 2011/10/03 $</owl:versionInfo>
    <dc:format rdf:datatype="http://www.w3.org/2001/XMLSchema#string">rdf/xml</
    dc:format>
  </owl:Ontology>

<!-- =============== Annotation Properties ========= -->

  <rdf:Description rdf:about="&dc;language">
    <rdf:type rdf:resource="&owl;AnnotationProperty"/>
  </rdf:Description>

  <rdf:Description rdf:about="&dc;title">
    <rdf:type rdf:resource="&owl;AnnotationProperty"/>
  </rdf:Description>

  <rdf:Description rdf:about="&dc;creator">
    <rdf:type rdf:resource="&owl;AnnotationProperty"/>
  </rdf:Description>

  <rdf:Description rdf:about="&dc;contributor">
    <rdf:type rdf:resource="&owl;AnnotationProperty"/>
  </rdf:Description>

  <rdf:Description rdf:about="&dc;publisher">
    <rdf:type rdf:resource="&owl;AnnotationProperty"/>
  </rdf:Description>

  <rdf:Description rdf:about="&dc;date">
    <rdf:type rdf:resource="&owl;AnnotationProperty"/>
  </rdf:Description>

  <rdf:Description rdf:about="&dc;format">
    <rdf:type rdf:resource="&owl;AnnotationProperty"/>
  </rdf:Description>


  <!-- ========= OWL-DL Compliance statements ======== -->

  <rdf:Description rdf:about="&dcterms;BibliographicCitation">
    <rdf:type rdf:resource="&owl;Class"/>
  </rdf:Description>

  <rdf:Description rdf:about="&dcterms;modified">
    <rdf:type rdf:resource="&owl;DatatypeProperty"/>
    <rdfs:range rdf:resource="&xsd;dateTime"/>
  </rdf:Description>

  <rdf:Description rdf:about="&dcterms;title">
    <rdf:type rdf:resource="&owl;DatatypeProperty"/>
    <rdfs:range rdf:resource="&xsd;string"/>
  </rdf:Description>

  <rdf:Description rdf:about="&skos;ConceptScheme">
    <rdf:type rdf:resource="&owl;Class"/>
  </rdf:Description>

  <rdf:Description rdf:about="&mebase;text">
```

```
  <rdf:type rdf:resource="&owl;DatatypeProperty"/>
</rdf:Description>

<rdf:Description rdf:about="&mebase;Annotatable">
  <rdf:type rdf:resource="&owl;Class"/>
</rdf:Description>

<rdf:Description rdf:about="&mebase;Annotation">
  <rdf:type rdf:resource="&owl;Class"/>
</rdf:Description>

<rdf:Description rdf:about="&mebase;Contribution">
  <rdf:type rdf:resource="&owl;Class"/>
</rdf:Description>

<rdf:Description rdf:about="&mebase;has-annotation">
  <rdf:type rdf:resource="&owl;ObjectProperty"/>
</rdf:Description>

<rdf:Description rdf:about="&mebase;Submission">
  <rdf:type rdf:resource="&owl;Class"/>
</rdf:Description>

<rdf:Description rdf:about="&mebase;User">
  <rdf:type rdf:resource="&owl;Class"/>
</rdf:Description>


<!-- ====================== Interfaces ====================== -->

<owl:Class rdf:about="Citationable">
  <rdfs:label>Citationable</rdfs:label>
  <rdfs:comment>An object can be annotated with a Citation</rdfs:comment>
  <rdfs:subClassOf rdf:resource="&mebase;Annotatable"/>
  <rdfs:isDefinedBy rdf:resource="&meannot;"/>
</owl:Class>

<owl:Class rdf:about="Commentable">
  <rdfs:label>Commentable</rdfs:label>
  <rdfs:comment>An object can be annotated with a Comment</rdfs:comment>
  <rdfs:subClassOf rdf:resource="&mebase;Annotatable"/>
  <rdfs:isDefinedBy rdf:resource="&meannot;"/>
</owl:Class>

<owl:Class rdf:about="Favouritable">
  <rdfs:label>Favouritable</rdfs:label>
  <rdfs:comment>An object can be made a Favourite</rdfs:comment>
  <rdfs:subClassOf rdf:resource="&mebase;Annotatable"/>
  <rdfs:isDefinedBy rdf:resource="&meannot;"/>
</owl:Class>

<owl:Class rdf:about="Reviewable">
  <rdfs:label>Reviewable</rdfs:label>
  <rdfs:comment>An object can be annotated with a Review</rdfs:comment>
  <rdfs:subClassOf rdf:resource="&mebase;Annotatable"/>
  <rdfs:isDefinedBy rdf:resource="&meannot;"/>
</owl:Class>

<owl:Class rdf:about="Rateable">
```

```
  <rdfs:label>Rateable</rdfs:label>
  <rdfs:comment>An object can be annotated with a Rating</rdfs:comment>
  <rdfs:subClassOf rdf:resource="&mebase;Annotatable"/>
  <rdfs:isDefinedBy rdf:resource="&meannot;"/>
</owl:Class>

<owl:Class rdf:about="Taggable">
  <rdfs:label>Taggable</rdfs:label>
  <rdfs:comment>An object that can be tagged</rdfs:comment>
  <rdfs:subClassOf rdf:resource="&mebase;Annotatable"/>
  <rdfs:isDefinedBy rdf:resource="&meannot;"/>
</owl:Class>


<!-- ================== Abstract Classes ================== -->

<owl:Class rdf:about="Citation">
  <rdfs:label>Citation</rdfs:label>
  <rdfs:comment>A Citation made by a User about a piece of work associated with the
  Citationable object</rdfs:comment>
  <rdfs:subClassOf rdf:resource="&dcterms;BibliographicCitation"/>
  <rdfs:subClassOf rdf:resource="&mebase;Annotation"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&dcterms;title" />
      <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&dcterms;modified" />
      <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:isDefinedBy rdf:resource="&meannot;"/>
</owl:Class>

<owl:Class rdf:about="Comment">
  <rdfs:label>Comment</rdfs:label>
  <rdfs:comment>A Comment made by a User about Contribution</rdfs:comment>
  <rdfs:subClassOf rdf:resource="&mebase;Annotation"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&mebase;text" />
      <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&dcterms;title" />
      <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1</
  owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:isDefinedBy rdf:resource="&meannot;"/>
</owl:Class>

<owl:Class rdf:about="Favourite">
  <rdfs:label>Favourite</rdfs:label>
```

```
    <rdfs:comment>A Favourite created by a User of a Favouritable object</rdfs:comment
    >
    <rdfs:subClassOf rdf:resource="&mebase;Annotation"/>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="&dcterms;title" />
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:isDefinedBy rdf:resource="&meannot;"/>
  </owl:Class>

  <owl:Class rdf:about="Rating">
    <rdfs:label>Rating</rdfs:label>
    <rdfs:comment>A Rating for a Rateable object</rdfs:comment>
    <rdfs:subClassOf rdf:resource="&mebase;Annotation"/>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="rating-score" />
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:isDefinedBy rdf:resource="&meannot;"/>
  </owl:Class>

  <owl:Class rdf:about="Review">
    <rdfs:label>Review</rdfs:label>
    <rdfs:comment>A Review for a Reviewable object</rdfs:comment>
    <rdfs:subClassOf rdf:resource="&mebase;Annotation"/>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="&mebase;text" />
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="&dcterms;title" />
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="&dcterms;modified" />
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:isDefinedBy rdf:resource="&meannot;"/>
  </owl:Class>

  <owl:Class rdf:about="Tag">
    <rdfs:label>Tag</rdfs:label>
    <rdfs:comment>A Tag to a word or phrase</rdfs:comment>
    <rdfs:subClassOf rdf:resource="&mebase;Submission"/>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="&dcterms;title" />
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
      </owl:Restriction>
```

```
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&dcterms;modified" />
      <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:isDefinedBy rdf:resource="&meannot;"/>
</owl:Class>

<owl:Class rdf:about="Tagging">
  <rdfs:label>Taggings</rdfs:label>
  <rdfs:comment>A Taggable object can have Taggings</rdfs:comment>
  <rdfs:subClassOf rdf:resource="&mebase;Annotation" />
  <rdfs:isDefinedBy rdf:resource="&meannot;"/>
</owl:Class>

<owl:Class rdf:about="Vocabulary">
  <rdfs:label>Vocabulary</rdfs:label>
  <rdfs:comment>A Vocabulary of Tags</rdfs:comment>
  <rdfs:subClassOf rdf:resource="&mebase;Contribution"/>
  <rdfs:subClassOf rdf:resource="&skos;ConceptScheme"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&dcterms;title" />
      <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:isDefinedBy rdf:resource="&meannot;"/>
</owl:Class>


<!-- =============== Object Properties ============== -->

<owl:ObjectProperty rdf:about="citation-url">
  <rdfs:label>citation-url</rdfs:label>
  <rdfs:comment>The URL where a citation is located</rdfs:comment>
  <rdfs:domain rdf:resource="Citation"/>
  <rdfs:isDefinedBy rdf:resource="&meannot;"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="uses-tag">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:label>uses-tag</rdfs:label>
  <rdfs:comment>A Tagging uses a Tag</rdfs:comment>
  <rdfs:domain rdf:resource="Tagging"/>
  <rdfs:range rdf:resource="Tag"/>
  <owl:inverseOf rdf:resource="for-tagging"/>
  <rdfs:isDefinedBy rdf:resource="&meannot;"/>
</owl:ObjectProperty>

 <owl:ObjectProperty rdf:about="for-tagging">
  <rdfs:label>for-tagging</rdfs:label>
  <rdfs:comment>A Tag may be for a tagging</rdfs:comment>
  <rdfs:domain rdf:resource="Tag"/>
  <rdfs:range rdf:resource="Tagging"/>
  <owl:inverseOf rdf:resource="uses-tag"/>
  <rdfs:isDefinedBy rdf:resource="&meannot;"/>
</owl:ObjectProperty>
```

```
<owl:ObjectProperty rdf:about="has-tag">
  <rdfs:label>has-tag</rdfs:label>
  <rdfs:comment>A Taggable object has Tags that tag it</rdfs:comment>
  <rdfs:domain rdf:resource="Taggable"/>
  <rdfs:range rdf:resource="Tag"/>
  <rdfs:isDefinedBy rdf:resource="&meannot;"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="has-citation">
  <rdfs:subPropertyOf rdf:resource="&mebase;has-annotation"/>
  <rdfs:label>has-citation</rdfs:label>
  <rdfs:comment>A Citationable object has Citations</rdfs:comment>
  <rdfs:domain rdf:resource="Citationable"/>
  <rdfs:range rdf:resource="Citation"/>
  <rdfs:isDefinedBy rdf:resource="&meannot;"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="has-comment">
  <rdfs:subPropertyOf rdf:resource="&mebase;has-annotation"/>
  <rdfs:label>has-comment</rdfs:label>
  <rdfs:comment>A Commentable object has Comments</rdfs:comment>
  <rdfs:domain rdf:resource="Commentable"/>
  <rdfs:range rdf:resource="Comment"/>
  <rdfs:isDefinedBy rdf:resource="&meannot;"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="has-favourite">
  <rdfs:label>has-favourite</rdfs:label>
  <rdfs:comment>A Favouritable object has Favourites</rdfs:comment>
  <rdfs:domain rdf:resource="&mebase;User"/>
  <rdfs:range rdf:resource="Favourite"/>
  <rdfs:isDefinedBy rdf:resource="&meannot;"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="has-rating">
  <rdfs:subPropertyOf rdf:resource="&mebase;has-annotation"/>
  <rdfs:label>has-rating</rdfs:label>
  <rdfs:comment>A Rateable object has Rating</rdfs:comment>
  <rdfs:domain rdf:resource="Rateable"/>
  <rdfs:range rdf:resource="Rating"/>
  <rdfs:isDefinedBy rdf:resource="&meannot;"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="has-review">
  <rdfs:subPropertyOf rdf:resource="&mebase;has-annotation"/>
  <rdfs:label>has-review</rdfs:label>
  <rdfs:comment>A Reviewable object has Reviews</rdfs:comment>
  <rdfs:domain rdf:resource="Reviewable"/>
  <rdfs:range rdf:resource="Review"/>
  <rdfs:isDefinedBy rdf:resource="&meannot;"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="has-tagging">
  <rdfs:subPropertyOf rdf:resource="&mebase;has-annotation"/>
  <rdfs:label>has-tagging</rdfs:label>
  <rdfs:comment>A Taggable object has Taggings</rdfs:comment>
  <rdfs:domain rdf:resource="Taggable"/>
  <rdfs:range rdf:resource="Tagging"/>
```

```
    <rdfs:isDefinedBy rdf:resource="&meannot;"/>
  </owl:ObjectProperty>


<!-- =============== Datatype Properties ============== -->

  <owl:DatatypeProperty rdf:about="accessed-at">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:label>accessed-at</rdfs:label>
    <rdfs:comment>A piece of work cited by a Citation was accessed-at a particular
    dateTime</rdfs:comment>
    <rdfs:domain rdf:resource="Citation"/>
    <rdfs:range rdf:resource="&xsd;dateTime"/>
    <rdfs:isDefinedBy rdf:resource="&meannot;"/>
  </owl:DatatypeProperty>

  <owl:DatatypeProperty rdf:about="authors">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:label>authors</rdfs:label>
    <rdfs:comment>A piece of work cited by a Citation has authors that are represented
     by a string</rdfs:comment>
    <rdfs:domain rdf:resource="Citation"/>
    <rdfs:range rdf:resource="&xsd;string"/>
    <rdfs:isDefinedBy rdf:resource="&meannot;"/>
  </owl:DatatypeProperty>

  <owl:DatatypeProperty rdf:about="tag-count">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:label>tag-count</rdfs:label>
    <rdfs:comment>A Tag has a count of the number of times it is used</rdfs:comment>
    <rdfs:domain rdf:resource="Tag"/>
    <rdfs:range rdf:resource="&xsd;nonNegativeInteger"/>
    <rdfs:isDefinedBy rdf:resource="&meannot;"/>
  </owl:DatatypeProperty>

  <owl:DatatypeProperty rdf:about="isbn">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:label>isbn</rdfs:label>
    <rdfs:comment>An piece of work cited by a Citation may have a isbn</rdfs:comment>
    <rdfs:domain rdf:resource="Citation"/>
    <rdfs:range rdf:resource="&xsd;string"/>
    <rdfs:isDefinedBy rdf:resource="&meannot;"/>
  </owl:DatatypeProperty>

  <owl:DatatypeProperty rdf:about="issn">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:label>issn</rdfs:label>
    <rdfs:comment>An piece of work cited by a Citation may have an issn</rdfs:comment>
    <rdfs:domain rdf:resource="Citation"/>
    <rdfs:range rdf:resource="&xsd;string"/>
    <rdfs:isDefinedBy rdf:resource="&meannot;"/>
  </owl:DatatypeProperty>

  <owl:DatatypeProperty rdf:about="publication">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:label>publication</rdfs:label>
    <rdfs:comment>An piece of work cited by a Citation may be from a publication</
    rdfs:comment>
    <rdfs:domain rdf:resource="Citation"/>
```

```
    <rdfs:range rdf:resource="&xsd;string"/>
    <rdfs:isDefinedBy rdf:resource="&meannot;"/>
  </owl:DatatypeProperty>

  <owl:DatatypeProperty rdf:about="published-at">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:label>published-at</rdfs:label>
    <rdfs:comment>A piece of work cited by a Citation was published-at a particular
    dateTime</rdfs:comment>
    <rdfs:domain rdf:resource="Citation"/>
    <rdfs:range rdf:resource="&xsd;dateTime"/>
    <rdfs:isDefinedBy rdf:resource="&meannot;"/>
  </owl:DatatypeProperty>

  <owl:DatatypeProperty rdf:about="rating-score">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:label>rating-score</rdfs:label>
    <rdfs:comment>A Rating has a rating-score between 1 and 5</rdfs:comment>
    <rdfs:domain rdf:resource="Rating"/>
    <rdfs:range rdf:resource="&xsd;positiveInteger"/>
    <rdfs:isDefinedBy rdf:resource="&meannot;"/>
  </owl:DatatypeProperty>

</rdf:RDF>
```

Listing A.5: myExperiment's Annotations Module

A.2.5 Contributions Ontology Module

`http://rdf.myexperiment.org/ontologies/contributions/` as of 16/10/2011.

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE rdf:RDF [
 <!ENTITY rdf  'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
 <!ENTITY rdfs 'http://www.w3.org/2000/01/rdf-schema#'>
 <!ENTITY owl 'http://www.w3.org/2002/07/owl#'>
 <!ENTITY xsd 'http://www.w3.org/2001/XMLSchema#'>
 <!ENTITY dc 'http://purl.org/dc/elements/1.1/'>
 <!ENTITY dcterms 'http://purl.org/dc/terms/'>
 <!ENTITY ore 'http://www.openarchives.org/ore/terms/'>
 <!ENTITY mebase 'http://rdf.myexperiment.org/ontologies/base/'>
 <!ENTITY meac 'http://rdf.myexperiment.org/ontologies/attrib_credit/'>
 <!ENTITY meannot 'http://rdf.myexperiment.org/ontologies/annotations/'>
 <!ENTITY meexp 'http://rdf.myexperiment.org/ontologies/experiments/'>
 <!ENTITY mecontrib 'http://rdf.myexperiment.org/ontologies/contributions/'>
]>

<rdf:RDF xml:base                ="&mecontrib;"
               xmlns                      ="&mecontrib;"
         xmlns:mebase   ="&mebase;"
               xmlns:meac               ="&meac;"
               xmlns:meannot="&meannot;"
               xmlns:meexp      ="&meexp;"
         xmlns:rdf              ="&rdf;"
         xmlns:rdfs             ="&rdfs;"
         xmlns:owl              ="&owl;"
```

```
         xmlns:dc                ="&dc;"
         xmlns:dcterms="&dcterms;"
         xmlns:ore              ="&ore;"
         xmlns:xsd              ="&xsd;"
>
  <!-- =============== Description =================== -->

  <owl:Ontology rdf:about="&mecontrib;">
    <rdfs:label>myExperiment Contributions v1.0</rdfs:label>
    <rdfs:comment>This provides the different types of contributions used in
    myExperiment.</rdfs:comment>
    <dc:language>en</dc:language>
    <dc:title xml:lang="en">The myExperiment Contributions Ontology</dc:title>
    <dc:creator rdf:resource="http://rdf.ecs.soton.ac.uk/person/9421"/>
    <dc:contributor rdf:datatype="http://www.w3.org/2001/XMLSchema#string">David R
    Newman</dc:contributor>
    <dc:publisher rdf:resource="http://rdf.myexperiment.org"/>
    <dc:date rdf:datatype="http://www.w3.org/2001/XMLSchema#date">2009-01-28</dc:date>
    <owl:versionInfo>$Date: 2011/02/20 $</owl:versionInfo>
    <dc:format rdf:datatype="http://www.w3.org/2001/XMLSchema#string">rdf/xml</
    dc:format>
  </owl:Ontology>

<!-- ============= Annotation Properties =========== -->

  <rdf:Description rdf:about="&dc;language">
    <rdf:type rdf:resource="&owl;AnnotationProperty"/>
  </rdf:Description>

  <rdf:Description rdf:about="&dc;title">
    <rdf:type rdf:resource="&owl;AnnotationProperty"/>
  </rdf:Description>

  <rdf:Description rdf:about="&dc;creator">
    <rdf:type rdf:resource="&owl;AnnotationProperty"/>
  </rdf:Description>

  <rdf:Description rdf:about="&dc;contributor">
    <rdf:type rdf:resource="&owl;AnnotationProperty"/>
  </rdf:Description>

  <rdf:Description rdf:about="&dc;publisher">
    <rdf:type rdf:resource="&owl;AnnotationProperty"/>
  </rdf:Description>

  <rdf:Description rdf:about="&dc;date">
    <rdf:type rdf:resource="&owl;AnnotationProperty"/>
  </rdf:Description>

  <rdf:Description rdf:about="&dc;format">
    <rdf:type rdf:resource="&owl;AnnotationProperty"/>
  </rdf:Description>


  <!-- ========= OWL-DL Compliance statements ======== -->

  <rdf:Description rdf:about="&dcterms;title">
    <rdf:type rdf:resource="&owl;DatatypeProperty"/>
    <rdfs:range rdf:resource="&xsd;string"/>
```

```
</rdf:Description>

<rdf:Description rdf:about="&mebase;Upload">
  <rdf:type rdf:resource="&owl;Class"/>
</rdf:Description>

<rdf:Description rdf:about="&mebase;Contribution">
  <rdf:type rdf:resource="&owl;Class"/>
</rdf:Description>

<rdf:Description rdf:about="&mebase;User">
  <rdf:type rdf:resource="&owl;Class"/>
</rdf:Description>

<rdf:Description rdf:about="&mebase;Version">
  <rdf:type rdf:resource="&owl;Class"/>
</rdf:Description>

<rdf:Description rdf:about="&mebase;Versionable">
  <rdf:type rdf:resource="&owl;Class"/>
</rdf:Description>

<rdf:Description rdf:about="&meannot;Commentable">
    <rdf:type rdf:resource="&owl;Class"/>
</rdf:Description>

<rdf:Description rdf:about="&meannot;Citationable">
  <rdf:type rdf:resource="&owl;Class"/>
</rdf:Description>

<rdf:Description rdf:about="&meannot;Favouritable">
  <rdf:type rdf:resource="&owl;Class"/>
</rdf:Description>

<rdf:Description rdf:about="&meannot;Rateable">
  <rdf:type rdf:resource="&owl;Class"/>
</rdf:Description>

<rdf:Description rdf:about="&meannot;Reviewable">
  <rdf:type rdf:resource="&owl;Class"/>
</rdf:Description>

<rdf:Description rdf:about="&meannot;Tag">
  <rdf:type rdf:resource="&owl;Class"/>
</rdf:Description>

<rdf:Description rdf:about="&meannot;Taggable">
  <rdf:type rdf:resource="&owl;Class"/>
</rdf:Description>

<rdf:Description rdf:about="&meac;Attributable">
  <rdf:type rdf:resource="&owl;Class"/>
</rdf:Description>

<rdf:Description rdf:about="&meac;Creditable">
  <rdf:type rdf:resource="&owl;Class"/>
</rdf:Description>
```

```
<!-- =============== myExperiment Entity Classes ================= -->

<owl:Class rdf:about="File">
  <rdfs:label>File</rdfs:label>
  <rdfs:comment>A File uploaded by an Actor</rdfs:comment>
  <owl:disjointWith rdf:resource="AbstractWorkflow"/>
  <rdfs:subClassOf rdf:resource="&mebase;Upload"/>
  <rdfs:subClassOf rdf:resource="&meannot;Commentable"/>
  <rdfs:subClassOf rdf:resource="&meannot;Rateable"/>
  <rdfs:subClassOf rdf:resource="&meannot;Taggable"/>
  <rdfs:subClassOf rdf:resource="&meac;Attributable"/>
  <rdfs:subClassOf rdf:resource="&meac;Creditable"/>
  <rdfs:subClassOf rdf:resource="&meannot;Favouritable"/>
  <rdfs:isDefinedBy rdf:resource="&mecontrib;"/>
</owl:Class>

<owl:Class rdf:about="AbstractWorkflow">
  <rdfs:label>AbstractWorkflow</rdfs:label>
  <rdfs:comment>An AbstractWorkflow from which Workflow and WorkflowVersion can be
  templated on</rdfs:comment>
  <owl:disjointWith rdf:resource="File"/>
  <rdfs:subClassOf rdf:resource="&meannot;Citationable"/>
  <rdfs:subClassOf rdf:resource="&meannot;Commentable"/>
  <rdfs:subClassOf rdf:resource="&meannot;Reviewable"/>
  <rdfs:subClassOf rdf:resource="&meannot;Rateable"/>
  <rdfs:subClassOf rdf:resource="&meannot;Taggable"/>
  <rdfs:subClassOf rdf:resource="&meac;Attributable"/>
  <rdfs:subClassOf rdf:resource="&meac;Creditable"/>
  <rdfs:subClassOf rdf:resource="&meannot;Favouritable"/>
  <rdfs:subClassOf rdf:resource="&mebase;Upload"/>
   <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="preview" />
      <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1</
  owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="svg" />
      <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1</
  owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="thumbnail" />
      <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1</
  owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="thumbnail-big" />
      <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1</
  owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:isDefinedBy rdf:resource="&mecontrib;"/>
```

```
  </owl:Class>

  <owl:Class rdf:about="Workflow">
    <rdfs:label>Workflow</rdfs:label>
    <rdfs:comment>A Workflow uploaded by an Actor</rdfs:comment>
    <owl:disjointWith rdf:resource="WorkflowVersion"/>
    <rdfs:subClassOf rdf:resource="&mebase;Versionable"/>
    <rdfs:subClassOf rdf:resource="AbstractWorkflow"/>
    <rdfs:isDefinedBy rdf:resource="&mecontrib;"/>
  </owl:Class>

  <owl:Class rdf:about="WorkflowVersion">
    <rdfs:label>WorkflowVersion</rdfs:label>
    <rdfs:comment>A Version of a Workflow</rdfs:comment>
    <owl:disjointWith rdf:resource="Workflow"/>
    <rdfs:subClassOf rdf:resource="&mebase;Version"/>
    <rdfs:subClassOf rdf:resource="AbstractWorkflow"/>
    <rdfs:isDefinedBy rdf:resource="&mecontrib;"/>
  </owl:Class>

<!-- =============== Datatype Properties ============== -->


<!-- ================ Object Properties =============== -->

  <owl:ObjectProperty rdf:about="preview">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:label>preview</rdfs:label>
    <rdfs:comment>A Workflow may have a uri that resolves to a preview image
    representation of it</rdfs:comment>
    <rdfs:domain rdf:resource="AbstractWorkflow"/>
    <rdfs:isDefinedBy rdf:resource="&mecontrib;"/>
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:about="svg">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:label>svg</rdfs:label>
    <rdfs:comment>A Workflow may have a uri that resolves to an svg representation of
    it</rdfs:comment>
    <rdfs:domain rdf:resource="AbstractWorkflow"/>
    <rdfs:isDefinedBy rdf:resource="&mecontrib;"/>
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:about="thumbnail">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:label>thumbnail</rdfs:label>
    <rdfs:comment>A link to a thumbnail image of the Workflow</rdfs:comment>
    <rdfs:domain rdf:resource="AbstractWorkflow"/>
    <rdfs:isDefinedBy rdf:resource="&mecontrib;"/>
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:about="thumbnail-big">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:label>thumbnail-big</rdfs:label>
    <rdfs:comment>A link to a big thumbnail image of the Workflow</rdfs:comment>
    <rdfs:domain rdf:resource="AbstractWorkflow"/>
    <rdfs:isDefinedBy rdf:resource="&mecontrib;"/>
  </owl:ObjectProperty>
```

```
  <owl:ObjectProperty rdf:about="last-edited-by">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:label>last-edited-by</rdfs:label>
    <rdfs:comment>A Workflow will have been last edited by a particular User</
    rdfs:comment>
    <rdfs:domain rdf:resource="AbstractWorkflow"/>
    <rdfs:range rdf:resource="&mebase;User"/>
    <rdfs:isDefinedBy rdf:resource="&mecontrib;"/>
  </owl:ObjectProperty>

</rdf:RDF>
```

LISTING A.6: myExperiment's Contributions Module

A.2.6 Packs Ontology Module

`http://rdf.myexperiment.org/ontologies/packs/` as of 16/10/2011.

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE rdf:RDF [
 <!ENTITY rdf  'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
 <!ENTITY rdfs 'http://www.w3.org/2000/01/rdf-schema#'>
 <!ENTITY owl 'http://www.w3.org/2002/07/owl#'>
 <!ENTITY xsd 'http://www.w3.org/2001/XMLSchema#'>
 <!ENTITY dc 'http://purl.org/dc/elements/1.1/'>
 <!ENTITY dcterms 'http://purl.org/dc/terms/'>
 <!ENTITY ore 'http://www.openarchives.org/ore/terms/'>
 <!ENTITY sioc 'http://rdfs.org/sioc/ns#'>
 <!ENTITY mebase 'http://rdf.myexperiment.org/ontologies/base/'>
 <!ENTITY meannot 'http://rdf.myexperiment.org/ontologies/annotations/'>
 <!ENTITY mepack 'http://rdf.myexperiment.org/ontologies/packs/'>
]>

<rdf:RDF xml:base                =&mepack;"
         xmlns                   =&mepack;"
               xmlns:mebase     ="&mebase;"
               xmlns:meannot="&meannot;"
         xmlns:rdf              ="&rdf;"
         xmlns:rdfs             ="&rdfs;"
         xmlns:owl              ="&owl;"
         xmlns:dc                 ="&dc;"
               xmlns:dcterms="&dcterms;"
         xmlns:ore    ="&ore;"
         xmlns:sioc             ="&sioc;"
         xmlns:xsd              ="&xsd;"
>
  <!-- =============== Description ==================== -->

  <owl:Ontology rdf:about="&mepack;">
    <rdfs:label>myExperiment Packs v1.1</rdfs:label>
    <rdfs:comment>This facilitates the use of packs to aggregate contributions and
    remote urls together and link these items together with relationships.</
    rdfs:comment>
    <dc:language>en</dc:language>
    <dc:title xml:lang="en">The myExperiment Packs Ontology</dc:title>
    <dc:creator rdf:resource="http://rdf.ecs.soton.ac.uk/person/9421"/>
```

```
    <dc:contributor rdf:datatype="http://www.w3.org/2001/XMLSchema#string">David R
    Newman</dc:contributor>
    <dc:publisher rdf:resource="http://rdf.myexperiment.org"/>
    <dc:date rdf:datatype="http://www.w3.org/2001/XMLSchema#date">2009-01-28</dc:date>
    <owl:versionInfo>$Date: 2011/10/03 $</owl:versionInfo>
    <dc:format rdf:datatype="http://www.w3.org/2001/XMLSchema#string">rdf/xml</
    dc:format>
  </owl:Ontology>

<!-- ============= Annotation Properties =========== -->

  <rdf:Description rdf:about="&dc;language">
    <rdf:type rdf:resource="&owl;AnnotationProperty"/>
  </rdf:Description>

  <rdf:Description rdf:about="&dc;title">
    <rdf:type rdf:resource="&owl;AnnotationProperty"/>
  </rdf:Description>

  <rdf:Description rdf:about="&dc;creator">
    <rdf:type rdf:resource="&owl;AnnotationProperty"/>
  </rdf:Description>

  <rdf:Description rdf:about="&dc;contributor">
    <rdf:type rdf:resource="&owl;AnnotationProperty"/>
  </rdf:Description>

  <rdf:Description rdf:about="&dc;publisher">
    <rdf:type rdf:resource="&owl;AnnotationProperty"/>
  </rdf:Description>

  <rdf:Description rdf:about="&dc;date">
    <rdf:type rdf:resource="&owl;AnnotationProperty"/>
  </rdf:Description>

  <rdf:Description rdf:about="&dc;format">
    <rdf:type rdf:resource="&owl;AnnotationProperty"/>
  </rdf:Description>


  <!-- ========= OWL-DL Compliance statements ======== -->

  <rdf:Description rdf:about="&dcterms;description">
    <rdf:type rdf:resource="&owl;DatatypeProperty"/>
    <rdfs:range rdf:resource="&xsd;string"/>
  </rdf:Description>

  <rdf:Description rdf:about="&dcterms;modified">
    <rdf:type rdf:resource="&owl;DatatypeProperty"/>
    <rdfs:range rdf:resource="&xsd;dateTime"/>
  </rdf:Description>

  <rdf:Description rdf:about="&ore;aggregates">
    <rdf:type rdf:resource="&owl;ObjectProperty"/>
  </rdf:Description>

  <rdf:Description rdf:about="&dcterms;title">
    <rdf:type rdf:resource="&owl;DatatypeProperty"/>
    <rdfs:range rdf:resource="&xsd;string"/>
```

```
</rdf:Description>

<rdf:Description rdf:about="&rdf;object">
  <rdf:type rdf:resource="&owl;ObjectProperty"/>
</rdf:Description>

<rdf:Description rdf:about="&rdf;predicate">
  <rdf:type rdf:resource="&owl;ObjectProperty"/>
</rdf:Description>

<rdf:Description rdf:about="&rdf;subject">
  <rdf:type rdf:resource="&owl;ObjectProperty"/>
</rdf:Description>

<rdf:Description rdf:about="&sioc;has_owner">
  <rdf:type rdf:resource="&owl;ObjectProperty"/>
</rdf:Description>

<rdf:Description rdf:about="&mebase;Annotation">
  <rdf:type rdf:resource="&owl;Class"/>
</rdf:Description>

<rdf:Description rdf:about="&mebase;Submission">
  <rdf:type rdf:resource="&owl;Class"/>
</rdf:Description>

<rdf:Description rdf:about="&mebase;Contribution">
  <rdf:type rdf:resource="&owl;Class"/>
</rdf:Description>

<rdf:Description rdf:about="&mebase;Upload">
  <rdf:type rdf:resource="&owl;Class"/>
</rdf:Description>

<rdf:Description rdf:about="&meannot;Commentable">
  <rdf:type rdf:resource="&owl;Class"/>
</rdf:Description>

<rdf:Description rdf:about="&meannot;Favouritable">
  <rdf:type rdf:resource="&owl;Class"/>
</rdf:Description>

<rdf:Description rdf:about="&meannot;Rateable">
    <rdf:type rdf:resource="&owl;Class"/>
</rdf:Description>

<rdf:Description rdf:about="&meannot;Taggable">
  <rdf:type rdf:resource="&owl;Class"/>
</rdf:Description>

<rdf:Description rdf:about="&mebase;uri">
  <rdf:type rdf:resource="&owl;ObjectProperty"/>
</rdf:Description>

<rdf:Description rdf:about="&ore;proxyIn">
  <rdf:type rdf:resource="&owl;ObjectProperty"/>
</rdf:Description>

<rdf:Description rdf:about="&ore;proxyFor">
```

```
  <rdf:type rdf:resource="&owl;ObjectProperty"/>
</rdf:Description>

<rdf:Description rdf:about="&ore;Proxy">
  <rdf:type rdf:resource="&owl;Class"/>
</rdf:Description>

<rdf:Description rdf:about="&ore;isDescribedBy">
  <rdf:type rdf:resource="&owl;ObjectProperty"/>
</rdf:Description>


<!-- ================== Abstract Classes ================== -->

<owl:Class rdf:about="Entry">
  <rdfs:label>Entry</rdfs:label>
  <rdfs:comment>An entry into some aggregation</rdfs:comment>
  <rdfs:subClassOf rdf:resource="&mebase;Submission" />
  <rdfs:subClassOf rdf:resource="&ore;Proxy" />
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&dcterms;modified" />
      <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&dcterms;description" />
      <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1</
  owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&sioc;has_owner" />
      <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&ore;proxyIn" />
      <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&ore;proxyFor" />
      <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:isDefinedBy rdf:resource="&mepack;"/>
</owl:Class>

<owl:Class rdf:about="PackEntry">
  <rdfs:label>Entry</rdfs:label>
  <rdfs:comment>An entry in a Pack</rdfs:comment>
  <rdfs:subClassOf rdf:resource="Entry" />
  <rdfs:subClassOf>
    <owl:Restriction>
```

```
      <owl:onProperty rdf:resource="&ore;proxyIn" />
      <owl:allValuesFrom rdf:resource="Pack" />
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>


<!-- =============== myExperiment Entity Classes ================= -->

<owl:Class rdf:about="LocalPackEntry">
  <rdfs:label>LocalPackEntry</rdfs:label>
  <rdfs:comment>An entry in a Pack that is a Contribution</rdfs:comment>
  <owl:disjointWith rdf:resource="RemotePackEntry"/>
  <rdfs:subClassOf rdf:resource="PackEntry" />
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&ore;proxyFor" />
      <owl:allValuesFrom rdf:resource="&mebase;Contribution" />
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:isDefinedBy rdf:resource="&mepack;"/>
</owl:Class>

<owl:Class rdf:about="Pack">
  <rdfs:label>Pack</rdfs:label>
  <rdfs:comment>A Pack of Contributions/remote urls</rdfs:comment>
  <owl:disjointWith rdf:resource="&mebase;Upload"/>
  <rdfs:subClassOf rdf:resource="&mebase;Contribution" />
  <rdfs:subClassOf rdf:resource="&meannot;Commentable"/>
  <rdfs:subClassOf rdf:resource="&meannot;Taggable"/>
  <rdfs:subClassOf rdf:resource="&meannot;Rateable"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&ore;aggregates" />
      <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">0</
  owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&ore;isDescribedBy" />
      <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">0</
  owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:isDefinedBy rdf:resource="&mepack;"/>
</owl:Class>

<owl:Class rdf:about="RelationshipEntry">
  <rdfs:label>RelationshipEntry</rdfs:label>
  <rdfs:comment>A Relationship in the context of a particular Pack</rdfs:comment>
  <rdfs:subClassOf rdf:resource="Entry"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&ore;proxyFor" />
      <owl:allValuesFrom rdf:resource="Relationship" />
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:isDefinedBy rdf:resource="&mepack;"/>
```

```
  </owl:Class>

  <owl:Class rdf:about="Relationship">
    <rdfs:label>Relationship</rdfs:label>
    <rdfs:comment>A Relationship containing a subject, predicate and object.  A
    reified triple.</rdfs:comment>
    <rdfs:isDefinedBy rdf:resource="&mepack;"/>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="&rdf;subject" />
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="&rdf;predicate" />
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="&rdf;object" />
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>

  <owl:Class rdf:about="RemotePackEntry">
    <rdfs:label>RemotePackEntry</rdfs:label>
    <rdfs:comment>An entry in a Pack that is a remote url.</rdfs:comment>
    <owl:disjointWith rdf:resource="LocalPackEntry"/>
    <rdfs:subClassOf rdf:resource="PackEntry" />
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="&dcterms;title" />
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:isDefinedBy rdf:resource="&mepack;"/>
  </owl:Class>

<!-- =============== Object Properties ============== -->

  <owl:ObjectProperty rdf:about="has-entry">
    <rdfs:label>has-pack-entry</rdfs:label>
    <rdfs:comment>A pack may have zero or more pack entries.</rdfs:comment>
    <rdfs:domain rdf:resource="Pack"/>
    <rdfs:range rdf:resource="Entry"/>
    <owl:inverseOf rdf:resource="&ore;proxyIn"/>
    <rdfs:isDefinedBy rdf:resource="&mepack;"/>
  </owl:ObjectProperty>

<!-- =============== Datatype Properties ============== -->

</rdf:RDF>
```

Listing A.7: myExperiment's Packs Module

A.2.7 Experiments Ontology Module

`http://rdf.myexperiment.org/ontologies/experiments/` as of 16/10/2011.

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE rdf:RDF [
 <!ENTITY rdf  'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
 <!ENTITY rdfs 'http://www.w3.org/2000/01/rdf-schema#'>
 <!ENTITY owl 'http://www.w3.org/2002/07/owl#'>
 <!ENTITY xsd 'http://www.w3.org/2001/XMLSchema#'>
 <!ENTITY dc 'http://purl.org/dc/elements/1.1/'>
 <!ENTITY dcterms 'http://purl.org/dc/terms/'>
 <!ENTITY foaf 'http://xmlns.com/foaf/0.1/'>
 <!ENTITY ore 'http://www.openarchives.org/ore/terms/'>
 <!ENTITY sioc 'http://rdfs.org/sioc/ns#'>
 <!ENTITY mebase 'http://rdf.myexperiment.org/ontologies/base/'>
 <!ENTITY mepack 'http://rdf.myexperiment.org/ontologies/packs/'>
 <!ENTITY meexp 'http://rdf.myexperiment.org/ontologies/experiments/'>
]>

<rdf:RDF xml:base                ="&meexp;"
               xmlns                      ="&meexp;"
         xmlns:mebase   ="&mebase;"
               xmlns:mepack     ="&mepack;"
         xmlns:rdf              ="&rdf;"
         xmlns:rdfs             ="&rdfs;"
         xmlns:owl              ="&owl;"
         xmlns:dc                 ="&dc;"
               xmlns:dcterms="&dcterms;"
               xmlns:foaf               ="&foaf;"
               xmlns:ore                ="&ore;"
               xmlns:sioc               ="&sioc;"
>
  <!-- =============== Description ==================== -->

  <owl:Ontology rdf:about="&meexp;">
    <rdfs:label>myExperiment Experiments v1.0</rdfs:label>
    <rdfs:comment>This contains the classes required to create experiments and
    annotate them with jobs that have been or are scheduled to run.</rdfs:comment>
    <dc:language>en</dc:language>
    <dc:title xml:lang="en">The myExperiment Experiments Ontology</dc:title>
    <dc:creator rdf:resource="http://rdf.ecs.soton.ac.uk/person/9421"/>
    <dc:contributor rdf:datatype="http://www.w3.org/2001/XMLSchema#string">David R
    Newman</dc:contributor>
    <dc:publisher rdf:resource="http://rdf.myexperiment.org"/>
    <dc:date rdf:datatype="http://www.w3.org/2001/XMLSchema#date">2009-01-28</dc:date>
    <owl:versionInfo>$Date: 2010/05/20 $</owl:versionInfo>
    <dc:format rdf:datatype="http://www.w3.org/2001/XMLSchema#string">rdf/xml</
    dc:format>
  </owl:Ontology>

<!-- ============= Annotation Properties =========== -->

  <rdf:Description rdf:about="&dc;language">
    <rdf:type rdf:resource="&owl;AnnotationProperty"/>
  </rdf:Description>

  <rdf:Description rdf:about="&dc;title">
```

```
  <rdf:type rdf:resource="&owl;AnnotationProperty"/>
</rdf:Description>

<rdf:Description rdf:about="&dc;creator">
  <rdf:type rdf:resource="&owl;AnnotationProperty"/>
</rdf:Description>

<rdf:Description rdf:about="&dc;contributor">
  <rdf:type rdf:resource="&owl;AnnotationProperty"/>
</rdf:Description>

<rdf:Description rdf:about="&dc;publisher">
  <rdf:type rdf:resource="&owl;AnnotationProperty"/>
</rdf:Description>

<rdf:Description rdf:about="&dc;date">
  <rdf:type rdf:resource="&owl;AnnotationProperty"/>
</rdf:Description>

<rdf:Description rdf:about="&dc;format">
  <rdf:type rdf:resource="&owl;AnnotationProperty"/>
</rdf:Description>

<!-- ========= OWL-DL Compliance statements ======== -->

<rdf:Description rdf:about="&dcterms;title">
  <rdf:type rdf:resource="&owl;DatatypeProperty"/>
  <rdfs:range rdf:resource="&xsd;string"/>
</rdf:Description>

<rdf:Description rdf:about="&mebase;Interface">
  <rdf:type rdf:resource="&owl;Class"/>
</rdf:Description>

<rdf:Description rdf:about="&mebase;Submission">
  <rdf:type rdf:resource="&owl;Class"/>
</rdf:Description>

<rdf:Description rdf:about="&mebase;Annotation">
  <rdf:type rdf:resource="&owl;Class"/>
</rdf:Description>

<rdf:Description rdf:about="&mebase;Contribution">
  <rdf:type rdf:resource="&owl;Class"/>
</rdf:Description>

<rdf:Description rdf:about="&mebase;User">
  <rdf:type rdf:resource="&owl;Class"/>
</rdf:Description>

<rdf:Description rdf:about="&mebase;username">
  <rdf:type rdf:resource="&owl;DatatypeProperty"/>
</rdf:Description>

<rdf:Description rdf:about="&mebase;uri">
  <rdf:type rdf:resource="&owl;ObjectProperty"/>
</rdf:Description>

<rdf:Description rdf:about="&mebase;text">
```

```
  <rdf:type rdf:resource="&owl;DatatypeProperty"/>
</rdf:Description>

<rdf:Description rdf:about="&mepack;Pack">
  <rdf:type rdf:resource="&owl;Class"/>
</rdf:Description>

<rdf:Description rdf:about="&mepack;PackEntry">
  <rdf:type rdf:resource="&owl;Class"/>
</rdf:Description>

<rdf:Description rdf:about="&sioc;has_owner">
  <rdf:type rdf:resource="&owl;ObjectProperty"/>
</rdf:Description>

<rdf:Description rdf:about="&dcterms;modified">
  <rdf:type rdf:resource="&owl;DatatypeProperty"/>
  <rdfs:range rdf:resource="&xsd;dateTime"/>
</rdf:Description>

<!-- ====================== Interfaces ====================== -->

<owl:Class rdf:about="Runnable">
  <rdfs:label>Runnable</rdfs:label>
  <rdfs:comment>An object that can be run by a Runner</rdfs:comment>
  <rdfs:subClassOf rdf:resource="&mebase;Interface"/>
  <rdfs:isDefinedBy rdf:resource="&meexp;"/>
</owl:Class>

<!-- =============== myExperiment Entity Classes ================= -->

<owl:Class rdf:about="Data">
  <rdfs:label>Data</rdfs:label>
  <rdfs:comment>Input to or output from a Job</rdfs:comment>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&mebase;text" />
      <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1</
  owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
   <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&mebase;uri" />
      <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1</
  owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:isDefinedBy rdf:resource="&meexp;"/>
</owl:Class>

<owl:Class rdf:about="Experiment">
  <rdfs:label>Experiment</rdfs:label>
  <rdfs:comment>An Experiment is a container for experimentation</rdfs:comment>
  <rdfs:subClassOf rdf:resource="&mepack;Pack"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&dcterms;title" />
      <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
```

```
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:isDefinedBy rdf:resource="&meexp;"/>
</owl:Class>

<owl:Class rdf:about="Job">
  <rdfs:label>Job</rdfs:label>
  <rdfs:comment>An enactment of a Workflow as part of an Experiment</rdfs:comment>
  <rdfs:subClassOf rdf:resource="&mebase;Contribution"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="has-runnable" />
      <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
   <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="has-runner" />
      <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&dcterms;title" />
      <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&mebase;uri" />
      <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1</
  owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:isDefinedBy rdf:resource="&meexp;"/>
</owl:Class>

<owl:Class rdf:about="Runner">
  <rdfs:label>Runner</rdfs:label>
  <rdfs:comment>A Job requires a Runner to run</rdfs:comment>
  <owl:disjointWith rdf:resource="&mepack;Pack"/>
  <rdfs:subClassOf rdf:resource="&mebase;Contribution"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="runner-url" />
      <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&mebase;username" />
      <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1</
  owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:isDefinedBy rdf:resource="&meexp;"/>
</owl:Class>

<!-- =============== Object Properties ============== -->
```

```
  <owl:ObjectProperty rdf:about="has-runner">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:label>has-runner</rdfs:label>
    <rdfs:comment>To run a Job, a Job must have a runner</rdfs:comment>
    <rdfs:domain rdf:resource="Job"/>
    <rdfs:range rdf:resource="Runner"/>
    <rdfs:isDefinedBy rdf:resource="&meexp;"/>
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:about="has-parent-job">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:label>has-parent-job</rdfs:label>
    <rdfs:comment>A Job may have a parent job</rdfs:comment>
    <rdfs:domain rdf:resource="Job"/>
    <rdfs:range rdf:resource="Job"/>
    <rdfs:isDefinedBy rdf:resource="&meexp;"/>
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:about="has-runnable">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:label>has-runnable</rdfs:label>
    <rdfs:comment>A Job has a Runnable object</rdfs:comment>
    <rdfs:domain rdf:resource="Job"/>
    <rdfs:range rdf:resource="Runnable"/>
    <rdfs:isDefinedBy rdf:resource="&meexp;"/>
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:about="has-input">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:label>has-input</rdfs:label>
    <rdfs:comment>A Job may have some Data as input</rdfs:comment>
    <rdfs:domain rdf:resource="Job"/>
    <rdfs:range rdf:resource="Data"/>
    <rdfs:isDefinedBy rdf:resource="&meexp;"/>
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:about="has-output">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:label>has-output</rdfs:label>
    <rdfs:comment>A Job may have some Data as input</rdfs:comment>
    <rdfs:domain rdf:resource="Job"/>
    <rdfs:range rdf:resource="Data"/>
    <rdfs:isDefinedBy rdf:resource="&meexp;"/>
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:about="runner-url">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:label>runner-url</rdfs:label>
    <rdfs:comment>A Runner must have an runner-url from where to invoke the runner</
    rdfs:comment>
    <rdfs:domain rdf:resource="Runner"/>
    <rdfs:isDefinedBy rdf:resource="&meexp;"/>
  </owl:ObjectProperty>


<!-- =============== Datatype Properties ============== -->

  <owl:DatatypeProperty rdf:about="completed-at">
```

```
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:label>completed-at</rdfs:label>
    <rdfs:comment>A Job is completed-at a particular dateTime to a Runner</
    rdfs:comment>
    <rdfs:domain rdf:resource="Job"/>
    <rdfs:range rdf:resource="&xsd;dateTime"/>
    <rdfs:isDefinedBy rdf:resource="&meexp;"/>
  </owl:DatatypeProperty>

  <owl:DatatypeProperty rdf:about="job-manifest">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:label>job-manifest</rdfs:label>
    <rdfs:comment>The Job's manifest</rdfs:comment>
    <rdfs:domain rdf:resource="Job"/>
    <rdfs:range rdf:resource="&xsd;string"/>
    <rdfs:isDefinedBy rdf:resource="&meexp;"/>
  </owl:DatatypeProperty>

  <owl:DatatypeProperty rdf:about="last-status">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:label>last-status</rdfs:label>
    <rdfs:comment>The last-status of the Job running in the Runner</rdfs:comment>
    <rdfs:domain rdf:resource="Job"/>
    <rdfs:range rdf:resource="&xsd;string"/>
    <rdfs:isDefinedBy rdf:resource="&meexp;"/>
  </owl:DatatypeProperty>

  <owl:DatatypeProperty rdf:about="last-status-at">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:label>last-status-at</rdfs:label>
    <rdfs:comment>The last dateTime when the last-status of the Job was polled</
    rdfs:comment>
    <rdfs:domain rdf:resource="Job"/>
    <rdfs:range rdf:resource="&xsd;dateTime"/>
    <rdfs:isDefinedBy rdf:resource="&meexp;"/>
  </owl:DatatypeProperty>

  <owl:DatatypeProperty rdf:about="started-at">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:label>started-at</rdfs:label>
    <rdfs:comment>A Job is started-at a particular dateTime in a Runner</rdfs:comment>
    <rdfs:domain rdf:resource="Job"/>
    <rdfs:range rdf:resource="&xsd;dateTime"/>
    <rdfs:isDefinedBy rdf:resource="&meexp;"/>
  </owl:DatatypeProperty>

  <owl:DatatypeProperty rdf:about="submitted-at">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:label>submitted-at</rdfs:label>
    <rdfs:comment>A Job is submitted-at a particular dateTime to a Runner</
    rdfs:comment>
    <rdfs:domain rdf:resource="Job"/>
    <rdfs:range rdf:resource="&xsd;dateTime"/>
    <rdfs:isDefinedBy rdf:resource="&meexp;"/>
  </owl:DatatypeProperty>

</rdf:RDF>
```

Listing A.8: myExperiment's Experiments Module

A.2.8 Viewings and Downloads Ontology Module

http://rdf.myexperiment.org/ontologies/viewings_downloads/ as of 16/10/2011.

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE rdf:RDF [
 <!ENTITY rdf  'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
 <!ENTITY rdfs 'http://www.w3.org/2000/01/rdf-schema#'>
 <!ENTITY owl 'http://www.w3.org/2002/07/owl#'>
 <!ENTITY xsd 'http://www.w3.org/2001/XMLSchema#'>
 <!ENTITY dc 'http://purl.org/dc/elements/1.1/'>
 <!ENTITY mebase 'http://rdf.myexperiment.org/ontologies/base/'>
 <!ENTITY mevd 'http://rdf.myexperiment.org/ontologies/viewings_downloads/'>
]>

<rdf:RDF xml:base               ="&mevd;"
         xmlns                  ="&mevd;"
         xmlns:mebase="&mebase;"
         xmlns:rdf       ="&rdf;"
         xmlns:rdfs      ="&rdfs;"
         xmlns:owl       ="&owl;"
         xmlns:dc                ="&dc;"
         xmlns:xsd       ="&xsd;"
>

  <!-- =============== Description ==================== -->

  <owl:Ontology rdf:about="&mevd;">
    <rdfs:label>myExperiment Viewings &amp; Downloads v1.0</rdfs:label>
    <rdfs:comment>This allows statistics on the viewings and downloads of
    contributions to be recorded.</rdfs:comment>
    <dc:language>en</dc:language>
    <dc:title xml:lang="en">The myExperiment Viewings &amp; Downloads Ontology</
    dc:title>
    <dc:creator rdf:resource="http://rdf.ecs.soton.ac.uk/person/9421"/>
    <dc:contributor rdf:datatype="http://www.w3.org/2001/XMLSchema#string">David R
    Newman</dc:contributor>
    <dc:publisher rdf:resource="http://rdf.myexperiment.org"/>
    <dc:date rdf:datatype="http://www.w3.org/2001/XMLSchema#date">2009-01-28</dc:date>
    <owl:versionInfo>$Date: 2009/05/06 $</owl:versionInfo>
    <dc:format rdf:datatype="http://www.w3.org/2001/XMLSchema#string">rdf/xml</
    dc:format>
  </owl:Ontology>

<!-- ============= Annotation Properties =========== -->

  <rdf:Description rdf:about="&dc;language">
    <rdf:type rdf:resource="&owl;AnnotationProperty"/>
  </rdf:Description>

  <rdf:Description rdf:about="&dc;title">
    <rdf:type rdf:resource="&owl;AnnotationProperty"/>
  </rdf:Description>

  <rdf:Description rdf:about="&dc;creator">
    <rdf:type rdf:resource="&owl;AnnotationProperty"/>
  </rdf:Description>
```

```
<rdf:Description rdf:about="&dc;contributor">
  <rdf:type rdf:resource="&owl;AnnotationProperty"/>
</rdf:Description>

<rdf:Description rdf:about="&dc;publisher">
  <rdf:type rdf:resource="&owl;AnnotationProperty"/>
</rdf:Description>

<rdf:Description rdf:about="&dc;date">
  <rdf:type rdf:resource="&owl;AnnotationProperty"/>
</rdf:Description>

<rdf:Description rdf:about="&dc;format">
  <rdf:type rdf:resource="&owl;AnnotationProperty"/>
</rdf:Description>

<!-- ========= OWL-DL Compliance statements ======== -->

<rdf:Description rdf:about="&mebase;Annotation">
  <rdf:type rdf:resource="&owl;Class"/>
</rdf:Description>

<rdf:Description rdf:about="&mebase;count">
  <rdf:type rdf:resource="&owl;DatatypeProperty"/>
</rdf:Description>

<rdf:Description rdf:about="&mebase;User">
  <rdf:type rdf:resource="&owl;Class"/>
</rdf:Description>

<rdf:Description rdf:about="&mebase;Contribution">
  <rdf:type rdf:resource="&owl;Class"/>
</rdf:Description>


<!-- =============== myExperiment Entity Classes ================= -->

<owl:Class rdf:about="Usage">
  <rdfs:label>Usage</rdfs:label>
  <rdfs:comment>A Usage of a Contribution, i.e. Viewing, Download, etc.</
  rdfs:comment>
  <rdfs:subClassOf rdf:resource="&mebase;Annotation"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="user-agent" />
      <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:isDefinedBy rdf:resource="&mevd;"/>
</owl:Class>

<owl:Class rdf:about="Download">
  <rdfs:label>Download</rdfs:label>
  <rdfs:comment>A Download of a Contribution by a User</rdfs:comment>
  <rdfs:subClassOf rdf:resource="Usage"/>
  <rdfs:isDefinedBy rdf:resource="&mevd;"/>
</owl:Class>

<owl:Class rdf:about="Downloads">
```

```
    <rdfs:label>Downloads</rdfs:label>
    <rdfs:comment>Downloads of a Contribution by a User</rdfs:comment>
    <rdfs:subClassOf rdf:resource="Download"/>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="&mebase;count" />
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:isDefinedBy rdf:resource="&mevd;"/>
  </owl:Class>

  <owl:Class rdf:about="Viewing">
    <rdfs:label>Viewing</rdfs:label>
    <rdfs:comment>A Viewing of a Contribution by a User</rdfs:comment>
    <rdfs:subClassOf rdf:resource="Usage"/>
    <rdfs:isDefinedBy rdf:resource="&mevd;"/>
  </owl:Class>

  <owl:Class rdf:about="Viewings">
    <rdfs:label>Viewings</rdfs:label>
    <rdfs:comment>Viewings of a Contribution by a User</rdfs:comment>
    <rdfs:subClassOf rdf:resource="Viewing"/>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="&mebase;count" />
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:isDefinedBy rdf:resource="&mevd;"/>
  </owl:Class>

<!-- =============== Datatype Properties ============== -->

  <owl:DatatypeProperty rdf:about="downloads-count">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:label>downloads-count</rdfs:label>
    <rdfs:comment>The count of contribution downloads by a User</rdfs:comment>
    <rdfs:domain rdf:resource="&mebase;User"/>
    <rdfs:range rdf:resource="&xsd;nonNegativeInteger"/>
    <rdfs:isDefinedBy rdf:resource="&mevd;"/>
  </owl:DatatypeProperty>

  <owl:DatatypeProperty rdf:about="viewings-count">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:label>viewings-count</rdfs:label>
    <rdfs:comment>The count of Contribution viewings by a User</rdfs:comment>
    <rdfs:domain rdf:resource="&mebase;User"/>
    <rdfs:range rdf:resource="&xsd;nonNegativeInteger"/>
    <rdfs:isDefinedBy rdf:resource="&mevd;"/>
  </owl:DatatypeProperty>

  <owl:DatatypeProperty rdf:about="downloaded">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:label>downloaded</rdfs:label>
    <rdfs:comment>The count of the number of times a Contribution has been downloaded<
    /rdfs:comment>
    <rdfs:domain rdf:resource="&mebase;Contribution"/>
    <rdfs:range rdf:resource="&xsd;nonNegativeInteger"/>
```

```
    <rdfs:isDefinedBy rdf:resource="&mevd;"/>
  </owl:DatatypeProperty>

  <owl:DatatypeProperty rdf:about="viewed">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:label>viewed</rdfs:label>
    <rdfs:comment>The count of the number of times a Contribution has been viewed</
    rdfs:comment>
    <rdfs:domain rdf:resource="&mebase;Contribution"/>
    <rdfs:range rdf:resource="&xsd;nonNegativeInteger"/>
    <rdfs:isDefinedBy rdf:resource="&mevd;"/>
  </owl:DatatypeProperty>

  <owl:DatatypeProperty rdf:about="user-agent">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:label>user-agent</rdfs:label>
    <rdfs:comment>The user-agent used to view/download</rdfs:comment>
    <rdfs:domain rdf:resource="Usage"/>
    <rdfs:range rdf:resource="&xsd;string"/>
    <rdfs:isDefinedBy rdf:resource="&mevd;"/>
  </owl:DatatypeProperty>

</rdf:RDF>
```

Listing A.9: myExperiment's Viewings and Downloads Module

A.2.9 Workflow Components Ontology Module

http://rdf.myexperiment.org/ontologies/components/ as of 16/10/2011.

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE rdf:RDF [
 <!ENTITY rdf  'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
 <!ENTITY rdfs 'http://www.w3.org/2000/01/rdf-schema#'>
 <!ENTITY owl 'http://www.w3.org/2002/07/owl#'>
 <!ENTITY xsd 'http://www.w3.org/2001/XMLSchema#'>
 <!ENTITY dc 'http://purl.org/dc/elements/1.1/'>
 <!ENTITY dcterms 'http://purl.org/dc/terms/'>
 <!ENTITY mebase 'http://rdf.myexperiment.org/ontologies/base/'>
 <!ENTITY mecontrib 'http://rdf.myexperiment.org/ontologies/contributions/'>
 <!ENTITY mecomp 'http://rdf.myexperiment.org/ontologies/components/'>
]>

<rdf:RDF xml:base       ="&mecomp;"
         xmlns          ="&mecomp;"
         xmlns:mecontrib="&mecontrib;"
         xmlns:mebase   ="&mebase;"
         xmlns:rdf      ="&rdf;"
         xmlns:rdfs     ="&rdfs;"
         xmlns:owl      ="&owl;"
         xmlns:dc       ="&dc;"
         xmlns:dcterms  ="&dcterms;"
         xmlns:xsd      ="&xsd;"
>

  <owl:Ontology rdf:about="&mecomp;">
```

```
    <rdfs:label>myExperiment Components v1.0</rdfs:label>
    <rdfs:comment>This provides classes for representing the components of a workflow<
    /rdfs:comment>
    <dc:language>en</dc:language>
    <dc:title xml:lang="en">The myExperiment Components Ontology</dc:title>
    <dc:creator rdf:resource="http://rdf.ecs.soton.ac.uk/person/9421"/>
    <dc:contributor rdf:datatype="http://www.w3.org/2001/XMLSchema#string">David R
    Newman</dc:contributor>
    <dc:publisher rdf:resource="http://rdf.myexperiment.org"/>
    <dc:date rdf:datatype="http://www.w3.org/2001/XMLSchema#date">2009-01-28</dc:date>
    <owl:versionInfo>$Date: 2010/05/20 $</owl:versionInfo>
    <dc:format rdf:datatype="http://www.w3.org/2001/XMLSchema#string">rdf/xml</
    dc:format>
  </owl:Ontology>

<!-- ============= Annotation Properties =========== -->

  <rdf:Description rdf:about="&dc;language">
    <rdf:type rdf:resource="&owl;AnnotationProperty"/>
  </rdf:Description>

  <rdf:Description rdf:about="&dc;title">
    <rdf:type rdf:resource="&owl;AnnotationProperty"/>
  </rdf:Description>

  <rdf:Description rdf:about="&dc;creator">
    <rdf:type rdf:resource="&owl;AnnotationProperty"/>
  </rdf:Description>

  <rdf:Description rdf:about="&dc;contributor">
    <rdf:type rdf:resource="&owl;AnnotationProperty"/>
  </rdf:Description>

  <rdf:Description rdf:about="&dc;publisher">
    <rdf:type rdf:resource="&owl;AnnotationProperty"/>
  </rdf:Description>

  <rdf:Description rdf:about="&dc;date">
    <rdf:type rdf:resource="&owl;AnnotationProperty"/>
  </rdf:Description>

  <rdf:Description rdf:about="&dc;format">
    <rdf:type rdf:resource="&owl;AnnotationProperty"/>
  </rdf:Description>

<!-- ========= OWL-DL Compliance statements ======== -->

  <rdf:Description rdf:about="&dcterms;title">
    <rdf:type rdf:resource="&owl;DatatypeProperty"/>
    <rdfs:range rdf:resource="&xsd;string"/>
  </rdf:Description>

  <rdf:Description rdf:about="&dcterms;description">
    <rdf:type rdf:resource="&owl;DatatypeProperty"/>
    <rdfs:range rdf:resource="&xsd;string"/>
  </rdf:Description>

  <rdf:Description rdf:about="&dcterms;identifier">
    <rdf:type rdf:resource="&owl;DatatypeProperty"/>
```

```
    <rdfs:range rdf:resource="&xsd;string"/>
  </rdf:Description>

  <rdf:Description rdf:about="&mecontrib;AbstractWorkflow">
    <rdf:type rdf:resource="&owl;Class"/>
  </rdf:Description>


<!-- ================== Classes ==================== -->

  <owl:Class rdf:about="WorkflowComponent">
    <rdfs:label>WorkflowComponent</rdfs:label>
    <rdfs:comment>A component of a Workflow (e.g. a Sink, Source, Processor or Link)</
    rdfs:comment>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="&dcterms;title"/>
        <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#
    nonNegativeInteger">1</owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="belongs-to-workflow"/>
        <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#
    nonNegativeInteger">1</owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:isDefinedBy rdf:resource="&mecomp;"/>
  </owl:Class>

  <owl:Class rdf:about="NodeComponent">
    <rdfs:label>NodeComponent</rdfs:label>
    <rdfs:comment>A component of a Workflow that is not a Link or IOComponent (i.e. a
    Sink, Source or Processor)</rdfs:comment>
    <rdfs:subClassOf rdf:resource="WorkflowComponent"/>
    <owl:disjointWith rdf:resource="IOComponent"/>
    <owl:disjointWith rdf:resource="Link"/>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="&dcterms;title"/>
        <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#
    nonNegativeInteger">1</owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:isDefinedBy rdf:resource="&mecomp;"/>
  </owl:Class>

  <owl:Class rdf:about="IOComponent">
    <rdfs:label>IOComponent</rdfs:label>
    <rdfs:comment>An Input or Output to a NodeComponent (e.g. a Sink, Source or
    Processor)</rdfs:comment>
    <rdfs:subClassOf rdf:resource="WorkflowComponent"/>
    <owl:disjointWith rdf:resource="NodeComponent"/>
    <owl:disjointWith rdf:resource="Link"/>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="&dcterms;title"/>
```

```
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#
  nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="for-component"/>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#
  nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:isDefinedBy rdf:resource="&mecomp;"/>
</owl:Class>

<owl:Class rdf:about="Link">
  <rdfs:label>Link</rdfs:label>
  <rdfs:comment>A component of a Workflow that links a Source to a Sink (Assuming
  the Link isn't an initial input or a final output)</rdfs:comment>
  <rdfs:subClassOf rdf:resource="WorkflowComponent"/>
  <owl:disjointWith rdf:resource="NodeComponent"/>
  <owl:disjointWith rdf:resource="IOComponent"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="to-input"/>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#
  nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="from-output"/>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#
  nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:isDefinedBy rdf:resource="&mecomp;"/>
</owl:Class>

<owl:Class rdf:about="Processor">
  <rdfs:label>Processor</rdfs:label>
  <rdfs:comment>A component of a Workflow that processes some data</rdfs:comment>
  <rdfs:subClassOf rdf:resource="NodeComponent"/>
  <owl:disjointWith rdf:resource="Source"/>
  <owl:disjointWith rdf:resource="Sink"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&dcterms;description"/>
      <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#
  nonNegativeInteger">1</owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:isDefinedBy rdf:resource="&mecomp;"/>
</owl:Class>

<owl:Class rdf:about="WSDLProcessor">
  <rdfs:label>WSDLProcessor</rdfs:label>
  <rdfs:comment>A Processor that executes some WSDL</rdfs:comment>
  <rdfs:subClassOf rdf:resource="Processor"/>
  <owl:disjointWith rdf:resource="BeanshellProcessor"/>
```

```
  <owl:disjointWith rdf:resource="DataflowProcessor"/>
  <owl:disjointWith rdf:resource="ConstantProcessor"/>
  <owl:disjointWith rdf:resource="OtherProcessor"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="processor-uri"/>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#
  nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="service-name"/>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#
  nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:isDefinedBy rdf:resource="&mecomp;"/>
</owl:Class>

<owl:Class rdf:about="BeanshellProcessor">
  <rdfs:label>BeanshellProcessor</rdfs:label>
  <rdfs:comment>A Processor that executes a Beanshell</rdfs:comment>
  <rdfs:subClassOf rdf:resource="Processor"/>
  <owl:disjointWith rdf:resource="WSDLProcessor"/>
  <owl:disjointWith rdf:resource="DataflowProcessor"/>
  <owl:disjointWith rdf:resource="ConstantProcessor"/>
  <owl:disjointWith rdf:resource="OtherProcessor"/>
   <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="processor-script"/>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#
  nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:isDefinedBy rdf:resource="&mecomp;"/>
</owl:Class>

<owl:Class rdf:about="DataflowProcessor">
  <rdfs:label>DataflowProcessor</rdfs:label>
  <rdfs:comment>A Processor that executes a Dataflow</rdfs:comment>
  <rdfs:subClassOf rdf:resource="Processor"/>
  <owl:disjointWith rdf:resource="BeanshellProcessor"/>
  <owl:disjointWith rdf:resource="WSDLProcessor"/>
  <owl:disjointWith rdf:resource="ConstantProcessor"/>
  <owl:disjointWith rdf:resource="OtherProcessor"/>
   <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="executes-dataflow"/>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#
  nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:isDefinedBy rdf:resource="&mecomp;"/>
</owl:Class>

<owl:Class rdf:about="ConstantProcessor">
  <rdfs:label>ConstantProcessor</rdfs:label>
```

```
  <rdfs:comment>A Processor that performs the same process each time.  (E.g. a
  stringconstant processor just echoes the same string each time).</rdfs:comment>
  <rdfs:subClassOf rdf:resource="Processor"/>
  <owl:disjointWith rdf:resource="BeanshellProcessor"/>
  <owl:disjointWith rdf:resource="WSDLProcessor"/>
  <owl:disjointWith rdf:resource="DataflowProcessor"/>
  <owl:disjointWith rdf:resource="OtherProcessor"/>
   <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="processor-value"/>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#
  nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:isDefinedBy rdf:resource="&mecomp;"/>
</owl:Class>

<owl:Class rdf:about="OtherProcessor">
  <rdfs:label>OtherProcessor</rdfs:label>
  <rdfs:comment>A Processor that executes something else</rdfs:comment>
  <rdfs:subClassOf rdf:resource="Processor"/>
  <owl:disjointWith rdf:resource="WSDLProcessor"/>
  <owl:disjointWith rdf:resource="DataflowProcessor"/>
  <owl:disjointWith rdf:resource="BeanshellProcessor"/>
  <owl:disjointWith rdf:resource="ConstantProcessor"/>
  <rdfs:isDefinedBy rdf:resource="&mecomp;"/>
</owl:Class>

<owl:Class rdf:about="Sink">
  <rdfs:label>Sink</rdfs:label>
  <rdfs:comment>A component of a Workflow that is the sink for the data being output
  </rdfs:comment>
  <rdfs:subClassOf rdf:resource="NodeComponent"/>
  <owl:disjointWith rdf:resource="Source"/>
  <owl:disjointWith rdf:resource="Processor"/>
  <rdfs:isDefinedBy rdf:resource="&mecomp;"/>
</owl:Class>

<owl:Class rdf:about="Source">
  <rdfs:label>Source</rdfs:label>
  <rdfs:comment>A component of a Workflow that is the source of data being input</
  rdfs:comment>
  <rdfs:subClassOf rdf:resource="NodeComponent"/>
  <owl:disjointWith rdf:resource="Source"/>
  <owl:disjointWith rdf:resource="Processor"/>
  <rdfs:isDefinedBy rdf:resource="&mecomp;"/>
</owl:Class>

<owl:Class rdf:about="Input">
  <rdfs:label>Input</rdfs:label>
  <rdfs:comment>A Link must have an Input into a NodeComponent (i.e. Source, Sink or
   Processor)</rdfs:comment>
  <rdfs:subClassOf rdf:resource="IOComponent"/>
  <owl:disjointWith rdf:resource="Output"/>
  <rdfs:isDefinedBy rdf:resource="&mecomp;"/>
</owl:Class>

<owl:Class rdf:about="Output">
  <rdfs:label>Output</rdfs:label>
```

```
    <rdfs:comment>A Link must have an Output from a NodeComponent (i.e. Source, Sink
    or Processor)</rdfs:comment>
    <rdfs:subClassOf rdf:resource="IOComponent"/>
    <owl:disjointWith rdf:resource="Input"/>
    <rdfs:isDefinedBy rdf:resource="&mecomp;"/>
  </owl:Class>

  <owl:Class rdf:about="Dataflow">
    <rdfs:label>Dataflow</rdfs:label>
    <rdfs:comment>A Dataflow is what is executed by an AbretactWorkflow subclass or a
    DataflowProcessor</rdfs:comment>
    <owl:disjointWith rdf:resource="WorkflowComponent"/>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="&dcterms;identifier"/>
        <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#
    nonNegativeInteger">1</owl:maxCardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:isDefinedBy rdf:resource="&mecomp;"/>
  </owl:Class>



<!-- ======================= Object Properties ======================== -->

  <owl:ObjectProperty rdf:about="belongs-to-workflow">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
    <rdfs:label>belongs-to-workflow</rdfs:label>
    <rdfs:comment>A WorkflowComponent belongs to a particular Workflow</rdfs:comment>
    <rdfs:domain rdf:resource="WorkflowComponent"/>
    <rdfs:range rdf:resource="&mecontrib;AbstractWorkflow"/>
    <rdfs:isDefinedBy rdf:resource="&mecomp;"/>
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:about="has-component">
    <rdfs:label>has-component</rdfs:label>
    <rdfs:comment>A Dataflow may have WorkflowComponents</rdfs:comment>
    <rdfs:domain rdf:resource="Dataflow"/>
    <rdfs:range rdf:resource="WorkflowComponent"/>
    <rdfs:isDefinedBy rdf:resource="&mecomp;"/>
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:about="to-input">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
    <rdfs:label>to-input</rdfs:label>
    <rdfs:comment>A Link WorkflowComponent will go to a Input</rdfs:comment>
    <rdfs:domain rdf:resource="Link"/>
    <rdfs:range rdf:resource="Input"/>
    <rdfs:isDefinedBy rdf:resource="&mecomp;"/>
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:about="from-output">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
    <rdfs:label>from-output</rdfs:label>
    <rdfs:comment>A Link WorkflowComponent will come from an Output</rdfs:comment>
    <rdfs:domain rdf:resource="Link"/>
    <rdfs:range rdf:resource="Output"/>
    <rdfs:isDefinedBy rdf:resource="&mecomp;"/>
```

```
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:about="for-component">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
    <rdfs:label>for-component</rdfs:label>
    <rdfs:comment>An IOComponent will input to or output from a NodeComponent (i.e.
    Sink, Source or Processor)</rdfs:comment>
    <rdfs:domain rdf:resource="IOComponent"/>
    <rdfs:range rdf:resource="NodeComponent"/>
    <rdfs:isDefinedBy rdf:resource="&mecomp;"/>
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:about="executes-dataflow">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
    <rdfs:label>executes-dataflow</rdfs:label>
    <rdfs:comment>An AbstractWorkflow subclass or a DataflowProcessor executes a
    Dataflow</rdfs:comment>
    <rdfs:domain>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="&mecontrib;AbstractWorkflow"/>
          <owl:Class rdf:about="DataflowProcessor"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:domain>
    <rdfs:range rdf:resource="Dataflow"/>
    <rdfs:isDefinedBy rdf:resource="&mecomp;"/>
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:about="processor-uri">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
    <rdfs:label>processor-uri</rdfs:label>
    <rdfs:comment>A Processor may have an URI where the service resides</rdfs:comment>
    <rdfs:domain rdf:resource="Processor"/>
    <rdfs:isDefinedBy rdf:resource="&mecomp;"/>
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:about="waits-on">
    <rdfs:label>waits-on</rdfs:label>
    <rdfs:comment>A Processor may have to wait on one or more processor to complete
    before it can execute</rdfs:comment>
    <rdfs:domain rdf:resource="Processor"/>
    <rdfs:range rdf:resource="Processor"/>
    <rdfs:isDefinedBy rdf:resource="&mecomp;"/>
  </owl:ObjectProperty>


<!-- ====================== Datatype Properties ======================== -->

  <owl:DatatypeProperty rdf:about="processor-type">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
    <rdfs:label>processor-type</rdfs:label>
    <rdfs:comment>A Processor must have a type property if it is not a specific
    Processor class</rdfs:comment>
    <rdfs:domain rdf:resource="Processor"/>
    <rdfs:range rdf:resource="&xsd;string"/>
    <rdfs:isDefinedBy rdf:resource="&mecomp;"/>
  </owl:DatatypeProperty>
```

```
<owl:DatatypeProperty rdf:about="processor-script">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdfs:label>processor-script</rdfs:label>
  <rdfs:comment>A Processor may have a script that it executes</rdfs:comment>
  <rdfs:domain rdf:resource="Processor"/>
  <rdfs:range rdf:resource="&xsd;string"/>
  <rdfs:isDefinedBy rdf:resource="&mecomp;"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:about="processor-value">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdfs:label>processor-value</rdfs:label>
  <rdfs:comment>A Processor may have a value that it represents</rdfs:comment>
  <rdfs:domain rdf:resource="ConstantProcessor"/>
  <rdfs:range rdf:resource="&xsd;string"/>
  <rdfs:isDefinedBy rdf:resource="&mecomp;"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:about="service-name">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdfs:label>service-name</rdfs:label>
  <rdfs:comment>A Processor may have a name for the service it executes</
  rdfs:comment>
  <rdfs:domain rdf:resource="Processor"/>
  <rdfs:range rdf:resource="&xsd;string"/>
  <rdfs:isDefinedBy rdf:resource="&mecomp;"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:about="service-category">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdfs:label>service-category</rdfs:label>
  <rdfs:comment>The service a Processor executes may have a category</rdfs:comment>
  <rdfs:domain rdf:resource="Processor"/>
  <rdfs:range rdf:resource="&xsd;string"/>
  <rdfs:isDefinedBy rdf:resource="&mecomp;"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:about="authority-name">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdfs:label>authority-name</rdfs:label>
  <rdfs:comment>A Processor may have the name of an authority that is responsible
  for the service it executes</rdfs:comment>
  <rdfs:domain rdf:resource="Processor"/>
  <rdfs:range rdf:resource="&xsd;string"/>
  <rdfs:isDefinedBy rdf:resource="&mecomp;"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:about="example-value">
  <rdfs:label>example-value</rdfs:label>
  <rdfs:comment>A Sink or Source may have one or more example values</rdfs:comment>
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="Sink"/>
        <owl:Class rdf:about="Source"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
  <rdfs:range rdf:resource="&xsd;string"/>
```

```
    <rdfs:isDefinedBy rdf:resource="&mecomp;"/>
  </owl:DatatypeProperty>

</rdf:RDF>
```

LISTING A.10: myExperiment's Workflow Components Module

A.2.10 Specific Ontology Module

`http://rdf.myexperiment.org/ontologies/specific/` as of 16/10/2011.

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE rdf:RDF [
 <!ENTITY rdf  'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
 <!ENTITY rdfs 'http://www.w3.org/2000/01/rdf-schema#'>
 <!ENTITY owl 'http://www.w3.org/2002/07/owl#'>
 <!ENTITY xsd 'http://www.w3.org/2001/XMLSchema#'>
 <!ENTITY dc 'http://purl.org/dc/elements/1.1/'>
 <!ENTITY dcterms 'http://purl.org/dc/terms/'>
 <!ENTITY cc  'http://web.resource.org/cc/'>
 <!ENTITY snarm 'http://rdf.myexperiment.org/ontologies/snarm/'>

 <!ENTITY sioc 'http://rdfs.org/sioc/ns#'>
 <!ENTITY mebase 'http://rdf.myexperiment.org/ontologies/base/'>
 <!ENTITY meac 'http://rdf.myexperiment.org/ontologies/attrib_credit/'>
 <!ENTITY meannot 'http://rdf.myexperiment.org/ontologies/annotations/'>
 <!ENTITY meexp 'http://rdf.myexperiment.org/ontologies/experiments/'>
 <!ENTITY mepack 'http://rdf.myexperiment.org/ontologies/packs/'>
 <!ENTITY mecontrib 'http://rdf.myexperiment.org/ontologies/contributions/'>
 <!ENTITY mevd 'http://rdf.myexperiment.org/ontologies/viewings_downloads/'>
 <!ENTITY mecomp 'http://rdf.myexperiment.org/ontologies/components/'>
 <!ENTITY mespec 'http://rdf.myexperiment.org/ontologies/specific/'>
]>

<rdf:RDF xml:base       ="&mespec;"
         xmlns          ="&mespec;"
         xmlns:mebase   ="&mebase;"
               xmlns:meac     ="&meac;"
               xmlns:meannot  ="&meannot;"
               xmlns:meexp    ="&meexp;"
         xmlns:mepack   ="&mepack;"
         xmlns:mecontrib="&mecontrib;"
               xmlns:mevd     ="&mevd;"
         xmlns:mecomp   ="&mecomp;"
         xmlns:sioc         ="&sioc;"
         xmlns:rdf      ="&rdf;"
         xmlns:rdfs     ="&rdfs;"
         xmlns:owl      ="&owl;"
         xmlns:dc       ="&dc;"
               xmlns:dcterms  ="&dcterms;"
         xmlns:cc             ="&cc;"
               xmlns:snarm        ="&snarm;"
         xmlns:xsd      ="&xsd;"
>

<!-- ======================== Description ============================== -->
```

```
  <owl:Ontology rdf:about="&mespec;">
    <rdfs:label>myExperiment Specific v1.0</rdfs:label>
    <rdfs:comment>This provides classes and objects specific to myExperiment,
    including SNARM AccessTypes, CreativeCommons Licenses, the TavernaEnactor (a
    Taverna specific Runner) and the myExperiment AnonymousUser.</rdfs:comment>
    <dc:language>en</dc:language>
    <dc:title xml:lang="en">The myExperiment Specific Ontology</dc:title>
    <dc:creator rdf:resource="http://rdf.ecs.soton.ac.uk/person/9421"/>
    <dc:contributor rdf:datatype="http://www.w3.org/2001/XMLSchema#string">David R
    Newman</dc:contributor>
    <dc:publisher rdf:resource="http://rdf.myexperiment.org"/>
    <dc:date rdf:datatype="http://www.w3.org/2001/XMLSchema#date">2009-01-28</dc:date>
    <owl:versionInfo>$Date: 2010/03/31 $</owl:versionInfo>
    <dc:format rdf:datatype="http://www.w3.org/2001/XMLSchema#string">rdf/xml</
    dc:format>
    <owl:imports rdf:resource="&mebase;"/>
    <owl:imports rdf:resource="&meac;"/>
    <owl:imports rdf:resource="&meannot;"/>
    <owl:imports rdf:resource="&mepack;"/>
    <owl:imports rdf:resource="&meexp;"/>
    <owl:imports rdf:resource="&mecontrib;"/>
    <owl:imports rdf:resource="&mevd;"/>
    <owl:imports rdf:resource="&mecomp;"/>
  </owl:Ontology>


<!-- ============= Annotation Properties =========== -->

  <rdf:Description rdf:about="&dc;language">
    <rdf:type rdf:resource="&owl;AnnotationProperty"/>
  </rdf:Description>

  <rdf:Description rdf:about="&dc;title">
    <rdf:type rdf:resource="&owl;AnnotationProperty"/>
  </rdf:Description>

  <rdf:Description rdf:about="&dc;creator">
    <rdf:type rdf:resource="&owl;AnnotationProperty"/>
  </rdf:Description>

  <rdf:Description rdf:about="&dc;contributor">
    <rdf:type rdf:resource="&owl;AnnotationProperty"/>
  </rdf:Description>

  <rdf:Description rdf:about="&dc;publisher">
    <rdf:type rdf:resource="&owl;AnnotationProperty"/>
  </rdf:Description>

  <rdf:Description rdf:about="&dc;date">
    <rdf:type rdf:resource="&owl;AnnotationProperty"/>
  </rdf:Description>

  <rdf:Description rdf:about="&dc;format">
    <rdf:type rdf:resource="&owl;AnnotationProperty"/>
  </rdf:Description>


<!-- ====================== Specific Subclasses ======================== -->
```

```
  <owl:Class rdf:about="TavernaEnactor">
    <rdfs:label>TavernaEnactor</rdfs:label>
    <rdfs:comment>Specific Runners that enact Taverna workflows</rdfs:comment>
    <rdfs:subClassOf rdf:resource="&meexp;Runner"/>
    <rdfs:isDefinedBy rdf:resource="&mespec;"/>
  </owl:Class>

<!-- ================ Retroactive assignments to classes ================ -->

  <rdf:Description rdf:about="&mebase;Group">
    <rdfs:subClassOf rdf:resource="&meannot;Commentable"/>
    <rdfs:subClassOf rdf:resource="&meannot;Taggable"/>
  </rdf:Description>

  <rdf:Description rdf:about="&mecontrib;AbstractWorkflow">
    <rdfs:subClassOf rdf:resource="&meexp;Runnable"/>
  </rdf:Description>


<!-- ========================== Accessers ============================== -->

  <snarm:Accesser rdf:about="Friends">
    <dcterms:title>Friends</dcterms:title>
    <dcterms:description>Anyone that the Contribution creator has a accepted
    Friendship with</dcterms:description>
    <rdfs:isDefinedBy rdf:resource="&mespec;"/>
  </snarm:Accesser>

<!-- ======================== Access Types ============================== -->

  <snarm:AccessType rdf:about="View">
    <dcterms:title rdf:datatype="&xsd;string">View a myExperiment Contribution</
    dcterms:title>
    <rdfs:isDefinedBy rdf:resource="&mespec;"/>
  </snarm:AccessType>

  <snarm:AccessType rdf:about="Download">
    <dcterms:title rdf:datatype="&xsd;string">Download a myExperiment Contribution</
    dcterms:title>
    <rdfs:isDefinedBy rdf:resource="&mespec;"/>
  </snarm:AccessType>

  <snarm:AccessType rdf:about="Edit">
    <dcterms:title rdf:datatype="&xsd;string">Edit a myExperiment Contribution</
    dcterms:title>
    <rdfs:isDefinedBy rdf:resource="&mespec;"/>
  </snarm:AccessType>


<!-- ======================= Access Options ============================ -->

  <snarm:Access rdf:about="PublicView">
    <dcterms:title rdf:datatype="&xsd;string">Anyone can view</dcterms:title>
    <snarm:has-access-type rdf:resource="View"/>
    <rdfs:isDefinedBy rdf:resource="&mespec;"/>
  </snarm:Access>

  <snarm:Access rdf:about="PublicDownload">
```

```
    <dcterms:title rdf:datatype="&xsd;string">Anyone can Download</dcterms:title>
    <snarm:has-access-type rdf:resource="Download"/>
    <rdfs:isDefinedBy rdf:resource="&mespec;"/>
  </snarm:Access>

  <snarm:RestrictedAccess rdf:about="FriendsView">
    <dcterms:title rdf:datatype="&xsd;string">Friends can View</dcterms:title>
    <snarm:has-accesser rdf:resource="Friends"/>
    <snarm:has-access-type rdf:resource="View"/>
    <rdfs:isDefinedBy rdf:resource="&mespec;"/>
  </snarm:RestrictedAccess>

  <snarm:RestrictedAccess rdf:about="FriendsDownload">
    <dcterms:title rdf:datatype="&xsd;string">Friends can Download</dcterms:title>
    <snarm:has-accesser rdf:resource="Friends"/>
    <snarm:has-access-type rdf:resource="Download"/>
    <rdfs:isDefinedBy rdf:resource="&mespec;"/>
  </snarm:RestrictedAccess>

  <snarm:RestrictedAccess rdf:about="FriendsEdit">
    <dcterms:title rdf:datatype="&xsd;string">Friends can Edit</dcterms:title>
    <snarm:has-accesser rdf:resource="Friends"/>
    <snarm:has-access-type rdf:resource="Edit"/>
    <rdfs:isDefinedBy rdf:resource="&mespec;"/>
  </snarm:RestrictedAccess>


<!-- ==================== Anonymous User ========================== -->

<mebase:User rdf:about="AnonymousUser">
  <sioc:name>Anonymous User</sioc:name>
  <dcterms:created rdf:datatype="&xsd;dateTime">1970-01-01T00:00:00</dcterms:created>
  <dcterms:modified rdf:datatype="&xsd;dateTime">1970-01-01T00:00:00</dcterms:modified
    >
  <rdfs:isDefinedBy rdf:resource="&mespec;"/>
</mebase:User>

</rdf:RDF>
```

Listing A.11: myExperiment's Specific Module

A.3 myExperiment Example Entity Data

A.3.1 Examples of All Entities

`http://rdf.myexperiment.org/examples/all_entities` as of 16/10/2011.

```
<?xml version="1.0" encoding="UTF-8" ?>

<!DOCTYPE rdf:RDF [
 <!ENTITY mebase 'http://rdf.myexperiment.org/ontologies/base/'>
 <!ENTITY meac 'http://rdf.myexperiment.org/ontologies/attrib_credit/'>
 <!ENTITY meannot 'http://rdf.myexperiment.org/ontologies/annotations/'>
 <!ENTITY mepack 'http://rdf.myexperiment.org/ontologies/packs/'>
 <!ENTITY meexp 'http://rdf.myexperiment.org/ontologies/experiments/'>
```

```
 <!ENTITY mecontrib 'http://rdf.myexperiment.org/ontologies/contributions/'>
 <!ENTITY mevd 'http://rdf.myexperiment.org/ontologies/viewings_downloads/'>
 <!ENTITY mecomp 'http://rdf.myexperiment.org/ontologies/components/'>
 <!ENTITY mespec 'http://rdf.myexperiment.org/ontologies/specific/'>
 <!ENTITY rdf 'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
 <!ENTITY rdfs 'http://www.w3.org/2000/01/rdf-schema#'>
 <!ENTITY owl 'http://www.w3.org/2002/07/owl#'>
 <!ENTITY dc 'http://purl.org/dc/elements/1.1/'>
 <!ENTITY dcterms 'http://purl.org/dc/terms/'>
 <!ENTITY cc 'http://web.resource.org/cc/'>
 <!ENTITY foaf 'http://xmlns.com/foaf/0.1/'>
 <!ENTITY sioc 'http://rdfs.org/sioc/ns#'>
 <!ENTITY ore 'http://www.openarchives.org/ore/terms/'>
 <!ENTITY dbpedia 'http://dbpedia.org/ontology/'>
 <!ENTITY snarm 'http://rdf.myexperiment.org/ontologies/snarm/'>
 <!ENTITY xsd 'http://www.w3.org/2001/XMLSchema#'>
]>

<rdf:RDF xmlns:mebase   ="&mebase;"
        xmlns:meac      ="&meac;"
        xmlns:meannot   ="&meannot;"
        xmlns:mepack    ="&mepack;"
        xmlns:meexp     ="&meexp;"
        xmlns:mecontrib ="&mecontrib;"
        xmlns:mevd      ="&mevd;"
        xmlns:mecomp    ="&mecomp;"
        xmlns:mespec    ="&mespec;"
        xmlns:rdf       ="&rdf;"
        xmlns:rdfs      ="&rdfs;"
        xmlns:owl       ="&owl;"
        xmlns:dc        ="&dc;"
        xmlns:dcterms   ="&dcterms;"
        xmlns:cc        ="&cc;"
        xmlns:foaf      ="&foaf;"
        xmlns:sioc      ="&sioc;"
        xmlns:ore       ="&ore;"
        xmlns:dbpedia   ="&dbpedia;"
        xmlns:snarm     ="&snarm;"
        xmlns:xsd       ="&xsd;"
>

  <mebase:Announcement rdf:about="http://www.myexperiment.org/announcements/1">
    <foaf:homepage rdf:resource="http://www.myexperiment.org/announcements/1.html"/>
    <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/announcements/1.rdf"/
    >
    <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/announcement.xml?id=1
    "/>
    <dcterms:title rdf:datatype="&xsd;string">New announcements feature!</
    dcterms:title>
    <mebase:has-announcer rdf:resource="http://www.myexperiment.org/users/70" />
    <dcterms:created rdf:datatype="&xsd;dateTime">2008-01-25T15:15:17Z</
    dcterms:created>
    <dcterms:modified rdf:datatype="&xsd;dateTime">2008-01-28T10:50:01Z</
    dcterms:modified>
    <mebase:text rdf:datatype="&xsd;string">&lt;p&gt;myExperiment now has a new
    Announcements feature that allows myExperiment admins to keep you informed on
    progress and other relevant news.&lt;/p&gt;^M
```

```
&lt;p&gt;You can subscribe to the &lt;a href=&quot;http://www.myexperiment.org/
    announcements.rss&quot;&gt;Announcements RSS feed&lt;/a&gt; (and soon we will have
     email notifications).&lt;/p&gt;^M
&lt;p&gt;Don&apos;t forget to give us &lt;a href=&quot;http://www.myexperiment.org/
    feedback&quot;&gt;feedback&lt;/a&gt;!&lt;/p&gt;</mebase:text>
  </mebase:Announcement>

  <meac:Attribution rdf:about="http://www.myexperiment.org/workflows/72/attributions
    /12">
    <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/workflows/72/
    attributions/12.rdf"/>
    <meac:attributes rdf:resource="http://www.myexperiment.org/workflows/72" />
    <meac:has-attributable rdf:resource="http://www.myexperiment.org/workflows/75" />
    <dcterms:created rdf:datatype="&xsd;dateTime">2007-11-15T09:00:44Z</
    dcterms:created>
    <dcterms:modified rdf:datatype="&xsd;dateTime">2007-11-15T09:00:44Z</
    dcterms:modified>
  </meac:Attribution>

  <meannot:Citation rdf:about="http://www.myexperiment.org/workflows/746/versions/1/
    citations/12">
    <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/workflows/746/
    versions/1/citations/12.rdf"/>
    <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/citation.xml?id="/>
    <mebase:has-annotator rdf:resource="http://www.myexperiment.org/users/1019" />
    <mebase:annotates rdf:resource="http://www.myexperiment.org/workflows/746/versions
    /1" />
    <meannot:authors rdf:datatype="&xsd;string">caArray</meannot:authors>
    <dcterms:title rdf:datatype="&xsd;string">Experiment data:  Diffuse large B-cell
    lymphoma outcome prediction</dcterms:title>
    <meannot:published-at rdf:datatype="&xsd;dateTime">2009-04-23T00:00:00Z</
    meannot:published-at>
    <meannot:accessed-at rdf:datatype="&xsd;dateTime">2009-04-23T00:00:00Z</
    meannot:accessed-at>
    <meannot:citation-url rdf:resource="https://array.nci.nih.gov/caarray/project/
    golub-00095" />
    <dcterms:created rdf:datatype="&xsd;dateTime">2009-04-23T17:51:56Z</
    dcterms:created>
    <dcterms:modified rdf:datatype="&xsd;dateTime">2009-04-23T17:51:56Z</
    dcterms:modified>
  </meannot:Citation>

  <mebase:ContentType rdf:about="http://www.myexperiment.org/content_types/1">
    <foaf:homepage rdf:resource="http://www.myexperiment.org/content_types/1.html"/>
    <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/content_types/1.rdf"/
    >
    <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/type.xml?id=1"/>
    <sioc:has_owner rdf:resource="http://www.myexperiment.org/users/30" />
    <dcterms:title rdf:datatype="&xsd;string">Taverna 1</dcterms:title>
    <dcterms:created rdf:datatype="&xsd;dateTime">2009-05-21T10:32:57Z</
    dcterms:created>
    <dcterms:modified rdf:datatype="&xsd;dateTime">2010-04-23T12:56:44Z</
    dcterms:modified>
```

```
  <dcterms:description rdf:datatype="&xsd;string">&lt;p&gt;Taverna is an open source
   family of tools for designing and executing workflows, created by the myGrid
  project and funded by OMII UK, the EPSRC, BBRC, ESRC, JISC and Microsoft.&lt;/p&gt
  ;&lt;p&gt;The family consists of the Taverna Engine (the workhorse), and the
  Taverna Workbench (desktop client) and Taverna Server (remote workflow execution
  server) that sit on top of the Engine. See &lt;a href=&quot;http://www.taverna.org
  .uk/&quot;&gt;http://www.taverna.org.uk/&lt;/a&gt; for further information.&lt;/p&
  gt;&lt;p&gt;The Taverna 1 Workbench is an earlier version of the workbench which
  is still available for download. Taverna 1 workflow descriptions are produced and
  consumed by this version of the workbench. They are in an XML format also known as
   SCUFL.&lt;/p&gt;&lt;p&gt;Note that the latest version of the workbench is Taverna
   2 and is the recommended version. Most Taverna 1 workflows can also be read using
   the Taverna 2 workbench.&lt;/p&gt;&lt;p&gt;Users of the Taverna 1 workbench can
  access myExperiment from the workbench using a ???plugin??? which can be
  downloaded within the workbench.  The plugin allows users to browse, download and
  open workflows from myExperiment within Taverna.&lt;/p&gt;</dcterms:description>
  <dcterms:type rdf:datatype="&xsd;string">application/vnd.taverna.scufl+xml</
  dcterms:type>
</mebase:ContentType>

<meannot:Comment rdf:about="http://www.myexperiment.org/workflows/38/comments/11">
  <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/workflows/38/comments
  /11.rdf"/>
  <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/comment.xml?id=11"/>
  <mebase:has-annotator rdf:resource="http://www.myexperiment.org/users/62" />
  <mebase:annotates rdf:resource="http://www.myexperiment.org/workflows/38" />
  <mebase:text rdf:datatype="&xsd;string">pretty good...will retrieve any pathway
  image</mebase:text>
  <dcterms:created rdf:datatype="&xsd;dateTime">2007-10-03T18:36:31Z</
  dcterms:created>
</meannot:Comment>

<meac:Creditation rdf:about="http://www.myexperiment.org/workflows/69/creditations
  /11">
  <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/workflows/69/
  creditations/11.rdf"/>
  <meac:credits rdf:resource="http://www.myexperiment.org/users/61" />
  <meac:has-creditable rdf:resource="http://www.myexperiment.org/workflows/69" />
  <dcterms:created rdf:datatype="&xsd;dateTime">2007-11-06T17:10:16Z</
  dcterms:created>
  <dcterms:modified rdf:datatype="&xsd;dateTime">2007-11-06T17:10:16Z</
  dcterms:modified>
</meac:Creditation>

<meexp:Experiment rdf:about="http://www.myexperiment.org/experiments/20">
  <foaf:homepage rdf:resource="http://www.myexperiment.org/experiments/20.html"/>
  <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/experiments/20.rdf"/>
  <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/experiment.xml?id=20"
  />
  <ore:isDescribedBy rdf:resource="http://www.myexperiment.org/resource_maps/
  experiments/20" />
  <ore:isDescribedBy rdf:resource="http://www.myexperiment.org/atom_entries/
  experiments/20" />
  <dcterms:title rdf:datatype="&xsd;string">Don&apos;s unique tag experiment</
  dcterms:title>
  <dcterms:description rdf:datatype="&xsd;string">&lt;p&gt;This is a test experiment
   where I will run the Unique Tags example workflow a few times.&lt;/p&gt;</
  dcterms:description>
  <sioc:has_owner rdf:resource="http://www.myexperiment.org/users/22" />
```

```
    <ore:aggregates rdf:resource="http://www.myexperiment.org/experiments/20/jobs/66"/
    >
    <ore:aggregates rdf:resource="http://www.myexperiment.org/experiments/20/jobs/67"/
    >
    <dcterms:created rdf:datatype="&xsd;dateTime">2008-07-25T15:09:08Z</
    dcterms:created>
    <dcterms:modified rdf:datatype="&xsd;dateTime">2008-07-25T15:09:08Z</
    dcterms:modified>
  </meexp:Experiment>

  <meannot:Favourite rdf:about="http://www.myexperiment.org/users/22/favourites/78">
    <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/users/22/favourites
    /78.rdf"/>
    <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/favourite.xml?id=78"/
    >
    <mebase:annotates rdf:resource="http://www.myexperiment.org/workflows/749" />
    <mebase:has-annotator rdf:resource="http://www.myexperiment.org/users/22" />
    <dcterms:created rdf:datatype="&xsd;dateTime">2009-09-30T11:01:59Z</
    dcterms:created>
  </meannot:Favourite>

  <mecontrib:File rdf:about="http://www.myexperiment.org/files/22">
    <foaf:homepage rdf:resource="http://www.myexperiment.org/files/22.html"/>
    <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/files/22.rdf"/>
    <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/file.xml?id=22"/>
    <mebase:content-url rdf:resource="http://www.myexperiment.org/blobs/22/download/
    gaim-forum-users-helping-users.png" />
    <mebase:filename rdf:datatype="&xsd;string">gaim-forum-users-helping-users.png</
    mebase:filename>
    <sioc:has_owner rdf:resource="http://www.myexperiment.org/users/384" />
    <mebase:has-content-type rdf:resource="http://www.myexperiment.org/content_types
    /26" />
    <mebase:has-license rdf:resource="http://www.myexperiment.org/licenses/2" />
    <dcterms:created rdf:datatype="&xsd;dateTime">2008-02-10T01:56:29Z</
    dcterms:created>
    <dcterms:modified rdf:datatype="&xsd;dateTime">2008-06-09T21:53:03Z</
    dcterms:modified>
    <dcterms:title rdf:datatype="&xsd;string">Output from FLOSS Communication
    Centralization Workflow</dcterms:title>
    <dcterms:description rdf:datatype="&xsd;string">&lt;p&gt;This example shows the
    communication dynamics in the Gaim project&apos;s &amp;quot;users helping users&
    amp;quot; forum.&lt;/p&gt;</dcterms:description>
    <mevd:viewed rdf:datatype="&xsd;nonNegativeInteger">489</mevd:viewed>
    <mevd:downloaded rdf:datatype="&xsd;nonNegativeInteger">172</mevd:downloaded>
    <mebase:has-policy rdf:resource="http://www.myexperiment.org/files/22/policies/350
    " />
    <meannot:has-tagging rdf:resource="http://www.myexperiment.org/tags/1353/taggings
    /3872"/>
    <meannot:has-tagging rdf:resource="http://www.myexperiment.org/tags/1354/taggings
    /3873"/>
    <meannot:has-tagging rdf:resource="http://www.myexperiment.org/tags/1355/taggings
    /3874"/>
    <meannot:has-tagging rdf:resource="http://www.myexperiment.org/tags/1356/taggings
    /3875"/>
  </mecontrib:File>

  <mebase:Friendship rdf:about="http://www.myexperiment.org/users/26/friendships/51">
    <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/users/26/friendships
    /51.rdf"/>
```

```
    <mebase:has-requester rdf:resource="http://www.myexperiment.org/users/26" />
    <mebase:has-accepter rdf:resource="http://www.myexperiment.org/users/22" />
    <dcterms:created rdf:datatype="&xsd;dateTime">2007-07-23T12:44:27Z</
    dcterms:created>
    <mebase:accepted-at rdf:datatype="&xsd;dateTime">2007-07-24T16:02:00Z</
    mebase:accepted-at>
  </mebase:Friendship>

  <mebase:Group rdf:about="http://www.myexperiment.org/groups/9">
    <foaf:homepage rdf:resource="http://www.myexperiment.org/groups/9.html"/>
    <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/groups/9.rdf"/>
    <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/group.xml?id=9"/>
    <sioc:has_owner rdf:resource="http://www.myexperiment.org/users/70" />
    <dcterms:created rdf:datatype="&xsd;dateTime">2007-07-23T17:02:58Z</
    dcterms:created>
    <dcterms:modified rdf:datatype="&xsd;dateTime">2008-09-17T09:53:02Z</
    dcterms:modified>
    <sioc:name rdf:datatype="&xsd;string">myExperiment</sioc:name>
    <dcterms:description rdf:datatype="&xsd;string">&lt;p&gt;This is the official
    group for the myExperiment team&lt;/p&gt;</dcterms:description>
    <mebase:auto-accept rdf:datatype="&xsd;boolean">0</mebase:auto-accept>
  </mebase:Group>

  <mebase:GroupAnnouncement rdf:about="http://www.myexperiment.org/groups/183/
    announcements/24">
    <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/groups/183/
    announcements/24.rdf"/>
    <dcterms:title rdf:datatype="&xsd;string">Visit Paris to discuss WikiPedia Scholar
     with Segolene and Denis</dcterms:title>
    <mebase:announced-to rdf:resource="http://www.myexperiment.org/groups/183" />
    <mebase:has-announcer rdf:resource="http://www.myexperiment.org/users/2213" />
    <mebase:public-announcement rdf:datatype="&xsd;boolean">1</mebase:public-
    announcement>
    <dcterms:created rdf:datatype="&xsd;dateTime">2009-08-28T08:19:07Z</
    dcterms:created>
    <dcterms:modified rdf:datatype="&xsd;dateTime">2009-08-28T08:19:07Z</
    dcterms:modified>
    <mebase:text rdf:datatype="&xsd;string">&lt;p&gt;Hi, this coming Monday (31st of
    August) I will be in Paris to discuss with Segolene Ayme and Denis Costello about
    the WikiPedia Scholar project. The discussions will be based on the document
    uploaded to thsi group earlier. If anyone has points that seem to be missed in the
     document or discussions so far, can you please make them as comments to this
    group ?&lt;/p&gt;^M

&lt;p&gt;I will report about the meeting in this group as well.&lt;br /&gt;^M
Barend&lt;/p&gt;</mebase:text>
  </mebase:GroupAnnouncement>

  <meexp:Job rdf:about="http://www.myexperiment.org/experiments/20/jobs/67">
    <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/experiments/20/jobs
    /67.rdf"/>
    <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/job.xml?id=67"/>
    <ore:isDescribedBy rdf:resource="http://www.myexperiment.org/resource_maps/jobs/67
    " />
    <ore:isDescribedBy rdf:resource="http://www.myexperiment.org/atom_entries/jobs/67"
     />
    <dcterms:title rdf:datatype="&xsd;string">Job_20080910-1323_Don Cruickshank</
    dcterms:title>
    <ore:isAggregatedBy rdf:resource="http://www.myexperiment.org/experiments/20" />
```

```
    <sioc:has_owner rdf:resource="http://www.myexperiment.org/users/22" />
    <meexp:has-runnable rdf:resource="http://www.myexperiment.org/workflows/154/
    versions/8" />
    <meexp:has-runner rdf:resource="http://www.myexperiment.org/runners/5" />
    <meexp:submitted-at rdf:datatype="&xsd;dateTime">2008-09-10T13:24:38Z</
    meexp:submitted-at>
    <meexp:started-at rdf:datatype="&xsd;dateTime">2008-09-10T14:24:38Z</meexp:started
    -at>
    <meexp:completed-at rdf:datatype="&xsd;dateTime">2008-09-10T14:25:40Z</
    meexp:completed-at>
    <meexp:last-status rdf:datatype="&xsd;string">COMPLETE</meexp:last-status>
    <meexp:last-status-at rdf:datatype="&xsd;dateTime">2009-01-30T14:45:14Z</
    meexp:last-status-at>
    <mebase:uri rdf:resource="http://tavernaenactor.example.com:1234/remotetaverna/v1/
    jobs/f1ae6518-134c-467a-9b93-ff922fa586d0" />
    <meexp:has-input>
      <meexp:Data rdf:about="http://www.myexperiment.org/experiments/20/jobs/67/input"
    >
        <mebase:uri rdf:resource="http://tavernaenactor.example.com:1234/remotetaverna
    /v1/data/d782bc1b-9a2e-4a80-a71e-21e19a643470"/>
        <mebase:text rdf:datatype="&xsd;string">---
query_protein: P53
        </mebase:text>
      </meexp:Data>
    </meexp:has-input>
    <meexp:has-output>
      <meexp:Data rdf:about="http://www.myexperiment.org/experiments/20/jobs/67/output
    ">
        <mebase:uri rdf:resource="http://tavernaenactor.example.com:1234/remotetaverna
    /v1/data/578c39d6-4a94-494b-81ac-a520e00eeac6"/>
      </meexp:Data>
    </meexp:has-output>
    <dcterms:created rdf:datatype="&xsd;dateTime">2008-09-10T13:24:13Z</
    dcterms:created>
    <dcterms:modified rdf:datatype="&xsd;dateTime">2009-01-30T14:45:14Z</
    dcterms:modified>
  </meexp:Job>

  <mepack:LocalPackEntry rdf:about="http://www.myexperiment.org/packs/1/
    local_pack_entries/587">
    <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/packs/1/
    local_pack_entries/587.rdf"/>
    <ore:proxyIn rdf:resource="http://www.myexperiment.org/packs/1" />
    <ore:proxyFor rdf:resource="http://www.myexperiment.org/files/362" />
    <sioc:has_owner rdf:resource="http://www.myexperiment.org/users/221" />
    <dcterms:created rdf:datatype="&xsd;dateTime">2010-02-04T11:41:11Z</
    dcterms:created>
    <dcterms:modified rdf:datatype="&xsd;dateTime">2010-02-04T11:41:11Z</
    dcterms:modified>
  </mepack:LocalPackEntry>

  <mebase:Membership rdf:about="http://www.myexperiment.org/users/26/memberships/9">
    <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/users/26/memberships
    /9.rdf"/>
    <mebase:has-requester rdf:resource="http://www.myexperiment.org/users/26" />
    <mebase:has-accepter rdf:resource="http://www.myexperiment.org/groups/9" />
    <dcterms:created rdf:datatype="&xsd;dateTime">2007-10-03T18:35:44Z</
    dcterms:created>
```

```
  <mebase:accepted-at rdf:datatype="&xsd;dateTime">2007-10-03T18:35:44Z</
  mebase:accepted-at>
</mebase:Membership>

<mebase:Message rdf:about="http://www.myexperiment.org/messages/26">
  <foaf:homepage rdf:resource="http://www.myexperiment.org/messages/26.html"/>
  <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/messages/26.rdf"/>
  <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/message.xml?id=26"/>
  <mebase:from rdf:resource="http://www.myexperiment.org/users/16" />
  <mebase:to rdf:resource="http://www.myexperiment.org/users/22" />
  <mebase:subject rdf:datatype="&xsd;string">remember remember</mebase:subject>
  <mebase:text rdf:datatype="&xsd;string">The 5th of November</mebase:text>
  <dcterms:created rdf:datatype="&xsd;dateTime">2007-11-05T20:38:01Z</
  dcterms:created>
  <mebase:read-at rdf:datatype="&xsd;dateTime">2007-12-11T18:01:14Z</mebase:read-at>
  <mebase:deleted-by-sender rdf:datatype="&xsd;boolean">0</mebase:deleted-by-sender>
</mebase:Message>

<mepack:Pack rdf:about="http://www.myexperiment.org/packs/1">
  <foaf:homepage rdf:resource="http://www.myexperiment.org/packs/1.html"/>
  <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/packs/1.rdf"/>
  <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/packs/1.xml"/>
  <ore:aggregates rdf:resource="http://www.myexperiment.org/files/2"/>
  <ore:aggregates rdf:resource="http://www.myexperiment.org/files/1"/>
  <ore:aggregates rdf:resource="http://www.myexperiment.org/workflows/1"/>
  <ore:aggregates rdf:resource="http://www.myexperiment.org/relationships/
  sodfudghsodfugfhsd"/>
  <ore:aggregates rdf:resource="http://www.myexperiment.org/relationships/
  wuodfghufoghurosf"/>
  <mepack:has-pack-entry rdf:resource="http://www.myexperiment.org/packs/1/
  local_pack_entries/1"/>
  <mepack:has-pack-entry rdf:resource="http://www.myexperiment.org/packs/1/
  local_pack_entries/2"/>
  <mepack:has-pack-entry rdf:resource="http://www.myexperiment.org/packs/1/
  local_pack_entries/4"/>
  <mepack:has-pack-entry rdf:resource="http://www.myexperiment.org/packs/1/
  relationships/1"/>
  <mepack:has-pack-entry rdf:resource="http://www.myexperiment.org/packs/1/
  relationships/2"/>
  <sioc:has_owner rdf:resource="http://www.myexperiment.org/users/1" />
  <dcterms:title rdf:datatype="&xsd;string">Test pack</dcterms:title>
  <dcterms:description rdf:datatype="&xsd;string">&lt;p&gt;This is a pack that will&
  amp;nbsp;contain relationships.&lt;/p&gt;</dcterms:description>
  <dcterms:created rdf:datatype="&xsd;dateTime">2011-02-01T16:55:05Z</
  dcterms:created>
  <dcterms:modified rdf:datatype="&xsd;dateTime">2011-02-01T16:55:05Z</
  dcterms:modified>
  <mevd:viewed rdf:datatype="&xsd;nonNegativeInteger">10</mevd:viewed>
  <mebase:has-policy rdf:resource="http://www.myexperiment.org/packs/1/policies/26"
  />
</mepack:Pack>

<mepack:RelationshipEntry rdf:about="http://www.myexperiment.org/packs/1/
  relationship_entries/1">
  <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/packs/1/
  relationship_entries/1.rdf"/>
  <ore:proxyIn rdf:resource="http://www.myexperiment.org/packs/1" />
  <ore:proxyFor rdf:resource="http://www.myexperiment.org/relationships/
  wdoufhduioghrhgsdhudo"/>
```

```
    <sioc:has_owner rdf:resource="http://www.myexperiment.org/users/1" />
    <dcterms:created rdf:datatype="&xsd;dateTime">2011-02-10T17:54:21Z</
    dcterms:created>
  </mepack:RelationshipEntry>

  <snarm:Policy rdf:about="http://www.myexperiment.org/Workflow/16/policies/185">
    <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/Workflow/16/policies
    /185.rdf"/>
    <snarm:has-access>
      <snarm:RestrictedAccess>
        <snarm:has-accesser rdf:resource="http://www.myexperiment.org/users/43"/>
        <snarm:has-access-type rdf:resource="&mespec;View"/>
      </snarm:RestrictedAccess>
    </snarm:has-access>
    <snarm:has-access>
      <snarm:RestrictedAccess>
        <snarm:has-accesser rdf:resource="http://www.myexperiment.org/users/43"/>
        <snarm:has-access-type rdf:resource="&mespec;Download"/>
      </snarm:RestrictedAccess>
    </snarm:has-access>
    <snarm:has-access>
      <snarm:RestrictedAccess>
        <snarm:has-accesser rdf:resource="http://www.myexperiment.org/users/43"/>
        <snarm:has-access-type rdf:resource="&mespec;Edit"/>
      </snarm:RestrictedAccess>
    </snarm:has-access>
    <snarm:has-access rdf:resource="&mespec;PublicView"/>
    <snarm:has-access rdf:resource="&mespec;PublicDownload"/>
    <snarm:has-access rdf:resource="&mespec;FriendsEdit"/>
  </snarm:Policy>

  <meannot:Rating rdf:about="http://www.myexperiment.org/workflows/7/ratings/2">
    <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/workflows/7/ratings
    /2.rdf"/>
    <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/rating.xml?id=2"/>
    <mebase:annotates rdf:resource="http://www.myexperiment.org/workflows/7" />
    <meannot:rating-score rdf:datatype="&xsd;positiveInteger">5</meannot:rating-score>
    <mebase:has-annotator rdf:resource="http://www.myexperiment.org/users/1" />
    <dcterms:created rdf:datatype="&xsd;dateTime">2007-10-03T18:36:31Z</
    dcterms:created>
  </meannot:Rating>

  <mepack:Relationship rdf:about="http://www.myexperiment.org/relationships/
    wdoufhduioghrhgsdhudo">
    <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/relationships/
    wdoufhduioghrhgsdhudo.rdf"/>
    <rdf:subject rdf:resource="http://www.myexperiment.org/files/2" />
    <rdf:predicate rdf:resource="http://www.myexperiment.org/vocabularies/1/concepts/9
    " />
    <rdf:object rdf:resource="http://www.myexperiment.org/workflows/1" />
  </mepack:Relationship>

  <mepack:RemotePackEntry rdf:about="http://www.myexperiment.org/packs/8/
    remote_pack_entries/13">
    <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/packs/8/
    remote_pack_entries/13.rdf"/>
    <ore:proxyIn rdf:resource="http://www.myexperiment.org/packs/8" />
    <dcterms:title rdf:datatype="&xsd;string">Link</dcterms:title>
    <ore:proxyFor>
```

```
    <rdf:Description rdf:about="http://www.mygrid.org.uk/ontology"/>
  </ore:proxyFor>
  <dcterms:description rdf:datatype="&xsd;string">the myGrid ontology</
  dcterms:description>
  <sioc:has_owner rdf:resource="http://www.myexperiment.org/users/61" />
  <dcterms:created rdf:datatype="&xsd;dateTime">2008-07-12T08:35:01Z</
  dcterms:created>
  <dcterms:modified rdf:datatype="&xsd;dateTime">2008-07-12T08:35:01Z</
  dcterms:modified>
</mepack:RemotePackEntry>

<meannot:Tag rdf:about="http://www.myexperiment.org/tags/69">
  <foaf:homepage rdf:resource="http://www.myexperiment.org/tags/69.html"/>
  <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/tags/69.rdf"/>
  <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/tag.xml?id=69"/>
  <dcterms:title rdf:datatype="&xsd;string">chromosome</dcterms:title>
  <meannot:for-tagging rdf:resource="http://www.myexperiment.org/tags/69/taggings
  /3422"/>
  <meannot:for-tagging rdf:resource="http://www.myexperiment.org/tags/69/taggings
  /4513"/>
  <meannot:for-tagging rdf:resource="http://www.myexperiment.org/tags/69/taggings
  /3822"/>
  <meannot:for-tagging rdf:resource="http://www.myexperiment.org/tags/69/taggings
  /4631"/>
  <meannot:for-tagging rdf:resource="http://www.myexperiment.org/tags/69/taggings
  /4826"/>
</meannot:Tag>

<meannot:Tagging rdf:about="http://www.myexperiment.org/tags/402/taggings/214">
  <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/tags/402/taggings
  /214.rdf"/>
  <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/tagging.xml?id=214"/>
  <meannot:uses-tag rdf:resource="http://www.myexperiment.org/tags/402" />
  <mebase:annotates rdf:resource="http://www.myexperiment.org/workflows/31" />
  <mebase:has-annotator rdf:resource="http://www.myexperiment.org/users/18" />
  <dcterms:created rdf:datatype="&xsd;dateTime">2007-11-07T19:13:23Z</
  dcterms:created>
</meannot:Tagging>

<mespec:TavernaEnactor rdf:about="http://www.myexperiment.org/runners/5">
  <foaf:homepage rdf:resource="http://www.myexperiment.org/runners/5.html"/>
  <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/runners/5.rdf"/>
  <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/runner.xml?id=5"/>
  <dcterms:title rdf:datatype="&xsd;string">aida runner</dcterms:title>
  <dcterms:description rdf:datatype="&xsd;string">aida runner</dcterms:description>
  <sioc:has_owner rdf:resource="http://www.myexperiment.org/groups/62" />
  <meexp:runner-url rdf:resource="http://tavernaenactor.example.com:1234/
  remotetaverna/v1/" />
  <mebase:username rdf:datatype="&xsd;string">marco</mebase:username>
  <dcterms:created rdf:datatype="&xsd;dateTime">2008-04-10T16:05:45Z</
  dcterms:created>
  <dcterms:modified rdf:datatype="&xsd;dateTime">2008-12-03T11:20:58Z</
  dcterms:modified>
</mespec:TavernaEnactor>

<mebase:User rdf:about="http://www.myexperiment.org/users/26">
  <foaf:homepage rdf:resource="http://www.myexperiment.org/users/26.html"/>
  <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/users/26.rdf"/>
  <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/user.xml?id=26"/>
```

```
    <dcterms:created rdf:datatype="&xsd;dateTime">2007-07-23T12:34:33Z</
    dcterms:created>
    <dcterms:modified rdf:datatype="&xsd;dateTime">2010-05-24T14:22:43Z</
    dcterms:modified>
    <sioc:name rdf:datatype="&xsd;string">David R Newman</sioc:name>
    <sioc:avatar rdf:resource="http://www.myexperiment.org/pictures/show/45?size=160
    x160.png" />
    <foaf:based_near rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
    Southampton</foaf:based_near>
    <mebase:country rdf:datatype="&xsd;string">United Kingdom</mebase:country>
    <dbpedia:residence rdf:resource="http://dbpedia.org/resource/Southampton"/>
    <dbpedia:residence rdf:resource="http://dbpedia.org/resource/United_Kingdom"/>
    <foaf:homepage rdf:resource="http://www.ecs.soton.ac.uk/~drn05r" />
    <mebase:last-seen-at rdf:datatype="&xsd;dateTime">2010-05-24T14:07:31Z</
    mebase:last-seen-at>
    <mebase:activated-at rdf:datatype="&xsd;dateTime">2007-07-23T12:34:33Z</
    mebase:activated-at>
    <mebase:receive-notifications rdf:datatype="&xsd;boolean">1</mebase:receive-
    notifications>
    <foaf:mbox rdf:resource="mailto:drn05r@ecs.soton.ac.uk" />
    <foaf:mbox_sha1sum rdf:datatype="&xsd;string">
    da39a3ee5e6b4b0d3255bfef95601890afd80709</foaf:mbox_sha1sum>
    <mebase:field rdf:datatype="&xsd;string">University Research</mebase:field>
    <mebase:has-friendship rdf:resource="http://www.myexperiment.org/users/26/
    friendships/51"/>
    <mebase:has-friendship rdf:resource="http://www.myexperiment.org/users/26/
    friendships/52"/>
    <mebase:has-friendship rdf:resource="http://www.myexperiment.org/users/26/
    friendships/53"/>
    <mebase:has-friendship rdf:resource="http://www.myexperiment.org/users/26/
    friendships/54"/>
    <mebase:has-friendship rdf:resource="http://www.myexperiment.org/users/43/
    friendships/92"/>
    <mebase:has-friendship rdf:resource="http://www.myexperiment.org/users/1/
    friendships/156"/>
    <mebase:has-friendship rdf:resource="http://www.myexperiment.org/users/70/
    friendships/245"/>
    <mebase:has-friendship rdf:resource="http://www.myexperiment.org/users/164/
    friendships/474"/>
    <mebase:has-friendship rdf:resource="http://www.myexperiment.org/users/286/
    friendships/946"/>
    <mebase:has-membership rdf:resource="http://www.myexperiment.org/users/26/
    memberships/9"/>
    <mebase:has-membership rdf:resource="http://www.myexperiment.org/users/26/
    memberships/548"/>
    <mebase:has-membership rdf:resource="http://www.myexperiment.org/users/26/
    memberships/556"/>
    <mebase:has-membership rdf:resource="http://www.myexperiment.org/users/26/
    memberships/601"/>
    <mebase:has-favourite rdf:resource="http://www.myexperiment.org/users/26/
    favourites/22"/>
  </mebase:User>

  <mecontrib:Workflow rdf:about="http://www.myexperiment.org/workflows/16">
    <foaf:homepage rdf:resource="http://www.myexperiment.org/workflows/16.html"/>
    <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/workflows/16.rdf"/>
    <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/workflow.xml?id=16"/>
    <dcterms:title rdf:datatype="&xsd;string">Pathways and Gene annotations for QTL
    region</dcterms:title>
```

```
    <dcterms:description rdf:datatype="&xsd;string">&lt;p&gt;This workflow searches
    for genes which reside in a QTL (Quantitative Trait Loci) region in the mouse, Mus
     musculus. The workflow requires an input of: a chromosome name or number; a QTL
    start base pair position; QTL end base pair position. Data is then extracted from
    BioMart to annotate each of the genes found in this region. The Entrez and UniProt
     identifiers are then sent to KEGG to obtain KEGG gene identifiers. The KEGG gene
    identifiers are then used to searcg for pathways in the KEGG pathway database.&lt
    ;/p&gt;^M

&lt;p&gt;Example input:&lt;/p&gt;^M
&lt;p&gt;chromosome_name: 17&lt;/p&gt;^M
&lt;p&gt;start_position: 28500000&lt;/p&gt;^M
&lt;p&gt;end_position: 32500000&lt;/p&gt;</dcterms:description>
    <mebase:has-content-type rdf:resource="http://www.myexperiment.org/content_types/1
    " />
    <sioc:has_owner rdf:resource="http://www.myexperiment.org/users/43" />
    <dcterms:created rdf:datatype="&xsd;dateTime">2009-11-19T18:18:52Z</
    dcterms:created>
    <dcterms:modified rdf:datatype="&xsd;dateTime">2009-11-20T10:33:25Z</
    dcterms:modified>
    <mebase:filename rdf:datatype="&xsd;string">
    pathways_and_gene_annotations_forqtl_region_740037.xml</mebase:filename>
    <ore:isDescribedBy rdf:resource="http://www.myexperiment.org/resource_maps/
    workflows/16" />
    <ore:isDescribedBy rdf:resource="http://www.myexperiment.org/atom_entries/
    workflows/16" />
    <mebase:content-url rdf:resource="http://www.myexperiment.org/workflows/16/
    download/pathways_and_gene_annotations_forqtl_region_740037.xml" />
    <mecontrib:preview rdf:resource="http://www.myexperiment.org/workflow/image/16/
    pathways_and_gene_annotations_forqtl_region.png" />
    <mecontrib:thumbnail rdf:resource="http://www.myexperiment.org/workflow/image/16/
    thumb/pathways_and_gene_annotations_forqtl_region.png" />
    <mecontrib:thumbnail-big rdf:resource="http://www.myexperiment.org/workflow/image
    /16/medium/pathways_and_gene_annotations_forqtl_region.png" />
    <mecontrib:svg rdf:resource="http://www.myexperiment.org/workflow/svg/16/
    pathways_and_gene_annotations_forqtl_region.svg.xml" />
    <mebase:has-current-version rdf:resource="http://www.myexperiment.org/workflows
    /16/versions/5" />
    <mebase:has-version rdf:resource="http://www.myexperiment.org/workflows/16/
    versions/1"/>
    <mebase:has-version rdf:resource="http://www.myexperiment.org/workflows/16/
    versions/2"/>
    <mebase:has-version rdf:resource="http://www.myexperiment.org/workflows/16/
    versions/3"/>
    <mebase:has-version rdf:resource="http://www.myexperiment.org/workflows/16/
    versions/4"/>
    <mebase:has-license rdf:resource="http://www.myexperiment.org/licenses/2" />
    <mebase:last-edited-by rdf:resource="http://www.myexperiment.org/users/43" />
    <mevd:viewed rdf:datatype="&xsd;nonNegativeInteger">5457</mevd:viewed>
    <mevd:downloaded rdf:datatype="&xsd;nonNegativeInteger">1867</mevd:downloaded>
    <mebase:has-policy rdf:resource="http://www.myexperiment.org/workflows/16/policies
    /185" />
    <mecomp:executes-dataflow rdf:resource="http://www.myexperiment.org/workflows/16/
    versions/5/dataflow" />
    <meannot:has-tagging rdf:resource="http://www.myexperiment.org/tags/402/taggings
    /214"/>
    <meannot:has-tagging rdf:resource="http://www.myexperiment.org/tags/561/taggings
    /215"/>
```

```
    <meannot:has-tagging rdf:resource="http://www.myexperiment.org/tags/562/taggings
    /216"/>
    <meannot:has-tagging rdf:resource="http://www.myexperiment.org/tags/38/taggings
    /446"/>
    <meannot:has-tagging rdf:resource="http://www.myexperiment.org/tags/5/taggings/447
    "/>
    <meannot:has-tagging rdf:resource="http://www.myexperiment.org/tags/611/taggings
    /448"/>
    <meannot:has-tagging rdf:resource="http://www.myexperiment.org/tags/612/taggings
    /449"/>
    <meannot:has-tagging rdf:resource="http://www.myexperiment.org/tags/522/taggings
    /450"/>
  </mecontrib:Workflow>

  <mecontrib:WorkflowVersion rdf:about="http://www.myexperiment.org/workflows/16/
    versions/1">
    <foaf:homepage rdf:resource="http://www.myexperiment.org/workflows/16/versions/1.
    html"/>
    <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/workflows/16/versions
    /1.rdf"/>
    <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/workflow.xml?id=16&
    amp;version=1"/>
    <dcterms:title rdf:datatype="&xsd;string">Pathways and Gene annotations for QTL
    Phenotype</dcterms:title>
    <dcterms:description rdf:datatype="&xsd;string">This workflow searches for genes
    which reside in a QTL (Quantitative Trait Loci) region in the mouse, Mus musculus.
     The workflow requires an input of: a chromosome name or number; a QTL start base
    pair position; QTL end base pair position. Data is then extracted from BioMart to
    annotate each of the genes found in this region. The Entrez and UniProt
    identifiers are then sent to KEGG to obtain KEGG gene identifiers. The KEGG gene
    identifiers are then used to searcg for pathways in the KEGG pathway database.</
    dcterms:description>
    <mebase:has-content-type rdf:resource="http://www.myexperiment.org/content_types/1
    " />
    <dcterms:isVersionOf rdf:resource="http://www.myexperiment.org/workflows/16" />
    <mebase:version-number rdf:datatype="&xsd;positiveInteger">1</mebase:version-
    number>
    <mebase:is-current-version rdf:datatype="&xsd;boolean">0</mebase:is-current-
    version>
    <sioc:has_owner rdf:resource="http://www.myexperiment.org/users/43" />
    <dcterms:created rdf:datatype="&xsd;dateTime">2007-10-03T18:36:02Z</
    dcterms:created>
    <dcterms:modified rdf:datatype="&xsd;dateTime">2007-10-03T18:36:02Z</
    dcterms:modified>
    <mebase:filename rdf:datatype="&xsd;string">
    pathways_and_gene_annotations_for_qtl_phenotype_25941.xml</mebase:filename>
    <ore:isDescribedBy rdf:resource="http://www.myexperiment.org/resource_maps/
    workflow_versions/31" />
    <ore:isDescribedBy rdf:resource="http://www.myexperiment.org/atom_entries/
    workflow_versions/31" />
    <mebase:content-url rdf:resource="http://www.myexperiment.org/workflows/16/
    download/pathways_and_gene_annotations_for_qtl_phenotype_25941.xml?version=1" />
    <mecontrib:preview rdf:resource="http://www.myexperiment.org/workflow/version/
    image/31/pathways_and_gene_annotations_for_qtl_phenotype_25941_1.png" />
    <mecontrib:thumbnail rdf:resource="http://www.myexperiment.org/workflow/version/
    image/31/thumb/pathways_and_gene_annotations_for_qtl_phenotype_25941_1.png" />
    <mecontrib:thumbnail-big rdf:resource="http://www.myexperiment.org/workflow/
    version/image/31/medium/pathways_and_gene_annotations_for_qtl_phenotype_25941_1.
    png" />
```

```
    <mecontrib:svg rdf:resource="http://www.myexperiment.org/workflow/version/svg/31/
    pathways_and_gene_annotations_for_qtl_phenotype_25941_1.svg" />
    <mebase:has-license rdf:resource="http://www.myexperiment.org/licenses/2" />
    <mebase:last-edited-by rdf:resource="http://www.myexperiment.org/users/43" />
    <mebase:has-policy rdf:resource="http://www.myexperiment.org/workflows/16/policies
    /185" />
    <mecomp:executes-dataflow rdf:resource="http://www.myexperiment.org/workflows/16/
    versions/1/dataflow" />
  </mecontrib:WorkflowVersion>

</rdf:RDF>
```

Listing A.12: Examples of All myExperiment Entities

A.3.2 Example of a Relationship Entry

`http://rdf.myexperiment.org/examples/relationship_entries` as of 10/10/2011.

```
<?xml version="1.0" encoding="UTF-8" ?>

<!DOCTYPE rdf:RDF [
 <!ENTITY mebase 'http://rdf.myexperiment.org/ontologies/base/'>
 <!ENTITY meac 'http://rdf.myexperiment.org/ontologies/attrib_credit/'>
 <!ENTITY meannot 'http://rdf.myexperiment.org/ontologies/annotations/'>
 <!ENTITY mepack 'http://rdf.myexperiment.org/ontologies/packs/'>
 <!ENTITY meexp 'http://rdf.myexperiment.org/ontologies/experiments/'>
 <!ENTITY mecontrib 'http://rdf.myexperiment.org/ontologies/contributions/'>
 <!ENTITY mevd 'http://rdf.myexperiment.org/ontologies/viewings_downloads/'>
 <!ENTITY mecomp 'http://rdf.myexperiment.org/ontologies/components/'>
 <!ENTITY mespec 'http://rdf.myexperiment.org/ontologies/specific/'>
 <!ENTITY rdf 'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
 <!ENTITY rdfs 'http://www.w3.org/2000/01/rdf-schema#'>
 <!ENTITY owl 'http://www.w3.org/2002/07/owl#'>
 <!ENTITY dc 'http://purl.org/dc/elements/1.1/'>
 <!ENTITY dcterms 'http://purl.org/dc/terms/'>
 <!ENTITY cc 'http://web.resource.org/cc/'>
 <!ENTITY foaf 'http://xmlns.com/foaf/0.1/'>
 <!ENTITY sioc 'http://rdfs.org/sioc/ns#'>
 <!ENTITY skos 'http://www.w3.org/2004/02/skos/core#'>
 <!ENTITY ore 'http://www.openarchives.org/ore/terms/'>
 <!ENTITY dbpedia 'http://dbpedia.org/ontology/'>
 <!ENTITY snarm 'http://rdf.myexperiment.org/ontologies/snarm/'>
 <!ENTITY xsd 'http://www.w3.org/2001/XMLSchema#'>
]>

<rdf:RDF xmlns:mebase   ="&mebase;"
         xmlns:meac     ="&meac;"
         xmlns:meannot  ="&meannot;"
         xmlns:mepack   ="&mepack;"
         xmlns:meexp    ="&meexp;"
         xmlns:mecontrib="&mecontrib;"
         xmlns:mevd     ="&mevd;"
         xmlns:mecomp   ="&mecomp;"
         xmlns:mespec   ="&mespec;"
         xmlns:rdf      ="&rdf;"
         xmlns:rdfs     ="&rdfs;"
         xmlns:owl      ="&owl;"
```

```
        xmlns:dc        ="&dc;"
        xmlns:dcterms   ="&dcterms;"
        xmlns:cc        ="&cc;"
        xmlns:foaf      ="&foaf;"
        xmlns:sioc      ="&sioc;"
        xmlns:skos      ="&skos;"
        xmlns:ore       ="&ore;"
        xmlns:dbpedia   ="&dbpedia;"
        xmlns:snarm     ="&snarm;"
        xmlns:xsd       ="&xsd;"
>
  <rdf:Description rdf:about="http://www.myexperiment.org/packs/187/
    relationship_entries/1.rdf">
    <foaf:primaryTopic rdf:resource="http://www.myexperiment.org/packs/187/
    relationship_entries/1"/>
  </rdf:Description>

  <mepack:RelationshipEntry rdf:about="http://www.myexperiment.org/packs/187/
    relationship_entries/1">
    <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/packs/187/
    relationship_entries/1.rdf"/>
    <ore:proxyFor>
      <mepack:Relationship rdf:about="urn:uuid:77a2b953-3b32-5b9f-bfb6-3c0b09a54669">
        <rdf:subject rdf:resource="http://www.myexperiment.org/files/141" />
        <rdf:predicate rdf:resource="http://www.myexperiment.org/2010/pack/inputDataTo
    " />
        <rdf:object rdf:resource="http://www.myexperiment.org/workflows/173" />
      </mepack:Relationship>
    </ore:proxyFor>
    <ore:proxyIn rdf:resource="http://www.myexperiment.org/packs/187" />
    <sioc:has_owner rdf:resource="http://www.myexperiment.org/users/22" />
    <dcterms:created rdf:datatype="&xsd;dateTime">2011-06-03T18:05:04Z</
    dcterms:created>
  </mepack:RelationshipEntry>

  <mecontrib:File rdf:about="http://www.myexperiment.org/files/141">
    <foaf:homepage rdf:resource="http://www.myexperiment.org/files/141.html"/>
    <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/files/141.rdf"/>
    <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/files/141.xml"/>
    <mebase:content-url rdf:resource="http://www.myexperiment.org/blobs/141/download/
    Astronomic" />
    <mebase:filename rdf:datatype="&xsd;string">Astronomic</mebase:filename>
    <sioc:has_owner rdf:resource="http://www.myexperiment.org/users/22" />
    <mebase:has-content-type rdf:resource="http://www.myexperiment.org/content_types
    /16" />
    <mebase:has-license rdf:resource="http://www.myexperiment.org/licenses/2" />
    <dcterms:created rdf:datatype="&xsd;dateTime">2008-10-08T13:31:47Z</
    dcterms:created>
    <dcterms:modified rdf:datatype="&xsd;dateTime">2008-10-08T13:31:47Z</
    dcterms:modified>
    <dcterms:title rdf:datatype="&xsd;string">One</dcterms:title>
    <dcterms:description rdf:datatype="&xsd;string">&lt;p&gt;Two&lt;/p&gt;</
    dcterms:description>
    <mevd:viewed rdf:datatype="&xsd;nonNegativeInteger">2</mevd:viewed>
    <mebase:has-policy rdf:resource="http://www.myexperiment.org/files/141/policies
    /895" />
    <meannot:has-tagging rdf:resource="http://www.myexperiment.org/tags/1869/taggings
    /6041"/>
  </mecontrib:File>
```

```
<owl:ObjectProperty rdf:about="http://www.myexperiment.org/2010/pack/inputDataTo">
  <rdfs:label rdf:datatype="&xsd;string">inputDataTo</rdfs:label>
  <rdfs:comment rdf:datatype="&xsd;string">This concept is used to indicate that the
   subject is an input to the object.</rdfs:comment>
  <rdfs:isDefinedBy rdf:resource="http://www.myexperiment.org/2010/pack" />
  <dcterms:created rdf:datatype="&xsd;dateTime">2011-06-03T18:04:56Z</
  dcterms:created>
  <dcterms:modified rdf:datatype="&xsd;dateTime">2011-06-03T18:04:56Z</
  dcterms:modified>
</owl:ObjectProperty>

<mecontrib:Workflow rdf:about="http://www.myexperiment.org/workflows/173">
  <foaf:homepage rdf:resource="http://www.myexperiment.org/workflows/173.html"/>
  <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/workflows/173.rdf"/>
  <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/workflows/173.xml"/>
  <dcterms:title rdf:datatype="&xsd;string">Unique tags</dcterms:title>
  <dcterms:description rdf:datatype="&xsd;string">This workflow takes a comma
  separated list of tags and removes duplicate entries.  Tags may have multiple
  words in them.  An example string is &quot;carrots,handbags,carrots,cheese&quot;.<
  /dcterms:description>
  <mebase:has-content-type rdf:resource="http://www.myexperiment.org/content_types/1
  " />
  <sioc:has_owner rdf:resource="http://www.myexperiment.org/users/22" />
  <dcterms:created rdf:datatype="&xsd;dateTime">2008-03-11T16:52:42Z</
  dcterms:created>
  <dcterms:modified rdf:datatype="&xsd;dateTime">2008-03-11T16:53:13Z</
  dcterms:modified>
  <mebase:filename rdf:datatype="&xsd;string">unique_tags_18054.xml</mebase:filename
  >
  <mebase:content-url rdf:resource="http://www.myexperiment.org/workflows/173/
  download/unique_tags_18054.xml" />
  <mecontrib:preview rdf:resource="http://www.myexperiment.org/workflows/173/
  previews/full" />
  <mecontrib:thumbnail rdf:resource="http://www.myexperiment.org/workflows/173/
  previews/thumb" />
  <mecontrib:thumbnail-big rdf:resource="http://www.myexperiment.org/workflows/173/
  previews/medium" />
  <mecontrib:svg rdf:resource="http://www.myexperiment.org/workflows/173/previews/
  svg" />
  <mebase:has-current-version rdf:resource="http://www.myexperiment.org/workflows
  /173/versions/2" />
  <mebase:has-version rdf:resource="http://www.myexperiment.org/workflows/173/
  versions/1"/>
  <mebase:has-license rdf:resource="http://www.myexperiment.org/licenses/2" />
  <mebase:last-edited-by rdf:resource="http://www.myexperiment.org/users/22" />
  <mevd:viewed rdf:datatype="&xsd;nonNegativeInteger">874</mevd:viewed>
  <mevd:downloaded rdf:datatype="&xsd;nonNegativeInteger">918</mevd:downloaded>
  <mebase:has-policy rdf:resource="http://www.myexperiment.org/workflows/173/
  policies/387" />
  <mecomp:executes-dataflow rdf:resource="http://www.myexperiment.org/workflows/173/
  versions/2#dataflow" />
  <meannot:has-comment rdf:resource="http://www.myexperiment.org/workflows/173/
  comments/327"/>
  <meannot:has-comment rdf:resource="http://www.myexperiment.org/workflows/173/
  comments/357"/>
  <meannot:has-comment rdf:resource="http://www.myexperiment.org/workflows/173/
  comments/358"/>
```

```
    <meannot:has-comment rdf:resource="http://www.myexperiment.org/workflows/173/
    comments/360"/>
    <meannot:has-comment rdf:resource="http://www.myexperiment.org/workflows/173/
    comments/366"/>
    <meannot:has-comment rdf:resource="http://www.myexperiment.org/workflows/173/
    comments/387"/>
    <meannot:has-rating rdf:resource="http://www.myexperiment.org/workflows/173/
    ratings/109"/>
    <meannot:has-rating rdf:resource="http://www.myexperiment.org/workflows/173/
    ratings/124"/>
    <meannot:has-tagging rdf:resource="http://www.myexperiment.org/tags/555/taggings
    /975"/>
    <meannot:has-tagging rdf:resource="http://www.myexperiment.org/tags/450/taggings
    /976"/>
    <meannot:has-tagging rdf:resource="http://www.myexperiment.org/tags/760/taggings
    /1024"/>
  </mecontrib:Workflow>

  <mepack:Pack rdf:about="http://www.myexperiment.org/packs/187">
    <foaf:homepage rdf:resource="http://www.myexperiment.org/packs/187.html"/>
    <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/packs/187.rdf"/>
    <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/packs/187.xml"/>
    <ore:isDescribedBy rdf:resource="http://www.myexperiment.org/packs/187.rdf" />
    <ore:aggregates rdf:resource="http://www.myexperiment.org/workflows/173"/>
    <ore:aggregates rdf:resource="http://www.myexperiment.org/files/141"/>
    <ore:aggregates rdf:resource="http://www.myexperiment.org/packs/147"/>
    <ore:aggregates rdf:resource="http://www.myexperiment.org/files/16"/>
    <ore:aggregates rdf:resource="urn:uuid:77a2b953-3b32-5b9f-bfb6-3c0b09a54669"/>
    <mepack:has-entry rdf:resource="http://www.myexperiment.org/packs/187/
    local_pack_entries/800"/>
    <mepack:has-entry rdf:resource="http://www.myexperiment.org/packs/187/
    local_pack_entries/801"/>
    <mepack:has-entry rdf:resource="http://www.myexperiment.org/packs/187/
    local_pack_entries/802"/>
    <mepack:has-entry rdf:resource="http://www.myexperiment.org/packs/187/
    local_pack_entries/803"/>
    <mepack:has-entry rdf:resource="http://www.myexperiment.org/packs/187/
    relationship_entries/1"/>
    <sioc:has_owner rdf:resource="http://www.myexperiment.org/users/22" />
    <dcterms:title rdf:datatype="&xsd;string">Test pack for relationships</
    dcterms:title>
    <dcterms:description rdf:datatype="&xsd;string">&lt;p&gt;This is my private pack
    for testing out the pack relationships.&lt;/p&gt;</dcterms:description>
    <dcterms:created rdf:datatype="&xsd;dateTime">2011-06-03T12:48:09Z</
    dcterms:created>
    <dcterms:modified rdf:datatype="&xsd;dateTime">2011-06-03T12:48:09Z</
    dcterms:modified>
    <mebase:has-policy rdf:resource="http://www.myexperiment.org/packs/187/policies
    /3078" />
  </mepack:Pack>

  <mebase:User rdf:about="http://www.myexperiment.org/users/22">
    <foaf:homepage rdf:resource="http://www.myexperiment.org/users/22.html"/>
    <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/users/22.rdf"/>
    <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/users/22.xml"/>
    <dcterms:created rdf:datatype="&xsd;dateTime">2007-07-23T09:14:48Z</
    dcterms:created>
    <dcterms:modified rdf:datatype="&xsd;dateTime">2011-07-07T10:34:45Z</
    dcterms:modified>
```

```
<sioc:name rdf:datatype="&xsd;string">Don Cruickshank</sioc:name>
<foaf:name rdf:datatype="&xsd;string">Don Cruickshank</foaf:name>
<sioc:avatar rdf:resource="http://www.myexperiment.org/images/avatar.png" />
<foaf:based_near rdf:datatype="&xsd;string">Southampton</foaf:based_near>
<mebase:country rdf:datatype="&xsd;string">United Kingdom</mebase:country>
<dbpedia:residence rdf:resource="http://dbpedia.org/resource/Southampton"/>
<dbpedia:residence rdf:resource="http://dbpedia.org/resource/United_Kingdom"/>
<foaf:homepage rdf:resource="http://www.ecs.soton.ac.uk/people/dgc" />
<mebase:last-seen-at rdf:datatype="&xsd;dateTime">2011-07-07T10:24:18Z</
mebase:last-seen-at>
<mebase:activated-at rdf:datatype="&xsd;dateTime">2007-07-23T09:14:48Z</
mebase:activated-at>
<mebase:receive-notifications rdf:datatype="&xsd;boolean">1</mebase:receive-
notifications>
<foaf:mbox rdf:resource="mailto:dgc@ecs.soton.ac.uk" />
<foaf:mbox_sha1sum rdf:datatype="&xsd;string">
a1d03dcfb2c4bc3ce9c50d890ccf380ec80aa8ee</foaf:mbox_sha1sum>
<mebase:has-friendship rdf:resource="http://www.myexperiment.org/users/22/
friendships/33"/>
<mebase:has-friendship rdf:resource="http://www.myexperiment.org/users/22/
friendships/34"/>
<mebase:has-friendship rdf:resource="http://www.myexperiment.org/users/22/
friendships/35"/>
<mebase:has-friendship rdf:resource="http://www.myexperiment.org/users/22/
friendships/36"/>
<mebase:has-friendship rdf:resource="http://www.myexperiment.org/users/25/
friendships/39"/>
<mebase:has-friendship rdf:resource="http://www.myexperiment.org/users/26/
friendships/51"/>
<mebase:has-friendship rdf:resource="http://www.myexperiment.org/users/43/
friendships/93"/>
<mebase:has-friendship rdf:resource="http://www.myexperiment.org/users/6/
friendships/139"/>
<mebase:has-friendship rdf:resource="http://www.myexperiment.org/users/70/
friendships/241"/>
<mebase:has-friendship rdf:resource="http://www.myexperiment.org/users/22/
friendships/267"/>
<mebase:has-friendship rdf:resource="http://www.myexperiment.org/users/22/
friendships/268"/>
<mebase:has-friendship rdf:resource="http://www.myexperiment.org/users/18/
friendships/290"/>
<mebase:has-friendship rdf:resource="http://www.myexperiment.org/users/162/
friendships/404"/>
<mebase:has-friendship rdf:resource="http://www.myexperiment.org/users/15/
friendships/446"/>
<mebase:has-friendship rdf:resource="http://www.myexperiment.org/users/22/
friendships/453"/>
<mebase:has-friendship rdf:resource="http://www.myexperiment.org/users/164/
friendships/470"/>
<mebase:has-friendship rdf:resource="http://www.myexperiment.org/users/2213/
friendships/754"/>
<mebase:has-friendship rdf:resource="http://www.myexperiment.org/users/2705/
friendships/780"/>
<mebase:has-friendship rdf:resource="http://www.myexperiment.org/users/22/
friendships/805"/>
<mebase:has-friendship rdf:resource="http://www.myexperiment.org/users/22/
friendships/806"/>
<mebase:has-friendship rdf:resource="http://www.myexperiment.org/users/437/
friendships/983"/>
```

```
    <mebase:has-membership rdf:resource="http://www.myexperiment.org/users/22/
    memberships/6"/>
    <mebase:has-membership rdf:resource="http://www.myexperiment.org/users/22/
    memberships/267"/>
    <mebase:has-membership rdf:resource="http://www.myexperiment.org/users/22/
    memberships/360"/>
    <mebase:has-membership rdf:resource="http://www.myexperiment.org/users/22/
    memberships/543"/>
    <mebase:has-membership rdf:resource="http://www.myexperiment.org/users/22/
    memberships/590"/>
    <mebase:has-membership rdf:resource="http://www.myexperiment.org/users/22/
    memberships/824"/>
    <mebase:has-favourite rdf:resource="http://www.myexperiment.org/users/22/
    favourites/78"/>
    <mebase:has-favourite rdf:resource="http://www.myexperiment.org/users/22/
    favourites/148"/>
  </mebase:User>

</rdf:RDF>
```

Listing A.13: Example of a Relationship Entry

A.3.3 Example of a User-defined Ontology

`http://rdf.myexperiment.org/examples/ontologies` as of 10/10/2011.

```
<?xml version="1.0" encoding="UTF-8" ?>

<!DOCTYPE rdf:RDF [
 <!ENTITY rdf 'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
 <!ENTITY rdfs 'http://www.w3.org/2000/01/rdf-schema#'>
 <!ENTITY owl 'http://www.w3.org/2002/07/owl#'>
 <!ENTITY dc 'http://purl.org/dc/elements/1.1/'>
 <!ENTITY dcterms 'http://purl.org/dc/terms/'>
 <!ENTITY xsd 'http://www.w3.org/2001/XMLSchema#'>
 <!ENTITY pack 'http://www.myexperiment.org/2010/pack/'>
]>

<rdf:RDF xmlns:rdf      ="&rdf;"
        xmlns:rdfs      ="&rdfs;"
        xmlns:owl       ="&owl;"
        xmlns:dc        ="&dc;"
        xmlns:dcterms   ="&dcterms;"
        xmlns:xsd       ="&xsd;"
        xmlns:pack      ="&pack;"
>

  <owl:Ontology rdf:about="http://www.myexperiment.org/2010/pack">
    <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/2010/pack.rdf"/>
    <rdfs:label rdf:datatype="&xsd;string">myExperiment pack ontology</rdfs:label>
    <rdfs:comment rdf:datatype="&xsd;string">This is an ontology for expressing the
    relationships between pack items in myExperiment.</rdfs:comment>
    <dcterms:created rdf:datatype="&xsd;dateTime">2011-06-03T18:04:56Z</
    dcterms:created>
    <dcterms:modified rdf:datatype="&xsd;dateTime">2011-06-03T18:04:56Z</
    dcterms:modified>
    <dc:language rdf:datatype="&xsd;string">en</dc:language>
```

```
  <dc:publisher rdf:resource="http://www.myexperiment.org"/>
  <dc:format rdf:datatype="&xsd;string">rdf/xml</dc:format>
</owl:Ontology>

<owl:ObjectProperty rdf:about="http://www.myexperiment.org/2010/pack/inputDataTo">
  <rdfs:label rdf:datatype="&xsd;string">inputDataTo</rdfs:label>
  <rdfs:comment rdf:datatype="&xsd;string">This concept is used to indicate that the
   subject is an input to the object.</rdfs:comment>
  <rdfs:isDefinedBy rdf:resource="http://www.myexperiment.org/2010/pack" />
  <dcterms:created rdf:datatype="&xsd;dateTime">2011-06-03T18:04:56Z</
  dcterms:created>
  <dcterms:modified rdf:datatype="&xsd;dateTime">2011-06-03T18:04:56Z</
  dcterms:modified>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="http://www.myexperiment.org/2010/pack/outputDataFrom"
  >
  <rdfs:label rdf:datatype="&xsd;string">outputDataFrom</rdfs:label>
  <rdfs:comment rdf:datatype="&xsd;string">This concept is used to indicate that the
   subject is an output from the object.</rdfs:comment>
  <rdfs:isDefinedBy rdf:resource="http://www.myexperiment.org/2010/pack" />
  <dcterms:created rdf:datatype="&xsd;dateTime">2011-06-03T18:04:56Z</
  dcterms:created>
  <dcterms:modified rdf:datatype="&xsd;dateTime">2011-06-03T18:04:56Z</
  dcterms:modified>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="http://www.myexperiment.org/2010/pack/
  exampleInputDataTo">
  <rdfs:label rdf:datatype="&xsd;string">exampleInputDataTo</rdfs:label>
  <rdfs:comment rdf:datatype="&xsd;string">This concept is used to indicate that the
   subject is an example input to the object.</rdfs:comment>
  <rdfs:isDefinedBy rdf:resource="http://www.myexperiment.org/2010/pack" />
  <dcterms:created rdf:datatype="&xsd;dateTime">2011-06-03T18:04:56Z</
  dcterms:created>
  <dcterms:modified rdf:datatype="&xsd;dateTime">2011-06-03T18:04:56Z</
  dcterms:modified>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="http://www.myexperiment.org/2010/pack/
  exampleOutputDataFrom">
  <rdfs:label rdf:datatype="&xsd;string">exampleOutputDataFrom</rdfs:label>
  <rdfs:comment rdf:datatype="&xsd;string">This concept is used to indicate that the
   subject is an example output from the object.</rdfs:comment>
  <rdfs:isDefinedBy rdf:resource="http://www.myexperiment.org/2010/pack" />
  <dcterms:created rdf:datatype="&xsd;dateTime">2011-06-03T18:04:56Z</
  dcterms:created>
  <dcterms:modified rdf:datatype="&xsd;dateTime">2011-06-03T18:04:56Z</
  dcterms:modified>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="http://www.myexperiment.org/2010/pack/presentAt">
  <rdfs:label rdf:datatype="&xsd;string">presentAt</rdfs:label>
  <rdfs:comment rdf:datatype="&xsd;string">This concept is used to indicate that the
   subject was presented at the object.</rdfs:comment>
  <rdfs:isDefinedBy rdf:resource="http://www.myexperiment.org/2010/pack" />
  <dcterms:created rdf:datatype="&xsd;dateTime">2011-06-28T17:22:20Z</
  dcterms:created>
```

```
    <dcterms:modified rdf:datatype="&xsd;dateTime">2011-06-28T17:22:20Z</
    dcterms:modified>
  </owl:ObjectProperty>

</rdf:RDF>
```

Listing A.14: Example of User-defined Ontology

A.4 myExperiment's VoID Specification

`http://rdf.myexperiment.org/void.rdf` as of 16/10/2011.

```
<?xml version="1.0" encoding="UTF-8" ?>

<rdf:RDF xmlns="http://www.myexperiment.org/void.ttl/#"
         xmlns:dcterms="http://purl.org/dc/terms/"
         xmlns:foaf="http://xmlns.com/foaf/0.1/"
         xmlns:log="http://www.w3.org/2000/10/swap/log#"
         xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
               xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
         xmlns:void="http://rdfs.org/ns/void#"
               xmlns:scovo="http://purl.org/NET/scovo#">

  <void:Dataset rdf:about="http://rdf.myexperiment.org/void.rdf#myExpDataset">
    <dcterms:title rdf:datatype="http://www.w3.org/2001/XMLSchema#string">myExperiment
    </dcterms:title>
    <dcterms:description rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
    myExperiment's Public Dataset</dcterms:description>
    <dcterms:source rdf:resource="http://www.myexperiment.org/"/>
    <dcterms:license rdf:resource="http://www.opendatacommons.org/licenses/pddl
    /1.0/"/>
    <dcterms:publisher rdf:resource="http://www.myexperiment.org/"/>
    <dcterms:subject rdf:resource="http://dbpedia.org/resource/Annotation"/>
    <dcterms:subject rdf:resource="http://dbpedia.org/resource/
    Content_management_system"/>
    <dcterms:subject rdf:resource="http://dbpedia.org/resource/Discourse"/>
    <dcterms:subject rdf:resource="http://dbpedia.org/resource/Social_network_service
    "/>
    <dcterms:subject rdf:resource="http://dbpedia.org/resource/Workflow"/>
    <void:dataDump rdf:resource="http://rdf.myexperiment.org/myexperiment.rdf.gz"/>
    <void:statItem>
      <scovo:Item>
        <scovo:dimension rdf:resource="http://rdfs.org/ns/void#noOfTriples"/>
        <rdf:value rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger
    ">980719</rdf:value>
      </scovo:Item>
    </void:statItem>
    <void:exampleResource rdf:resource="http://www.myexperiment.org/comments/2"/>
    <void:exampleResource rdf:resource="http://www.myexperiment.org/files/4"/>
    <void:exampleResource rdf:resource="http://www.myexperiment.org/groups/9"/>
    <void:exampleResource rdf:resource="http://www.myexperiment.org/packs/3"/>
    <void:exampleResource rdf:resource="http://www.myexperiment.org/taggings/2"/>
    <void:exampleResource rdf:resource="http://www.myexperiment.org/tags/39"/>
    <void:exampleResource rdf:resource="http://www.myexperiment.org/users/26"/>
    <void:exampleResource rdf:resource="http://www.myexperiment.org/workflows/12"/>
```

```
  <void:sparqlEndpoint rdf:resource="http://rdf.myexperiment.org/sparql"/>
  <void:vocabulary rdf:resource="http://purl.org/dc/terms/"/>
  <void:vocabulary rdf:resource="http://rdf.myexperiment.org/ontologies/annotations
  /"/>
  <void:vocabulary rdf:resource="http://rdf.myexperiment.org/ontologies/
  attrib_credit/"/>
  <void:vocabulary rdf:resource="http://rdf.myexperiment.org/ontologies/base/"/>
  <void:vocabulary rdf:resource="http://rdf.myexperiment.org/ontologies/components
  /"/>
  <void:vocabulary rdf:resource="http://rdf.myexperiment.org/ontologies/
  contributions/"/>
  <void:vocabulary rdf:resource="http://rdf.myexperiment.org/ontologies/experiments
  /"/>
  <void:vocabulary rdf:resource="http://rdf.myexperiment.org/ontologies/packs/"/>
  <void:vocabulary rdf:resource="http://rdf.myexperiment.org/ontologies/snarm/"/>
  <void:vocabulary rdf:resource="http://rdf.myexperiment.org/ontologies/specific/"/>
  <void:vocabulary rdf:resource="http://rdf.myexperiment.org/ontologies/
  viewings_downloads/"/>
  <void:vocabulary rdf:resource="http://rdfs.org/sioc/ns#"/>
  <void:vocabulary rdf:resource="http://web.resource.org/cc/"/>
  <void:vocabulary rdf:resource="http://www.openarchives.org/ore/terms/"/>
  <void:vocabulary rdf:resource="http://xmlns.com/foaf/0.1/"/>
  <foaf:homepage rdf:resource="http://www.myexperiment.org/"/>
</void:Dataset>

<void:Linkset rdf:about="http://rdf.myexperiment.org/linksets/myExperiment-DBPedia">
  <void:subjectsTarget rdf:resource="http://rdf.myexperiment.org/void.rdf#
  myexpDataset"/>
  <void:objectsTarget rdf:resource="http://dbpedia.org/void.ttl#Geonames"/>
  <void:dataDump rdf:resource="http://rdf.myexperiment.org/linksets/myExperiment-
  DBPedia.nt"/>
  <void:linkPredicate rdf:resource="http://dbpedia.org/ontology/residence"/>
  <void:statItem>
    <scovo:Item>
      <scovo:dimension rdf:resource="http://rdfs.org/ns/void#noOfTriples"/>
      <rdf:value rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger
  ">2165</rdf:value>
    </scovo:Item>
  </void:statItem>
</void:Linkset>

<void:Linkset rdf:about="http://rdf.myexperiment.org/linksets/myExperiment-DBLP">
  <void:subjectsTarget rdf:resource="http://rdf.myexperiment.org/void.rdf#
  myexpDataset"/>
  <void:objectsTarget rdf:resource="http://dblp.rkbexplorer.com/id/void"/>
  <void:dataDump rdf:resource="http://rdf.myexperiment.org/linksets/myExperiment-
  DBLP.nt"/>
  <void:linkPredicate rdf:resource="http://www.openarchives.org/ore/terms/aggregates
  "/>
  <void:linkPredicate rdf:resource="http://www.openarchives.org/ore/terms/proxyFor
  "/>
  <void:statItem>
    <scovo:Item>
      <scovo:dimension rdf:resource="http://rdfs.org/ns/void#noOfTriples"/>
      <rdf:value rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger
  ">348</rdf:value>
    </scovo:Item>
  </void:statItem>
</void:Linkset>
```

```
  <void:Linkset rdf:about="http://rdf.myexperiment.org/linksets/myExperiment-ECS-
    EPrints">
    <void:subjectsTarget rdf:resource="http://rdf.myexperiment.org/void.rdf#
    myexpDataset"/>
    <void:objectsTarget rdf:resource="http://eprints.rkbexplorer.com/id/void"/>
    <void:dataDump rdf:resource="http://rdf.myexperiment.org/linksets/myExperiment-ECS
    -EPrints.nt"/>
    <void:linkPredicate rdf:resource="http://www.openarchives.org/ore/terms/aggregates
    "/>
    <void:linkPredicate rdf:resource="http://www.openarchives.org/ore/terms/proxyFor
    "/>
    <void:statItem>
      <scovo:Item>
        <scovo:dimension rdf:resource="http://rdfs.org/ns/void#noOfTriples"/>
        <rdf:value rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger
    ">50</rdf:value>
      </scovo:Item>
    </void:statItem>
  </void:Linkset>

</rdf:RDF>
```

Listing A.15: myExperiment's VoID Specification

A.5 Example Response for RDF Pack Request

`http://www.myexperiment.org/packs/8.rdf` as of 28/11/2010.

```
<?xml version="1.0" encoding="UTF-8" ?>

<!DOCTYPE rdf:RDF [
 <!ENTITY mebase 'http://rdf.myexperiment.org/ontologies/base/'>
 <!ENTITY meac 'http://rdf.myexperiment.org/ontologies/attrib_credit/'>
 <!ENTITY meannot 'http://rdf.myexperiment.org/ontologies/annotations/'>
 <!ENTITY mepack 'http://rdf.myexperiment.org/ontologies/packs/'>
 <!ENTITY meexp 'http://rdf.myexperiment.org/ontologies/experiments/'>
 <!ENTITY mecontrib 'http://rdf.myexperiment.org/ontologies/contributions/'>
 <!ENTITY mevd 'http://rdf.myexperiment.org/ontologies/viewings_downloads/'>
 <!ENTITY mecomp 'http://rdf.myexperiment.org/ontologies/components/'>
 <!ENTITY mespec 'http://rdf.myexperiment.org/ontologies/specific/'>
 <!ENTITY rdf 'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
 <!ENTITY rdfs 'http://www.w3.org/2000/01/rdf-schema#'>
 <!ENTITY owl 'http://www.w3.org/2002/07/owl#'>
 <!ENTITY dc 'http://purl.org/dc/elements/1.1/'>
 <!ENTITY dcterms 'http://purl.org/dc/terms/'>
 <!ENTITY cc 'http://web.resource.org/cc/'>
 <!ENTITY foaf 'http://xmlns.com/foaf/0.1/'>
 <!ENTITY sioc 'http://rdfs.org/sioc/ns#'>
 <!ENTITY ore 'http://www.openarchives.org/ore/terms/'>
 <!ENTITY dbpedia 'http://dbpedia.org/ontology/'>
 <!ENTITY snarm 'http://rdf.myexperiment.org/ontologies/snarm/'>
 <!ENTITY xsd 'http://www.w3.org/2001/XMLSchema#'>
]>

<rdf:RDF xmlns:mebase   ="&mebase;"
```

```
        xmlns:meac      ="&meac;"
        xmlns:meannot   ="&meannot;"
        xmlns:mepack    ="&mepack;"
        xmlns:meexp     ="&meexp;"
        xmlns:mecontrib ="&mecontrib;"
        xmlns:mevd      ="&mevd;"
        xmlns:mecomp    ="&mecomp;"
        xmlns:mespec    ="&mespec;"
        xmlns:rdf       ="&rdf;"
        xmlns:rdfs      ="&rdfs;"
        xmlns:owl       ="&owl;"
        xmlns:dc        ="&dc;"
        xmlns:dcterms   ="&dcterms;"
        xmlns:cc        ="&cc;"
        xmlns:foaf      ="&foaf;"
        xmlns:sioc      ="&sioc;"
        xmlns:ore       ="&ore;"
        xmlns:dbpedia   ="&dbpedia;"
        xmlns:snarm     ="&snarm;"
        xmlns:xsd       ="&xsd;"
>
  <rdf:Description rdf:about="http://www.myexperiment.org/packs/8.rdf">
    <foaf:primaryTopic rdf:resource="http://www.myexperiment.org/packs/8"/>
  </rdf:Description>

  <mepack:Pack rdf:about="http://www.myexperiment.org/packs/8">
    <foaf:homepage rdf:resource="http://www.myexperiment.org/packs/8.html"/>
    <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/packs/8.rdf"/>
    <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/packs/8.xml"/>
    <ore:aggregates rdf:resource="http://www.myexperiment.org/workflows/69"/>
    <ore:aggregates rdf:resource="http://www.myexperiment.org/files/4"/>
    <ore:aggregates rdf:resource="http://www.myexperiment.org/packs/9"/>
    <ore:aggregates rdf:resource="http://www.mygrid.org.uk/ontology"/>
    <mepack:has-pack-entry rdf:resource="http://www.myexperiment.org/packs/8/
    local_pack_entries/74"/>
    <mepack:has-pack-entry rdf:resource="http://www.myexperiment.org/packs/8/
    local_pack_entries/75"/>
    <mepack:has-pack-entry rdf:resource="http://www.myexperiment.org/packs/8/
    local_pack_entries/78"/>
    <mepack:has-pack-entry rdf:resource="http://www.myexperiment.org/packs/8/
    remote_pack_entries/13"/>
    <sioc:has_owner rdf:resource="http://www.myexperiment.org/users/61" />
    <dcterms:title rdf:datatype="&xsd;string">Testing123</dcterms:title>
    <dcterms:description rdf:datatype="&xsd;string">&lt;p&gt;A pack that has packs&lt
    ;/p&gt;</dcterms:description>
    <dcterms:created rdf:datatype="&xsd;dateTime">2008-07-12T08:33:53Z</
    dcterms:created>
    <dcterms:modified rdf:datatype="&xsd;dateTime">2008-07-12T08:33:53Z</
    dcterms:modified>
    <mevd:viewed rdf:datatype="&xsd;nonNegativeInteger">148</mevd:viewed>
    <mevd:downloaded rdf:datatype="&xsd;nonNegativeInteger">55</mevd:downloaded>
    <mebase:has-policy rdf:resource="http://www.myexperiment.org/packs/8/policies/493"
     />
  </mepack:Pack>

  <mecontrib:Workflow rdf:about="http://www.myexperiment.org/workflows/69">
    <foaf:homepage rdf:resource="http://www.myexperiment.org/workflows/69.html"/>
    <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/workflows/69.rdf"/>
    <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/workflows/69.xml"/>
```

```
    <dcterms:title rdf:datatype="&xsd;string">3</dcterms:title>
    <mebase:has-content-type rdf:resource="http://www.myexperiment.org/content_types/1
    " />
    <sioc:has_owner rdf:resource="http://www.myexperiment.org/users/61" />
    <dcterms:created rdf:datatype="&xsd;dateTime">2007-11-06T17:10:16Z</
    dcterms:created>
    <dcterms:modified rdf:datatype="&xsd;dateTime">2007-11-06T17:10:16Z</
    dcterms:modified>
    <mebase:filename rdf:datatype="&xsd;string">3_9479.xml</mebase:filename>
    <mebase:content-url rdf:resource="http://www.myexperiment.org/workflows/69/
    download/3_9479.xml" />
    <mecontrib:preview rdf:resource="http://www.myexperiment.org/workflow/image/69/3
    _9479.png" />
    <mecontrib:thumbnail rdf:resource="http://www.myexperiment.org/workflow/image/69/
    thumb/3_9479.png" />
    <mecontrib:thumbnail-big rdf:resource="http://www.myexperiment.org/workflow/image
    /69/medium/3_9479.png" />
    <mecontrib:svg rdf:resource="http://www.myexperiment.org/workflow/svg/69/3_9479.
    svg" />
    <mebase:has-current-version rdf:resource="http://www.myexperiment.org/workflows
    /69/versions/1" />
    <mebase:has-license rdf:resource="http://www.myexperiment.org/licenses/2" />
    <mebase:last-edited-by rdf:resource="http://www.myexperiment.org/users/61" />
    <mevd:viewed rdf:datatype="&xsd;nonNegativeInteger">315</mevd:viewed>
    <mebase:has-policy rdf:resource="http://www.myexperiment.org/workflows/69/policies
    /136" />
    <meannot:has-tagging rdf:resource="http://www.myexperiment.org/tags/551/taggings
    /200"/>
  </mecontrib:Workflow>

  <mecontrib:File rdf:about="http://www.myexperiment.org/files/4">
    <foaf:homepage rdf:resource="http://www.myexperiment.org/files/4.html"/>
    <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/files/4.rdf"/>
    <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/files/4.xml"/>
    <mebase:content-url rdf:resource="http://www.myexperiment.org/blobs/4/download/3.
    png" />
    <mebase:filename rdf:datatype="&xsd;string">3.png</mebase:filename>
    <sioc:has_owner rdf:resource="http://www.myexperiment.org/users/61" />
    <mebase:has-content-type rdf:resource="http://www.myexperiment.org/content_types
    /18" />
    <mebase:has-license rdf:resource="http://www.myexperiment.org/licenses/1" />
    <dcterms:created rdf:datatype="&xsd;dateTime">2007-11-06T17:12:59Z</
    dcterms:created>
    <dcterms:modified rdf:datatype="&xsd;dateTime">2008-10-09T14:04:26Z</
    dcterms:modified>
    <dcterms:title rdf:datatype="&xsd;string">Picture 3</dcterms:title>
    <mevd:viewed rdf:datatype="&xsd;nonNegativeInteger">122</mevd:viewed>
    <mevd:downloaded rdf:datatype="&xsd;nonNegativeInteger">72</mevd:downloaded>
    <mebase:has-policy rdf:resource="http://www.myexperiment.org/files/4/policies/902"
     />
  </mecontrib:File>

  <mepack:Pack rdf:about="http://www.myexperiment.org/packs/9">
    <foaf:homepage rdf:resource="http://www.myexperiment.org/packs/9.html"/>
    <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/packs/9.rdf"/>
    <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/packs/9.xml"/>
    <ore:aggregates rdf:resource="http://www.myexperiment.org/workflows/34"/>
    <mepack:has-pack-entry rdf:resource="http://www.myexperiment.org/packs/9/
    local_pack_entries/77"/>
```

```
    <sioc:has_owner rdf:resource="http://www.myexperiment.org/users/61" />
    <dcterms:title rdf:datatype="&xsd;string">Testing456</dcterms:title>
    <dcterms:description rdf:datatype="&xsd;string">&lt;p&gt;subpack 1&lt;/p&gt;</
    dcterms:description>
    <dcterms:created rdf:datatype="&xsd;dateTime">2008-07-12T08:42:06Z</
    dcterms:created>
    <dcterms:modified rdf:datatype="&xsd;dateTime">2008-07-12T08:42:20Z</
    dcterms:modified>
    <mevd:viewed rdf:datatype="&xsd;nonNegativeInteger">151</mevd:viewed>
    <mebase:has-policy rdf:resource="http://www.myexperiment.org/packs/9/policies/494"
     />
  </mepack:Pack>

  <mepack:LocalPackEntry rdf:about="http://www.myexperiment.org/packs/8/
    local_pack_entries/74">
    <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/packs/8/
    local_pack_entries/74.rdf"/>
    <ore:proxyIn rdf:resource="http://www.myexperiment.org/packs/8" />
    <ore:proxyFor rdf:resource="http://www.myexperiment.org/workflows/69" />
    <sioc:has_owner rdf:resource="http://www.myexperiment.org/users/61" />
    <dcterms:created rdf:datatype="&xsd;dateTime">2008-07-12T08:34:14Z</
    dcterms:created>
    <dcterms:modified rdf:datatype="&xsd;dateTime">2008-07-12T08:34:14Z</
    dcterms:modified>
  </mepack:LocalPackEntry>

  <mepack:LocalPackEntry rdf:about="http://www.myexperiment.org/packs/8/
    local_pack_entries/75">
    <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/packs/8/
    local_pack_entries/75.rdf"/>
    <ore:proxyIn rdf:resource="http://www.myexperiment.org/packs/8" />
    <ore:proxyFor rdf:resource="http://www.myexperiment.org/files/4" />
    <sioc:has_owner rdf:resource="http://www.myexperiment.org/users/61" />
    <dcterms:created rdf:datatype="&xsd;dateTime">2008-07-12T08:36:03Z</
    dcterms:created>
    <dcterms:modified rdf:datatype="&xsd;dateTime">2008-07-12T08:36:03Z</
    dcterms:modified>
  </mepack:LocalPackEntry>

  <mepack:LocalPackEntry rdf:about="http://www.myexperiment.org/packs/8/
    local_pack_entries/78">
    <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/packs/8/
    local_pack_entries/78.rdf"/>
    <ore:proxyIn rdf:resource="http://www.myexperiment.org/packs/8" />
    <ore:proxyFor rdf:resource="http://www.myexperiment.org/packs/9" />
    <sioc:has_owner rdf:resource="http://www.myexperiment.org/users/61" />
    <dcterms:created rdf:datatype="&xsd;dateTime">2008-07-12T08:44:11Z</
    dcterms:created>
    <dcterms:modified rdf:datatype="&xsd;dateTime">2008-07-12T08:44:11Z</
    dcterms:modified>
  </mepack:LocalPackEntry>

  <mepack:RemotePackEntry rdf:about="http://www.myexperiment.org/packs/8/
    remote_pack_entries/13">
    <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/packs/8/
    remote_pack_entries/13.rdf"/>
    <ore:proxyIn rdf:resource="http://www.myexperiment.org/packs/8" />
    <dcterms:title rdf:datatype="&xsd;string">Link</dcterms:title>
    <ore:proxyFor>
```

```
        <rdf:Description rdf:about="http://www.mygrid.org.uk/ontology"/>
    </ore:proxyFor>
    <dcterms:description rdf:datatype="&xsd;string">the myGrid ontology</
    dcterms:description>
    <sioc:has_owner rdf:resource="http://www.myexperiment.org/users/61" />
    <dcterms:created rdf:datatype="&xsd;dateTime">2008-07-12T08:35:01Z</
    dcterms:created>
    <dcterms:modified rdf:datatype="&xsd;dateTime">2008-07-12T08:35:01Z</
    dcterms:modified>
  </mepack:RemotePackEntry>

  <mebase:User rdf:about="http://www.myexperiment.org/users/61">
    <foaf:homepage rdf:resource="http://www.myexperiment.org/users/61.html"/>
    <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/users/61.rdf"/>
    <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/users/61.xml"/>
    <dcterms:created rdf:datatype="&xsd;dateTime">2007-07-31T17:23:00Z</
    dcterms:created>
    <dcterms:modified rdf:datatype="&xsd;dateTime">2009-07-27T09:48:55Z</
    dcterms:modified>
    <sioc:name rdf:datatype="&xsd;string">Antoon Goderis</sioc:name>
    <sioc:avatar rdf:resource="http://www.myexperiment.org/pictures/show/97?size=160
    x160.png" />
    <mebase:country rdf:datatype="&xsd;string">Belgium</mebase:country>
    <dbpedia:residence rdf:resource="http://dbpedia.org/resource/"/>
    <dbpedia:residence rdf:resource="http://dbpedia.org/resource/Belgium"/>
    <foaf:homepage rdf:resource="http://www.cs.man.ac.uk/~goderisa" />
    <mebase:last-seen-at rdf:datatype="&xsd;dateTime">2009-07-27T09:48:55Z</
    mebase:last-seen-at>
    <mebase:activated-at rdf:datatype="&xsd;dateTime">2007-07-31T17:23:00Z</
    mebase:activated-at>
    <mebase:receive-notifications rdf:datatype="&xsd;boolean">1</mebase:receive-
    notifications>
    <foaf:mbox_sha1sum rdf:datatype="&xsd;string">
    a4dc74aba4bbadddf643199c16bb1794232d3c86</foaf:mbox_sha1sum>
    <mebase:has-friendship rdf:resource="http://www.myexperiment.org/users/61/
    friendships/159"/>
    <mebase:has-friendship rdf:resource="http://www.myexperiment.org/users/61/
    friendships/160"/>
    <mebase:has-friendship rdf:resource="http://www.myexperiment.org/users/61/
    friendships/161"/>
    <mebase:has-friendship rdf:resource="http://www.myexperiment.org/users/61/
    friendships/162"/>
    <mebase:has-friendship rdf:resource="http://www.myexperiment.org/users/61/
    friendships/163"/>
    <mebase:has-friendship rdf:resource="http://www.myexperiment.org/users/61/
    friendships/164"/>
    <mebase:has-friendship rdf:resource="http://www.myexperiment.org/users/61/
    friendships/165"/>
    <mebase:has-friendship rdf:resource="http://www.myexperiment.org/users/61/
    friendships/166"/>
    <mebase:has-friendship rdf:resource="http://www.myexperiment.org/users/61/
    friendships/167"/>
    <mebase:has-friendship rdf:resource="http://www.myexperiment.org/users/61/
    friendships/168"/>
    <mebase:has-friendship rdf:resource="http://www.myexperiment.org/users/61/
    friendships/169"/>
    <mebase:has-friendship rdf:resource="http://www.myexperiment.org/users/61/
    friendships/170"/>
```

```
    <mebase:has-friendship rdf:resource="http://www.myexperiment.org/users/61/
    friendships/171"/>
    <mebase:has-friendship rdf:resource="http://www.myexperiment.org/users/61/
    friendships/182"/>
    <mebase:has-friendship rdf:resource="http://www.myexperiment.org/users/61/
    friendships/194"/>
    <mebase:has-friendship rdf:resource="http://www.myexperiment.org/users/162/
    friendships/257"/>
    <mebase:has-friendship rdf:resource="http://www.myexperiment.org/users/61/
    friendships/265"/>
    <mebase:has-friendship rdf:resource="http://www.myexperiment.org/users/70/
    friendships/269"/>
    <mebase:has-friendship rdf:resource="http://www.myexperiment.org/users/208/
    friendships/276"/>
    <mebase:has-friendship rdf:resource="http://www.myexperiment.org/users/221/
    friendships/301"/>
    <mebase:has-friendship rdf:resource="http://www.myexperiment.org/users/16/
    friendships/321"/>
    <mebase:has-friendship rdf:resource="http://www.myexperiment.org/users/61/
    friendships/337"/>
    <mebase:has-friendship rdf:resource="http://www.myexperiment.org/users/411/
    friendships/358"/>
    <mebase:has-friendship rdf:resource="http://www.myexperiment.org/users/61/
    friendships/394"/>
    <mebase:has-friendship rdf:resource="http://www.myexperiment.org/users/61/
    friendships/406"/>
    <mebase:has-friendship rdf:resource="http://www.myexperiment.org/users/625/
    friendships/407"/>
    <mebase:has-friendship rdf:resource="http://www.myexperiment.org/users/61/
    friendships/486"/>
    <mebase:has-friendship rdf:resource="http://www.myexperiment.org/users/1316/
    friendships/588"/>
    <mebase:has-membership rdf:resource="http://www.myexperiment.org/users/61/
    memberships/71"/>
    <mebase:has-membership rdf:resource="http://www.myexperiment.org/users/61/
    memberships/133"/>
  </mebase:User>

  <snarm:Policy rdf:about="http://www.myexperiment.org/packs/8/policies/493">
    <dcterms:hasFormat rdf:resource="http://www.myexperiment.org/packs/8/policies/493.
    rdf"/>
    <snarm:has-access>
      <snarm:RestrictedAccess>
        <snarm:has-accesser rdf:resource="http://www.myexperiment.org/users/61"/>
        <snarm:has-access-type rdf:resource="&mespec;View"/>
      </snarm:RestrictedAccess>
    </snarm:has-access>
    <snarm:has-access>
      <snarm:RestrictedAccess>
        <snarm:has-accesser rdf:resource="http://www.myexperiment.org/users/61"/>
        <snarm:has-access-type rdf:resource="&mespec;Download"/>
      </snarm:RestrictedAccess>
    </snarm:has-access>
    <snarm:has-access>
      <snarm:RestrictedAccess>
        <snarm:has-accesser rdf:resource="http://www.myexperiment.org/users/61"/>
        <snarm:has-access-type rdf:resource="&mespec;Edit"/>
      </snarm:RestrictedAccess>
    </snarm:has-access>
```

```
    <snarm:has-access rdf:resource="&mespec;PublicView"/>
    <snarm:has-access rdf:resource="&mespec;PublicDownload"/>
  </snarm:Policy>

</rdf:RDF>
```

Listing A.16: Example Response for RDF Pack Request

Appendix B

myExperiment Documentation

B.1 myExperiment Ontology Documentation

As of 16/10/2011.

my experiment **Ontology**

Submit Feedback/Bug Report

myExperiment is a collaborative environment where scientists can safely publish their workflows and experiment plans, share them with groups and find those of others. (Please see the myExperiment Wiki for more detailed information). This results in the myExperiment data model having three main underlying features:

- Content Management
- Social Networking
- Object Annotation

These features are the main focus of the myExperiment Ontology. Fig.1 gives a rough outline of how the main entities of myExperiment interact.

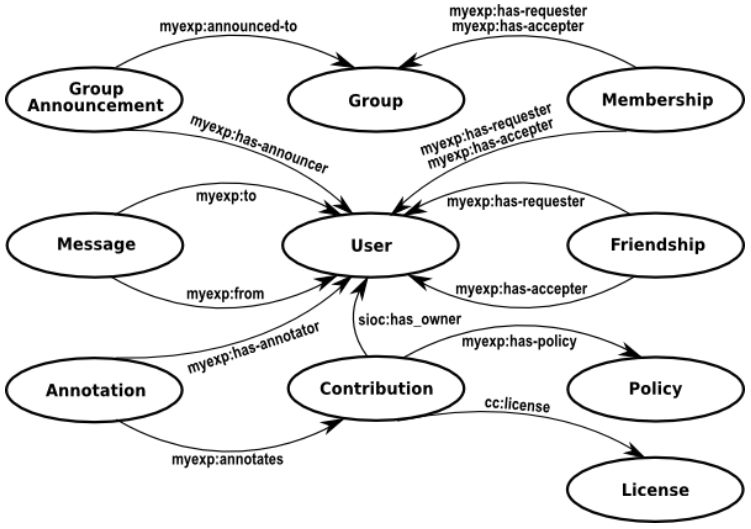


Fig.1 The Main Entities of the myExperiment

The myExperiment Ontology borrows terms from a number of well-known ontologies/schemas to foster reuse be simplifying ontology alignment:

- Dublin Core to provide common metadata properties.
- FOAF and SIOC to describe the social network.
- Creative Commons to define licenses for shared objects.
- OAI-ORE to support the aggregation of resources.
- DBPedia so users can specify where they are resident as a URI rather than a literal.

myExperiment borrows terms in a number of ways:

- Equivalence relationships for properties or classes, (e.g. Group is equivalent to SIOC's UserGroup).
- Subclass/subproperty relationships, (e.g. email property is a subproperty of FOAF's mbox).
- Required properties in class restrictions, (e.g. an uploaded Contribution must have a Creative Commons license property).

It was decided to construct the ontology in a modularized way to further encourage

reuse, where others could pick and choose the modules they wanted to use. The myExperiment ontology currently has 10 modules:

- **SNARM** or Simple Network Access Rights Management defines Policies for describing who can do what with certain objects.
- **Base** provides the base elements required by myExperiment for content management, social networking and object annotation.)
- **Attributions & Credit** allows contributions to give attribution to earlier contributions and pay credit to users and groups involved in their creation.
- **Annotations** provides the different types of annotations used in myExperiment.
- **Packs** facilitates the use of packs to aggregate contributions and remote urls together. (More Information)
- **Experiments** contains the classes required to create experiments and annotate them with jobs that have been or are scheduled to run. (More Information)
- **Viewings & Downloads** allows usage statistics on the viewings and downloads of contributions to be recorded.
- **Contributions** provides the different types of contributions used in myExperiment.
- **Components** allows components within workflows to be represented. (More Information)
- **Specific** provides classes and objects specific to myExperiment, including SNARM AccessTypes, CreativeCommons Licenses, the TavernaEnactor (a Taverna specific Runner) and the myExperiment AnonymousUser.

Fig.2 is a class hierarchy diagram for all the classes used in the myExperiment ontology, colour-coded by the module they belong to.

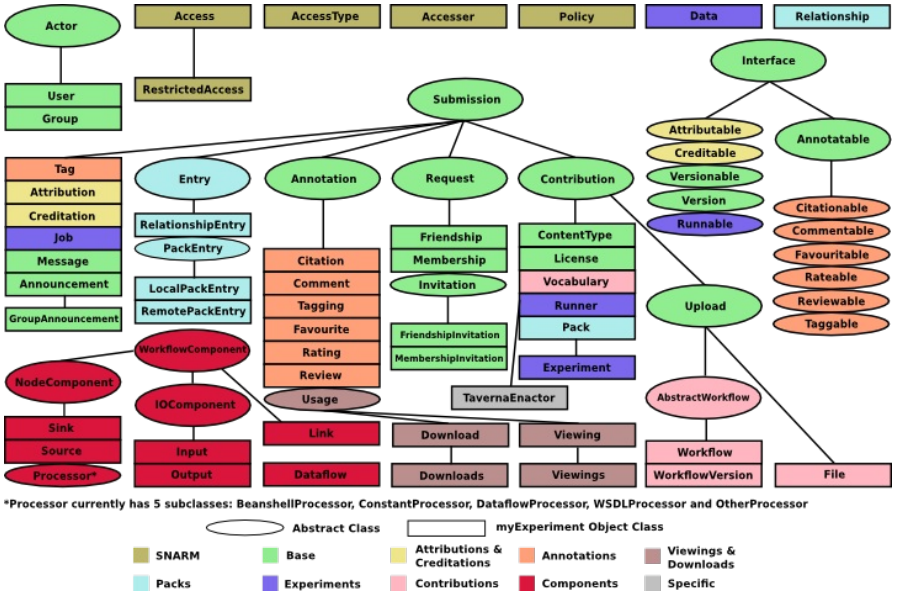


Fig.2 Class Hierarchy Diagram for Ontology Modules

For most of the classes described in Fig.2 RDF examples are available. Modules reuse terms from other modules as well as terms borrowed for the ontologies/schemas previously described. Fig.3 diagrams of how modules borrow from each other with each arrow going from the borrowed-from module/ontology/schema to the borrowing module. The "Base" module is built open by all the other modules (except SNARM). The "Specific" module is slightly different as it imports all the other modules below it. This

allows a single URI to be referred to when importing the myExperiment ontology set. Documentation explaining how all the ontology's classes and properties from the ontology can be used is available here.

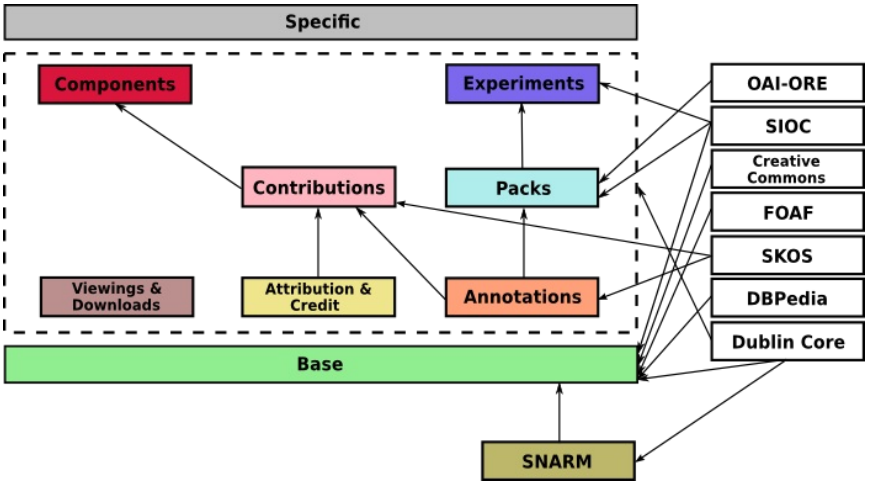


Fig.3 Ontology Modules Architecture

Content Management

The Base module contains two classes Contribution and its subclass Upload to represent content objects that can be managed. Contribution is a superclass to describe any content that can be contributed by a User. Upload is designed to only represent content that has been that is uploaded not created online. Upload therefore has additional properties for filename, file URL, file type and licensing.

During the course of the myExperiment project one of the key challenges was providing a flexible and user-friendly interface for user to define who can do what with their contributions. This led to the data model becoming slightly convuluted in this area. Therefore the SNARM ontology was defined to try to rationalise how this information is stored in RDF form.

SNARM uses policies which contain one or more Access objects. Access objects can be unrestricted or restricted to an Accesser that could be a single user, a single group or a more abstract concept such as all the friends or the content owner. The second component of an Access object is the AccessType they provide, e.g. view, edit, download, etc. These types are often quite particular and therefore Specific is used to store these along with abstract Accessers (e.g. Friends) and abstract/unrestricted Access objects (e.g. FriendsEdit, PublicView). A more detailed explanation of SNARM can be found here.

Social Networking

The Base module contains three abstract classes (Actor, Request and Invitation) for social network building. From these six object classes are derived:

- User
- Group
- Friendship
- Membership

- FriendshipInvitation
- MembershipInvitation

Users can request friendships with other users, which can then be accepted. Users can request membership of a group which may be accepted by the group's owner. A group owner may also request for a user to join a group which the user may then choose to join. All information about the friends a user has and groups they are a member of are store within the Friendship and Membership objects. FriendshipInvitation and MembershipInvitation are similar to their non-invitation counterpart except the request is made to an external party via an email address.

Friendships and Memberships are separate objects from the User or Group. However, it is possible to write SWRL rules to infer is-friends-with and (is/has)-member triples that can be stored as properties of the User or Group object.

The Base module also contains a Message class for sending messages between users within myExperiment.

Object Annotation

The Base module contains an abstract class Annotation. The Annotations module contains types of Annotation object used in myExperiment including:

- Citation
- Comment
- Favourite
- Rating
- Review
- Tagging

No data for the Annotation is stored as part of the Contribution rather the Contribution is pointed to by the Annotation. Fig.4 is an RDF graph of an example Annotation:

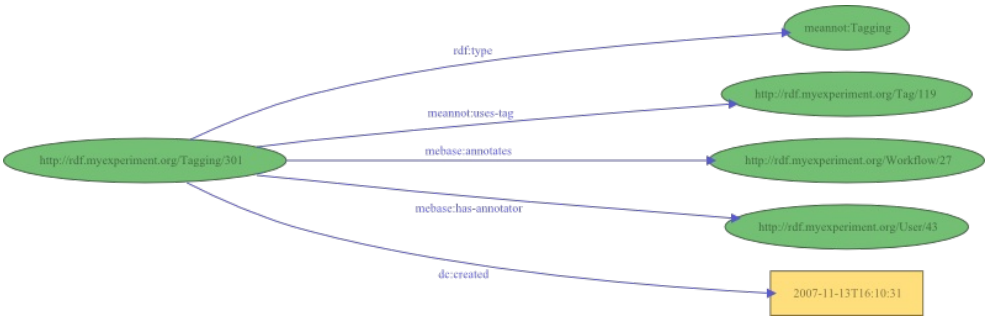


Fig.4 Example Annotation (A Tagging)

Packs

The Packs module provides a way for aggregating myExperiment Contributions and external URLs within a single object and implements the OAI-ORE Specification (see example). Packs are a subclass of OAI-ORE schema's Aggregation class. To allow metadata to be associated only with an AggregatedResource when it occurs within a particular Aggregation, the Packs module implements two ORE Proxy classes:

- LocalPackEntry
- RemotePackEntry

Remote Pack Entries are pointers to URLs (see example) whereas Local Pack Entries

have a pointer (mepack:requires) to specify the myExperiment Contribution they represent (see example). Packs contain the property ore:isDescribedBy to allow the discovery of the associated ResourceMaps allowing the Pack to be exported to another repository (see example).

Relationships in Packs

As well as describing items in a Pack it is possible to describe the relationships between them. A RelationshipEntry allows two items to be link together with a predicate. These predicates can be described in user-defined ontologies. A RelationshipEntry is similar to a LocalPackEntry or RemotePackEntry in that it refers to a Relationship and allows its use it with the context of a particular Pack. It does this by using a ore:proxyFor to the Relationship and an ore:ProxyIn to the Pack, (see Fig.5). A RelationshipEntry differs from LocalPackEntry and RemotePackEntry because it can be used to assert a Relationship in the context of any entity, not just a Pack.

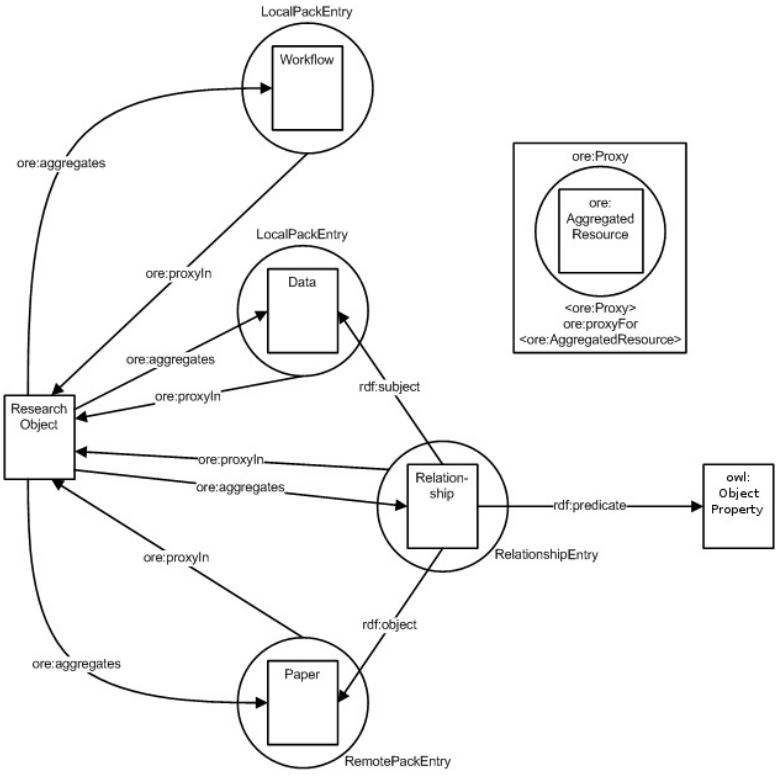


Fig.5 Example Structure of a Pack (Research Object)

A Relationship is made up of an RDF, subject, predicate and object. URI is a UUID version 5 URN (i.e. an SHA-1 hash) generated from the concatenation of the subject, predicate and object URIs. (N.B. So that anyone can generate the same URN for the same triple, no namespace parameter is provided to the UUID generating function). This ensures that all identical relationships have the same URI, essentially indexing relationships to make it easier to find packs that share the same relationships.

Experiments

myExperiment provides cloud services to run Runnable Contributions (e.g. Workflows) in remote Runners. These cloud services are provided by the Experiments module. An Experiment is a specialisation of a Pack that aggregates Jobs, (making Jobs AggregatedResources). Jobs are enacted workflows on a remote runner. A Job contains all the information about the Workflow being run, the Runner being used, the inputs/outputs of the Job and status information. The inputs and outputs of a Job have their own URIs so they can be referenced separately. (See an example of a Job). Like Packs, Experiments have an ore:isDescribedBy property to allow the discovery of the associated ResourceMaps allowing them exported to another repository.

Components

Workflows in essence are a number of interlinked processes that have initial inputs and final outputs. Using Taverna workflows as an exemplar six types of Workflow Components have been defined.

- **Processor** A service that performs some processing on one or more Inputs and produces one or more Ouputs.
- **Source** An initial piece of data to be processed by the Workflow
- **Sink** A final piece of data produced by the Workflow
- **Input** A piece of data going into a Processor
- **Output** A piece of data coming out of a Processor
- **Link** A connection between an Output of a Source of Processor to the Input for another Processor or Sink

All these Workflow Components are encompassed within a Dataflow. Fig.6 shows how these Workflow Components interlink to form a Dataflow. Currently every publicly available Taverna workflow has an executes-dataflow property to it's Dataflow.

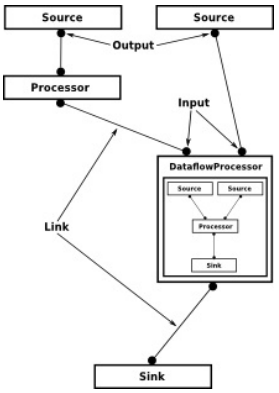


Fig.6 Overview of Organisation of Components into a Workflow

A Workflow may have many Processors, each must be one of the following types:

- **BeanshellProcessor** This executes a Beanshell script
- **ConstantProcessor** This produces a constant output, e.g. echoes a particular string
- **DataflowProcessor** This executes another Dataflow allowing workflows to be nested.
- **WSDLProcessor** This calls a remote WSDL service
- **OtherProcessor** Represent all other type of Processor. All Processor have a processor-type property that gives a more specific categorisation for the Processor

Due to it being possible to nest one Dataflow within another each Workflow Component has a belongs-to-workflow that makes its possible to execute a simple SPARQL query to find all the components of a paticular workflow.

B.2 myExperiment How To SPARQL Documentation

As of 16/102011.

B.2.1 Using the SPARQL Endpoint

my experiment **How To SPARQL**

Submit Feedback/Bug Report

1. Using the SPARQL Endpoint Go Back to Contents Page

1. Using the SPARQL Endpoint

1.1. Useful Prefixes

The myExperiment SPARQL endpoint has a number of features to assist in its use. As explained in the PREFIX page, both the PREFIX and BASE clauses facilitate writing more succinct and easier to follow queries. If you need to include any prefixes in your query just use the PREFIX as defined in Useful Prefixes (e.g. rdfs, mebase etc.) and the PREFIX line will be prepended to the query before it is executed.

1.2. Formatting

myExperiment's triplestore to which the SPARQL endpoint queries is updated with the previous day's data between 08:10 and 08:30 each morning UK time. The endpoint provides information about the time the latest snapshot was taken and the number of triples, so you can be sure how current the data is.

SPARQL results can be rendered in a number of formats:

- **HTML Table:** Renders the results in an HTML table, giving a more visual way to view your results.
- **XML:** Renders just the SPARQL results on an application/sparql-results+xml content type page.
- **Text:** Returns a Simple Subject-Predicate-Object text representation.
- **JSON:** Returns a JSON encoded version, of SPARQL results XML.
- **CSV:** Returns results as comma separated values, in table columns format.
- **CSV Matrix:** Returns results as comma separated values in a matrix format, where the first variable is enumrated on the x-axis, the second variable is enumerated on the y-axis and 1s are rendered for each tuplet.

An example of a use for the CSV Matrix format is for friendships:

```
PREFIX mebase: <http://rdf.myexperiment.org/ontologies/base/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?requester ?accepter
WHERE{
  ?friendship rdf:type mebase:Friendship ;
    mebase:accepted-at ?accepted_time ;
    mebase:has-requester ?requester ;
    mebase:has-accepter ?accepter .
}
```

[Run]
[Hide Example Results]

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1		http://www.myexperiment.org/users/1	http://www.myexperiment.org/users/10	http://www.myexperiment.org/users/1019	http://www.myexperiment.org/users/1027	http://www.myexperiment.org/users/1053	http://www.myexperiment.org/users/1056	http://www.myexperiment.org/users/1062	http://www.myexperiment.org/users/1071	http://www.myexperiment.org/users/1077	http://www.myexperiment.org/users/1102	http://www.myexperiment.org/users/1103	http://www.myexperiment.org/users/1106	http://www.myexperiment.org/users/1116	http://www.myexperiment.org/users/1165	http://www.myexperiment.org/users/1169
2	http://www.myexperiment.org/users/1	1														
3	http://www.myexperiment.org/users/10															
4	http://www.myexperiment.org/users/1007															
5	http://www.myexperiment.org/users/1017															
6	http://www.myexperiment.org/users/1019															
7	http://www.myexperiment.org/users/1027															
8	http://www.myexperiment.org/users/1063							1								
9	http://www.myexperiment.org/users/1070															
10	http://www.myexperiment.org/users/1071															
11	http://www.myexperiment.org/users/1077								1							
12	http://www.myexperiment.org/users/1102												1			
13	http://www.myexperiment.org/users/1105										1	1				
14	http://www.myexperiment.org/users/1112															
15	http://www.myexperiment.org/users/1117													1		
16	http://www.myexperiment.org/users/116															

1.3. Soft Limit

The *Soft Limit* option determines the amount of resources dedicated to returning all the matching results. In general 1% is sufficient. However, if all the results are not returned then a warning message will be displayed and you can try re-running the query with a greater Soft Limit percentage.

1.4. Reasoning

The *Enable RDFS Reasoning* option allows you to make use of 4Store Reasoner, an RDFS reasoner addition to 4Store. This will perform query-time RDFS reasoning on RDFS subClassOf, subPropertyOf, domain and range properties. The following query will return more results if RDFS reasoning is enabled because all the super classes of Workflow:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT DISTINCT ?type
WHERE {
  <http://www.myexperiment.org/workflows/16> rdf:type ?type
}
```

[Run (Without Reasoning)] [Run (With Reasoning)]
[Hide Example Results]

Without Reasoning

type
http://rdf.myexperiment.org/ontologies/contributions/Workflow

With Reasoning

type
b139c71020000004f
http://rdf.myexperiment.org/ontologies/base/Submission
http://rdf.myexperiment.org/ontologies/attrib_credit/Attributable
http://rdf.myexperiment.org/ontologies/annotations/Favouritable
http://rdf.myexperiment.org/ontologies/annotations/Commentable
http://rdf.myexperiment.org/ontologies/attrib_credit/Creditable
http://rdf.myexperiment.org/ontologies/base/Upload
http://rdf.myexperiment.org/ontologies/annotations/Taggable
http://rdf.myexperiment.org/ontologies/annotations/Citationable
http://rdf.myexperiment.org/ontologies/contributions/Workflow
http://rdf.myexperiment.org/ontologies/annotations/Rateable
http://rdf.myexperiment.org/ontologies/base/Annotatable
http://rdf.myexperiment.org/ontologies/base/Contribution
http://rdf.myexperiment.org/ontologies/base/Interface
http://rdf.myexperiment.org/ontologies/base/Versionable
http://rdf.myexperiment.org/ontologies/contributions/AbstractWorkflow
http://rdf.myexperiment.org/ontologies/experiments/Runnable
http://rdf.myexperiment.org/ontologies/annotations/Reviewable

1.5. Automated Querying

If you wish to write automated queries rather than using the endpoint form you can insert the query (in URL encoded format) into the URL as the *query* parameter in the HTTP GET header. If you have built a query using the endpoint form and want to use it as an automated service in something such as a workflow, instead of clicking "Submit Query", click on "Generate Service from Query". This will take you to a page with a link something like the one below, that you can copy and paste into your workflow or HTTP request capable application.

http://rdf.myexperiment.org/sparql?**query=PREFIX+mebase%3A+%3Chttp%3A%2F%2Frdf.myexperiment.org%2Fontologies%2Fbase%2F%3E%0D%0APREFIX+rdf%3A+%3Chttp%3A%2F%2Fwww.w3.org%2F1999%2F02%2F22-rdf-syntax-ns%23%3E%0D%0ASELECT+%3Frequester+%3Faccepter+%0D%0AWHERE%7B+%0D%0A++%3Ffriendship+rdf%3Atype+mebase%3AFriendship+%3B%0D%0A++++mebase%3Aaccepted-at+%3Faccepted_time+%3B+%0D%0A++++mebase%3Ahas-requester+%3Frequester+%3B%0D%0A++++mebase%3Ahas-accepter+%3Faccepter+%0D%0A%7D**

As you will notice is you click on the link above this will return results is raw SPARQL results XML format. If you wish to get the results in a different format you can also set the *formatting* parameter in the GET header:

http://rdf.myexperiment.org/sparql?query=PREFIX+mebase%3A+%3Chttp%3A%2F%2Frdf.myexperiment.org%2Fontologies%2Fbase%2F%3E%0D%0APREFIX+rdf%3A+%3Chttp%3A%2F%2Fwww.w3.org%2F1999%2F02%2F22-rdf-syntax-ns%23%3E%0D%0ASELECT+%3Frequester+%3Faccepter+%0D%0AWHERE%7B+%0D%0A++%3Ffriendship+rdf%3Atype+mebase%3AFriendship+%3B%0D%0A++++mebase%3Aaccepted-at+%3Faccepted_time+%3B+%0D%0A++++mebase%3Ahas-requester+%3Frequester+%3B%0D%0A++++mebase%3Ahas-accepter+%3Faccepter+%0D%0A%7D**&formatting=HTML Table**

The options for the formatting parameter are:

- HTML Table
- XML
- Text
- JSON
- CSV
- CSV Matrix

The Soft Limit can also be set as an integer between 1 (default) and 100 by using the GET header parameter *softlimit*:

http://rdf.myexperiment.org/sparql?query=PREFIX+mebase%3A+%3Chttp%3A%2F%2Frdf.myexperiment.org%2Fontologies%2Fbase%2F%3E%0D%0APREFIX+rdf%3A+%3Chttp%3A%2F%2Fwww.w3.org%2F1999%2F02%2F22-rdf-syntax-ns%23%3E%0D%0ASELECT+%3Frequester+%3Faccepter+%0D%0AWHERE%7B+%0D%0A++%3Ffriendship+rdf%3Atype+mebase%3AFriendship+%3B%0D%0A++++mebase%3Aaccepted-at+%3Faccepted_time+%3B+%0D%0A++++mebase%3Ahas-requester+%3Frequester+%3B%0D%0A++++mebase%3Ahas-accepter+%3Faccepter+%0D%0A%7D&formatting=HTML Table**&softlimit=5**

Finally reasoning can be enabled by setting the *reasoning* parameter to 1, yes, or true in the GET header:

B.2.2 PREFIX

myexperiment **How To SPARQL**

Submit Feedback/Bug Report

◀◀ ◀ 2. PREFIX Go ▶ ▶▶ Back to Contents Page

2. PREFIX

The first part of most queries is the listing of one or more prefixes:

```
PREFIX mebase: <http://rdf.myexperiment.org/ontologies/base/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?a ?text
WHERE {
  ?a rdf:type mebase:Announcement .
  ?a mebase:text ?text
}
```

[Run]
[Hide Example Results]

a	text
http://www.myexperiment.org/announcements/6	myExperiment is discussed by Jim Hendler in his article *Reinventing Academic Publishing, Part 3* in IEEE Intelligent Systems January/February 2008, page 2-3.
http://www.myexperiment.org/announcements/29	A new project to link the myExperiment system with two existing electronic laboratory notebook (ELN) systems developed in Southampton, currently in use in several departments, has been announced today (June 11, 2009). By integrating these ELNs with myExperiment we will pave the way for longer term takeup in the experimental laboratory science communities, including Chemistry, Physics, Biology and Engineering. The project will commence in July 2009 and will be led by Prof Jeremy Frey, who is based in the School of Chemistry at Southampton and is one of the partners in the myGrid Platform. For more info please see http://wiki.myexperiment.org/index.php/MyExperimentalScience -- Dave
http://www.myexperiment.org/announcements/14	We're pleased to announce that we've remodelled the myexperiment wiki to support the myExperiment developer community. There is now a Developers' section where those of you developing over the API can share information about your projects. It currently includes Google gadgets, the Taverna plugin, workflows for Facebook and the Java API. This section also carries information about people running their own myExperiments, and myExperiment developer documentation. The other information is now organised into Users, News and About. Previously you needed a wiki account to get at anything other than the main page, but now everything is publicly available, so you won't need to use your old accounts. We're really keen that everyone working over the API uses the wiki, if only to list who you are and what you're doing - then all the developers will be able to help out as the community grows. To get an account so that you can contribute content to the Developer pages please email Don Cruickshank on dgc@ecs.soton.ac.uk
http://www.myexperiment.org/announcement/3	In true 'perpetual beta' fashion, we continue our updates and additions to myExperiment: ◦ In the user profile page, you can now see how many times a user has been credited for Workflows and Files in myExperiment. You can also see these credited items directly in the new "Creditations" tab, when viewing a user's profile. ◦ Clearer statistics in the Workflow, File and User pages. ◦ Improved layout for the Workflow and File pages. Feedback always welcome!
http://www.myexperiment.org/announcement/25	Our paper "Software Design for Empowering Scientists", which discusses the design principles of Taverna and myExperiment, has appeared in the January/February 2009 issue of IEEE Software ("Developing Scientific Software, Part 2"). The article is available to subscribers on http://www2.computer.org/portal/web/csdl/doi/10.1109/MS.2009.22 and the preprint is available on http://eprints.ecs.soton.ac.uk/15032/ The full reference is De Roure, D. and Goble, C. Software Design for Empowering Scientists, IEEE Software, Volume 26, Issue 1, Jan-Feb 2009, pages 88-95. Digital Object Identifier 10.1109/MS.2009.22 Also, myExperiment, along with OpenWetWare and nanoHub, feature in the article "Research intelligence - Log on to a global laboratory" in the January 1st issue of THE (Times Higher Education) , written by Sarah Collinson and Zoe Corbyn. You can find the article on http://www.timeshighereducation.co.uk/story.asp?sectioncode=26&storycode=404834 ...and Happy New Year to all our users! -- Dave

Prefixes are not required within a query they just save rewriting the namespace each time you need to use it in a query and it

makes the query easier to read. The previous query could be re-written as follows if you didn't want to use prefixes:

```
SELECT ?a ?text
WHERE {
  ?a <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
     <http://rdf.myexperiment.org/ontologies/base/Announcement> .
  ?a <http://rdf.myexperiment.org/ontologies/base/text> ?text
}
```

[Run]
[Hide Example Results]

a	text
http://www.myexperiment.org/announcements/6	myExperiment is discussed by Jim Hendler in his article *Reinventing Academic Publishing, Part 3* in IEEE Intelligent Systems January/February 2008, page 2-3.
http://www.myexperiment.org/announcements/29	A new project to link the myExperiment system with two existing electronic laboratory notebook (ELN) systems developed in Southampton, currently in use in several departments, has been announced today (June 11, 2009). By integrating these ELNs with myExperiment we will pave the way for longer term takeup in the experimental laboratory science communities, including Chemistry, Physics, Biology and Engineering. The project will commence in July 2009 and will be led by Prof Jeremy Frey, who is based in the School of Chemistry at Southampton and is one of the partners in the myGrid Platform. For more info please see http://wiki.myexperiment.org/index.php/MyExperimentalScience -- Dave
http://www.myexperiment.org/announcements/14	We're pleased to announce that we've remodelled the myexperiment wiki to support the myExperiment developer community. There is now a Developers' section where those of you developing over the API can share information about your projects. It currently includes Google gadgets, the Taverna plugin, workflows for Facebook and the Java API. This section also carries information about people running their own myExperiments, and myExperiment developer documentation. The other information is now organised into Users, News and About. Previously you needed a wiki account to get at anything other than the main page, but now everything is publicly available, so you won't need to use your old accounts. We're really keen that everyone working over the API uses the wiki, if only to list who you are and what you're doing - then all the developers will be able to help out as the community grows. To get an account so that you can contribute content to the Developer pages please email Don Cruickshank on dgc@ecs.soton.ac.uk
http://www.myexperiment.org/announcement/3	In true 'perpetual beta' fashion, we continue our updates and additions to myExperiment: • In the user profile page, you can now see how many times a user has been credited for Workflows and Files in myExperiment. You can also see these credited items directly in the new "Creditations" tab, when viewing a user's profile. • Clearer statistics in the Workflow, File and User pages. • Improved layout for the Workflow and File pages. Feedback always welcome!
http://www.myexperiment.org/announcement/25	Our paper "Software Design for Empowering Scientists", which discusses the design principles of Taverna and myExperiment, has appeared in the January/February 2009 issue of IEEE Software ("Developing Scientific Software, Part 2"). The article is available to subscribers on http://www2.computer.org/portal/web/csdl/doi/10.1109/MS.2009.22 and the preprint is available on http://eprints.ecs.soton.ac.uk/15032/ The full reference is De Roure, D. and Goble, C. Software Design for Empowering Scientists, IEEE Software, Volume 26, Issue 1, Jan-Feb 2009, pages 88-95. Digital Object Identifier 10.1109/MS.2009.22 Also, myExperiment, along with OpenWetWare and nanoHub, feature in the article "Research intelligence - Log on to a global laboratory" in the January 1st issue of THE (Times Higher Education) , written by Sarah Collinson and Zoe Corbyn. You can find the article on http://www.timeshighereducation.co.uk/story.asp?sectioncode=26&storycode=404834 ...and Happy New Year to all our users! -- Dave

2.1. BASE

The *BASE* clause is similar to the PREFIX clause but a name is not required, just the URI path needs be defined. If you wish to use the BASE clause it must proceed any PREFIX clauses. A common use of the BASE clause is to define the path for data that is being queried. E.g.

```
SELECT ?p ?o
WHERE{
  <http://www.myexperiment.org/workflows/12> ?p ?o
}
```

[Run]
[Hide Example Results]

p	o
http://purl.org/dc/terms/modified	2007-11-13T16:16:54Z
http://purl.org/dc/terms/hasFormat	http://www.myexperiment.org/workflow.xml?id=12
http://purl.org/dc/terms/hasFormat	http://www.myexperiment.org/workflows/12.rdf
http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://rdf.myexperiment.org/ontologies/contributions/Workflow
http://purl.org/dc/terms/title	Transcribe a DNA sequence into an RNA sequence
http://xmlns.com/foaf/0.1/homepage	http://www.myexperiment.org/workflows/12.html
http://purl.org/dc/terms/created	2007-10-03T18:35:58Z
http://purl.org/dc/terms/description	This workflow transcribes a DNA sequence into an RNA sequence
http://rdfs.org/sioc/ns#has_owner	http://www.myexperiment.org/users/43
http://rdf.myexperiment.org/ontologies/base/content-url	http://www.myexperiment.org/workflows/12/download/transcribe_a_dna_sequence_into_an_rna_sequence_27205.xml
http://rdf.myexperiment.org/ontologies/base/has-content-type	http://www.myexperiment.org/content_types/1
http://rdf.myexperiment.org/ontologies/base/filename	transcribe_a_dna_sequence_into_an_rna_sequence_27205.xml
http://rdf.myexperiment.org/ontologies/base/has-policy	http://www.myexperiment.org/workflows/12/policies/180
http://rdf.myexperiment.org/ontologies/base/has-license	http://www.myexperiment.org/licenses/1
http://rdf.myexperiment.org/ontologies/annotations/has-tagging	http://www.myexperiment.org/tags/544/taggings/236
http://rdf.myexperiment.org/ontologies/annotations/has-tagging	http://www.myexperiment.org/tags/547/taggings/300
http://rdf.myexperiment.org/ontologies/annotations/has-tagging	http://www.myexperiment.org/tags/409/taggings/234
http://rdf.myexperiment.org/ontologies/annotations/has-tagging	http://www.myexperiment.org/tags/214/taggings/299
http://rdf.myexperiment.org/ontologies/annotations/has-tagging	http://www.myexperiment.org/tags/497/taggings/235
http://rdf.myexperiment.org/ontologies/annotations/has-tagging	http://www.myexperiment.org/tags/119/taggings/301
http://rdf.myexperiment.org/ontologies/annotations/has-tagging	http://www.myexperiment.org/tags/542/taggings/302
http://rdf.myexperiment.org/ontologies/viewings_downloads/downloaded	630
http://rdf.myexperiment.org/ontologies/viewings_downloads/viewed	887
http://www.openarchives.org/ore/terms/isDescribedBy	http://www.myexperiment.org/atom_entries/workflows/12
http://www.openarchives.org/ore/terms/isDescribedBy	http://www.myexperiment.org/resource_maps/workflows/12
http://rdf.myexperiment.org/ontologies/contributions/thumbnail	http://www.myexperiment.org/workflow/image/12/thumb/transcribe_a_dna_sequence_into_an_rna_sequence_27205.png
http://rdf.myexperiment.org/ontologies/components/executes-dataflow	http://www.myexperiment.org/workflows/12/versions/2/dataflow
http://rdf.myexperiment.org/ontologies/contributions/svg	http://www.myexperiment.org/workflow/svg/12/transcribe_a_dna_sequence_into_an_rna_sequence_27205.svg
http://rdf.myexperiment.org/ontologies/contributions/preview	http://www.myexperiment.org/workflow/image/12/transcribe_a_dna_sequence_into_an_rna_sequence_27205.png
http://rdf.myexperiment.org/ontologies/base/has-current-version	http://www.myexperiment.org/workflows/12/versions/2
http://rdf.myexperiment.org/ontologies/base/last-edited-by	http://www.myexperiment.org/users/43
http://rdf.myexperiment.org/ontologies/contributions/thumbnail-big	http://www.myexperiment.org/workflow/image/12/medium/transcribe_a_dna_sequence_into_an_rna_sequence_27205.png
http://rdf.myexperiment.org/ontologies/base/has-version	http://www.myexperiment.org/workflows/12/versions/1
http://rdf.myexperiment.org/ontologies/annotations/has-comment	http://www.myexperiment.org/users/9/favourites/5
http://rdf.myexperiment.org/ontologies/annotations/has-comment	http://www.myexperiment.org/workflows/27/comments/92
http://rdf.myexperiment.org/ontologies/annotations/has-comment	http://www.myexperiment.org/workflows/27/comments/275
http://rdf.myexperiment.org/ontologies/annotations/has-comment	http://www.myexperiment.org/users/1634/favourites/47
http://rdf.myexperiment.org/ontologies/annotations/has-rating	http://www.myexperiment.org/workflows/27/ratings/59
http://rdf.myexperiment.org/ontologies/annotations/has-rating	http://www.myexperiment.org/workflows/27/ratings/15

Could be rewritten:

```
BASE <http://www.myexperiment.org/>
SELECT ?p ?o
WHERE{
  <workflows/12> ?p ?o
}
```

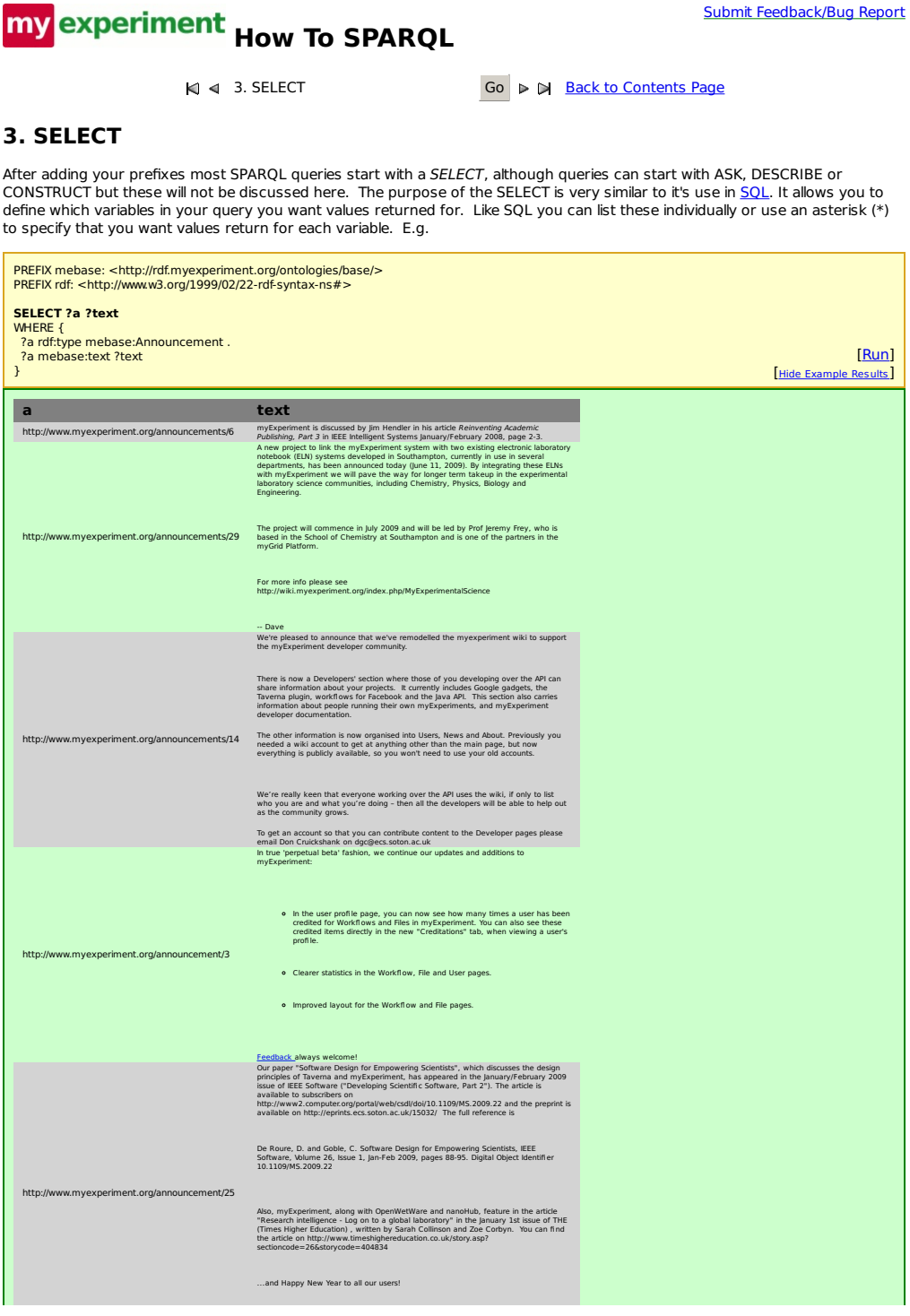
[Run]
[Hide Example Results]

p	o
http://purl.org/dc/terms/modified	2007-11-13T16:16:54Z
http://purl.org/dc/terms/hasFormat	http://www.myexperiment.org/workflow.xml?id=12
http://purl.org/dc/terms/hasFormat	http://www.myexperiment.org/workflows/12.rdf
http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://rdf.myexperiment.org/ontologies/contributions/Workflow
http://purl.org/dc/terms/title	Transcribe a DNA sequence into an RNA sequence
http://xmlns.com/foaf/0.1/homepage	http://www.myexperiment.org/workflows/12.html
http://purl.org/dc/terms/created	2007-10-03T18:35:58Z
http://purl.org/dc/terms/description	This workflow transcribes a DNA sequence into an RNA sequence
http://rdfs.org/sioc/ns#has_owner	http://www.myexperiment.org/users/43
http://rdf.myexperiment.org/ontologies/base/content-url	http://www.myexperiment.org/workflows/12/download/transcribe_a_dna_sequence_into_an_rna_sequence_27205.xml
http://rdf.myexperiment.org/ontologies/base/has-content-type	http://www.myexperiment.org/content_types/1
http://rdf.myexperiment.org/ontologies/base/filename	transcribe_a_dna_sequence_into_an_rna_sequence_27205.xml
http://rdf.myexperiment.org/ontologies/base/has-policy	http://www.myexperiment.org/workflows/12/policies/180
http://rdf.myexperiment.org/ontologies/base/has-license	http://www.myexperiment.org/licenses/1
http://rdf.myexperiment.org/ontologies/annotations/has-tagging	http://www.myexperiment.org/tags/544/taggings/236
http://rdf.myexperiment.org/ontologies/annotations/has-tagging	http://www.myexperiment.org/tags/547/taggings/300
http://rdf.myexperiment.org/ontologies/annotations/has-tagging	http://www.myexperiment.org/tags/409/taggings/234
http://rdf.myexperiment.org/ontologies/annotations/has-tagging	http://www.myexperiment.org/tags/214/taggings/299
http://rdf.myexperiment.org/ontologies/annotations/has-tagging	http://www.myexperiment.org/tags/497/taggings/235
http://rdf.myexperiment.org/ontologies/annotations/has-tagging	http://www.myexperiment.org/tags/119/taggings/301
http://rdf.myexperiment.org/ontologies/annotations/has-tagging	http://www.myexperiment.org/tags/542/taggings/302
http://rdf.myexperiment.org/ontologies/viewings_downloads/downloaded	630
http://rdf.myexperiment.org/ontologies/viewings_downloads/viewed	887
http://www.openarchives.org/ore/terms/isDescribedBy	http://www.myexperiment.org/atom_entries/workflows/12
http://www.openarchives.org/ore/terms/isDescribedBy	http://www.myexperiment.org/resource_maps/workflows/12
http://rdf.myexperiment.org/ontologies/contributions/thumbnail	http://www.myexperiment.org/workflow/image/12/thumb/transcribe_a_dna_sequence_into_an_rna_sequence_27205.png
http://rdf.myexperiment.org/ontologies/components/executes-dataflow	http://www.myexperiment.org/workflows/12/versions/2/dataflow
http://rdf.myexperiment.org/ontologies/contributions/svg	http://www.myexperiment.org/workflow/svg/12/transcribe_a_dna_sequence_into_an_rna_sequence_27205.svg
http://rdf.myexperiment.org/ontologies/contributions/preview	http://www.myexperiment.org/workflow/image/12/transcribe_a_dna_sequence_into_an_rna_sequence_27205.png
http://rdf.myexperiment.org/ontologies/base/has-current-version	http://www.myexperiment.org/workflows/12/versions/2
http://rdf.myexperiment.org/ontologies/base/last-edited-by	http://www.myexperiment.org/users/43
http://rdf.myexperiment.org/ontologies/contributions/thumbnail-big	http://www.myexperiment.org/workflow/image/12/medium/transcribe_a_dna_sequence_into_an_rna_sequence_27205.png
http://rdf.myexperiment.org/ontologies/base/has-version	http://www.myexperiment.org/workflows/12/versions/1
http://rdf.myexperiment.org/ontologies/annotations/has-comment	http://www.myexperiment.org/users/9/favourites/5

http://rdf.myexperiment.org/ontologies/annotations/has-comment	http://www.myexperiment.org/workflows/27/comments/92
http://rdf.myexperiment.org/ontologies/annotations/has-comment	http://www.myexperiment.org/workflows/27/comments/275
http://rdf.myexperiment.org/ontologies/annotations/has-comment	http://www.myexperiment.org/users/1634/favourites/47
http://rdf.myexperiment.org/ontologies/annotations/has-rating	http://www.myexperiment.org/workflows/27/ratings/59
http://rdf.myexperiment.org/ontologies/annotations/has-rating	http://www.myexperiment.org/workflows/27/ratings/15

2. PREFIX Go Back to Contents Page

B.2.3 SELECT



Submit Feedback/Bug Report

How To SPARQL

3. SELECT Go Back to Contents Page

3. SELECT

After adding your prefixes most SPARQL queries start with a *SELECT*, although queries can start with ASK, DESCRIBE or CONSTRUCT but these will not be discussed here. The purpose of the SELECT is very similar to it's use in SQL. It allows you to define which variables in your query you want values returned for. Like SQL you can list these individually or use an asterisk (*) to specify that you want values return for each variable. E.g.

```
PREFIX mebase: <http://rdf.myexperiment.org/ontologies/base/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?a ?text
WHERE {
  ?a rdf:type mebase:Announcement .
  ?a mebase:text ?text
}
```

[Run]
[Hide Example Results]

a	text
http://www.myexperiment.org/announcements/6	myExperiment is discussed by Jim Hendler in his article *Reinventing Academic Publishing, Part 3* in IEEE Intelligent Systems January/February 2008, page 2-3.
http://www.myexperiment.org/announcements/29	A new project to link the myExperiment system with two existing electronic laboratory notebook (ELN) systems developed in Southampton, currently in use in several departments, has been announced today (June 11, 2009). By integrating these ELNs with myExperiment we will pave the way for longer term takeup in the experimental laboratory science communities, including Chemistry, Physics, Biology and Engineering. The project will commence in July 2009 and will be led by Prof Jeremy Frey, who is based in the School of Chemistry at Southampton and is one of the partners in the myGrid Platform. For more info please see http://wiki.myexperiment.org/index.php/MyExperimentalScience -- Dave
http://www.myexperiment.org/announcements/14	We're pleased to announce that we've remodelled the myexperiment wiki to support the myExperiment developer community. There is now a Developers' section where those of you developing over the API can share information about your projects. It currently includes Google gadgets, the Taverna plugin, workflows for Facebook and the Java API. This section also carries information about people running their own myExperiments, and myExperiment developer documentation. The other information is now organised into Users, News and About. Previously you needed a wiki account to get at anything other than the main page, but now everything is publicly available, so you won't need to use your old accounts. We're really keen that everyone working over the API uses the wiki, if only to list who you are and what you're doing – then all the developers will be able to help out as the community grows. To get an account so that you can contribute content to the Developer pages please email Don Cruickshank on dgc@ecs.soton.ac.uk
http://www.myexperiment.org/announcement/3	In true 'perpetual beta' fashion, we continue our updates and additions to myExperiment: ◦ In the user profile page, you can now see how many times a user has been credited for Workflows and Files in myExperiment. You can also see these credited items directly in the new "Creditations" tab, when viewing a user's profile. ◦ Clearer statistics in the Workflow, File and User pages. ◦ Improved layout for the Workflow and File pages. Feedback always welcome!
http://www.myexperiment.org/announcement/25	Our paper "Software Design for Empowering Scientists", which discusses the design principles of Taverna and myExperiment, has appeared in the January/February 2009 issue of IEEE Software ("Developing Scientific Software, Part 2"). The article is available to subscribers on http://www2.computer.org/portal/web/csdl/doi/10.1109/MS.2009.22 and the preprint is available on http://eprints.ecs.soton.ac.uk/15032/ The full reference is De Roure, D. and Goble, C. Software Design for Empowering Scientists, IEEE Software, Volume 26, Issue 1, Jan-Feb 2009, pages 88-95. Digital Object Identifier 10.1109/MS.2009.22 Also, myExperiment, along with OpenWetWare and nanoHub, feature in the article "Research intelligence - Log on to a global laboratory" in the January 1st issue of THE (Times Higher Education) , written by Sarah Collinson and Zoe Corbyn. You can find the article on http://www.timeshighereducation.co.uk/story.asp?sectioncode=26&storycode=404834 ...and Happy New Year to all our users!

	-- Dave

Is the same query as:

```
PREFIX mebase: <http://rdf.myexperiment.org/ontologies/base/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT *
WHERE {
  ?a rdf:type mebase:Announcement .
  ?a mebase:text ?text
}
```

[Run]
[Hide Example Results]

a	text
http://www.myexperiment.org/announcements/6	myExperiment is discussed by Jim Hendler in his article *Reinventing Academic Publishing, Part 3* in IEEE Intelligent Systems January/February 2008, page 2-3.
http://www.myexperiment.org/announcements/29	A new project to link the myExperiment system with two existing electronic laboratory notebook (ELN) systems developed in Southampton, currently in use in several departments, has been announced today (June 11, 2009). By integrating these ELNs with myExperiment we will pave the way for longer term takeup in the experimental laboratory science communities, including Chemistry, Physics, Biology and Engineering. The project will commence in July 2009 and will be led by Prof Jeremy Frey, who is based in the School of Chemistry at Southampton and is one of the partners in the myGrid Platform. For more info please see http://wiki.myexperiment.org/index.php/MyExperimentalScience -- Dave
http://www.myexperiment.org/announcements/14	We're pleased to announce that we've remodelled the myexperiment wiki to support the myExperiment developer community. There is now a Developers' section where those of you developing over the API can share information about your projects. It currently includes Google gadgets, the Taverna plugin, workflows for Facebook and the Java API. This section also carries information about people running their own myExperiments, and myExperiment developer documentation. The other information is now organised into Users, News and About. Previously you needed a wiki account to get at anything other than the main page, but now everything is publicly available, so you won't need to use your old accounts. We're really keen that everyone working over the API uses the wiki, if only to list who you are and what you're doing – then all the developers will be able to help out as the community grows. To get an account so that you can contribute content to the Developer pages please email Don Cruickshank on dgc@ecs.soton.ac.uk
http://www.myexperiment.org/announcement/3	In true 'perpetual beta' fashion, we continue our updates and additions to myExperiment: • In the user profile page, you can now see how many times a user has been credited for Workflows and Files in myExperiment. You can also see these credited items directly in the new "Creditations" tab, when viewing a user's profile. • Clearer statistics in the Workflow, File and User pages. • Improved layout for the Workflow and File pages. Feedback always welcome!
http://www.myexperiment.org/announcement/25	Our paper "Software Design for Empowering Scientists", which discusses the design principles of Taverna and myExperiment, has appeared in the January/February 2009 issue of IEEE Software ("Developing Scientific Software, Part 2"). The article is available to subscribers on http://www2.computer.org/portal/web/csdl/doi/10.1109/MS.2009.22 and the preprint is available on http://eprints.ecs.soton.ac.uk/15032/ The full reference is De Roure, D. and Goble, C. Software Design for Empowering Scientists, IEEE Software, Volume 26, Issue 1, Jan-Feb 2009, pages 88-95. Digital Object Identifier 10.1109/MS.2009.22 Also, myExperiment, along with OpenWetWare and nanoHub, feature in the article "Research intelligence - Log on to a global laboratory" in the January 1st issue of THE (Times Higher Education) , written by Sarah Collinson and Zoe Corbyn. You can find the article on http://www.timeshighereducation.co.uk/story.asp?sectioncode=26&storycode=404834 ...and Happy New Year to all our users! -- Dave

3.1. DISTINCT

It is not uncommon that the sets of results will return duplicates. If you don't want duplicates you can append *DISTINCT* after SELECT.

```
PREFIX meannot: <http://rdf.myexperiment.org/ontologies/annotations/>
PREFIX sioc: <http://rdfs.org/sioc/ns#>
```

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX mebase: <http://rdf.myexperiment.org/ontologies/base/>
SELECT DISTINCT ?annotator_name
WHERE {
  ?comment mebase:annotates <http://www.myexperiment.org/workflows/52> .
  ?comment rdf:type meannot:Comment .
  ?comment mebase:has-annotator ?annotator .
  ?annotator sioc:name ?annotator_name
}
```

[Run (Without DISTINCT)] [Run (With DISTINCT)]
[Hide Example Results]

Without DISTINCT	With DISTINCT
annotator_name	annotator_name
Don Cruickshank	Don Cruickshank
Franck Tanoh	Franck Tanoh
Franck Tanoh	
Franck Tanoh	

3.2. COUNT

As of SPARQL 1.1 it has been possible to apply mathematical functions to selected variables. The most straightforward of these is COUNT. The example below is a way of founding out how many public workflows myExperiment has without requiring any post-processing.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX mecontrib: <http://rdf.myexperiment.org/ontologies/contributions/>

SELECT (COUNT(?workflow) AS ?no_workflows)
WHERE {
  ?workflow rdf:type mecontrib:Workflow
}
```

[Run]
[Hide Example Results]

no_workflows
1255

3.3. SUM

Another common mathematical function now available is SUM, which like COUNT works in the same way as it does in SQL. The following example gives a sum of the downloads of all workflows owned by the user with the ID 43.

```
BASE <http://www.myexperiment.org/>
PREFIX mevd: <http://rdf.myexperiment.org/ontologies/viewings_downloads/>
PREFIX sioc: <http://rdfs.org/sioc/ns#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX mecontrib: <http://rdf.myexperiment.org/ontologies/contributions/>

SELECT (SUM(?downloaded) AS ?total_downloaded)
WHERE {
  ?workflow rdf:type mecontrib:Workflow .
  ?workflow sioc:has_owner <users/43> .
  ?workflow mevd:downloaded ?downloaded
}
```

[Run]
[Hide Example Results]

total_downloaded
29382

3.4. AVG

The SUM example above is probably not very useful if you want to know how popular a user's workflows are, as it is dependent on how many they have. A better way might be to work out th average number of downloads for a user's workflows. This can be found using the AVG function.

```
BASE <http://www.myexperiment.org/>
PREFIX mevd: <http://rdf.myexperiment.org/ontologies/viewings_downloads/>
PREFIX sioc: <http://rdfs.org/sioc/ns#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX mecontrib: <http://rdf.myexperiment.org/ontologies/contributions/>

SELECT (AVG(?downloaded) AS ?average_downloads)
WHERE {
  ?workflow rdf:type mecontrib:Workflow .
  ?workflow sioc:has_owner <users/43> .
  ?workflow mevd:downloaded ?downloaded
}
```

[Run]
[Hide Example Results]

average_downloads
367.2749999999999970618

3.5. MAX and MIN

As well as knowing the average you might want to know the most and least times a workflow that belongs to a user has been downloaded. This can be achieved using the MAX and MIN functions.

```
BASE <http://www.myexperiment.org/>
PREFIX mevd: <http://rdf.myexperiment.org/ontologies/viewings_downloads/>
PREFIX sioc: <http://rdfs.org/sioc/ns#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX mecontrib: <http://rdf.myexperiment.org/ontologies/contributions/>

SELECT (MAX(?downloaded) AS ?max_downloaded) (MIN(?downloaded) AS ?min_downloaded)
WHERE {
  ?workflow rdf:type mecontrib:Workflow .
  ?workflow sioc:has_owner <users/43> .
  ?workflow mevd:downloaded ?downloaded
}
```

[Run]

[Hide Example Results]

max_downloaded	min_downloaded
3250	1

These examples of the use of mathematical functions in the SELECT clause are quite basic and return just a single row of results. The GROUP BY function allows aggregation on a particular subject so you could give a league table of users by the total number of downloads for their workflows.

◀ ◀ 3. SELECT Go ▶ ▶ Back to Contents Page

B.2.4 WHERE

my experiment **How To SPARQL**

Submit Feedback/Bug Report

◀◀ ◀ 4. WHERE Go ▶ ▶▶ Back to Contents Page

4. WHERE

The *WHERE* clause of a SPARQL query defines where you want to find values for the variables you have defined in the SELECT clause. Inside the curly parenthesis {} the basic unit is a triple, this is made up of three components, a subject, a predicate and an object. This is much like the grammatical structure of a basic natural language sentence.

The boy **catches** **a ball**
Subject Predicate Object

In SPARQL the subject, predicate or object can take one of two forms a variable which is defined by putting a ? prior to the variable name, (e.g. ?a, ?text, etc.) or a Universal Resource Identifier (URI). As discussed in PREFIX (and BASE) this can take one of two forms depending on whether a prefix for the namespace of the URI has been defined within the query. These triples can then be concatenated together using a full-stop (.). By doing this you can build an interconnected graph of nodes joined by relationships. It is generally good to ensure that each triple connects at some point in the graph. The first of the following query is a completely connected graph where as the second isn't and therefore will return the superset of all possible homepage/mbox combinations. This is such a high number of results that there is intentionally not a link to run this query as the webserver with run out of memory trying to process all the results.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX mebase: <http://rdf.myexperiment.org/ontologies/base/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?homepage ?mbox
WHERE {
  ?x rdf:type mebase:User .
  ?x foaf:homepage ?homepage .
  ?x foaf:mbox ?mbox
}
```

[Run]
[Hide Example Results]

homepage	mbox
http://www.linkedin.com/in/drneilravenhill	mailto:neilr@alto-marketing.com
http://www.msu.edu/~lampecli	mailto:lampecli@msu.edu
http://pbeltrao.blogspot.com	mailto:pedrobeltrao@gmail.com
http://bibliogum.wordpress.com/	mailto:bibliogum@yahoo.fr
http://www.leaphish.com/102/	mailto:jhnkelly010@gmail.com
http://www.informatics.sussex.ac.uk/escience-usability	mailto:hilarys@sussex.ac.uk
http://www.science.uva.nl/~adam	mailto:adam@science.uva.nl
http://www.rug.nl/staff/m.a.swertz	mailto:m.a.swertz@rug.nl
http://www.ucps.k12.nc.us	mailto:tom.moncrief@ucps.k12.nc.us
http://libresoft.es	mailto:israel.herraiz@urjc.es
http://ianfoster.typepad.com	mailto:foster@mcs.anl.gov

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX mebase: <http://rdf.myexperiment.org/ontologies/base/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?homepage ?mbox
WHERE {
  ?x rdf:type mebase:User .
  ?x foaf:homepage ?homepage .
  ?y foaf:mbox ?mbox
}
```

As like the query above it is not unusual to have the same subject multiple times over. A semi-colon (;) rather than a full-stop (.) can be used after each triple to replace the subject for the next triple if it is the same, leaving you only needing to define the predicate and object. This is useful because it helps reduce the chances typos like the one previously described. The query below is the same as the previous query but uses this shorthand syntax:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX mebase: <http://rdf.myexperiment.org/ontologies/base/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?homepage ?mbox
WHERE {
  ?x rdf:type mebase:User ;
     foaf:homepage ?homepage ;
     foaf:mbox ?mbox
}
```

[Run]
[Hide Example Results]

homepage	mbox
http://www.linkedin.com/in/drneilravenhill	mailto:neilr@alto-marketing.com

http://www.msu.edu/~lampecli	mailto:lampecli@msu.edu
http://pbeltrao.blogspot.com	mailto:pedrobeltrao@gmail.com
http://bibliogum.wordpress.com/	mailto:bibliogum@yahoo.fr
http://www.leaphish.com/102/	mailto:jhnkelly010@gmail.com
http://www.informatics.sussex.ac.uk/escience-usability	mailto:hilarys@sussex.ac.uk
http://www.science.uva.nl/~adam	mailto:adam@science.uva.nl
http://www.rug.nl/staff/m.a.swertz	mailto:m.a.swertz@rug.nl
http://www.ucps.k12.nc.us	mailto:tom.moncrief@ucps.k12.nc.us
http://libresoft.es	mailto:israel.herraiz@urjc.es
http://ianfoster.typepad.com	mailto:foster@mcs.anl.gov

Sometimes as well as having the same subject multiple times over you may also have the same predicate. in this case you can use a comma (,) to separate each object. Another way to save time writing your query is to use 'a' rather than rdf:type to specify the type of a particular entity.

```
BASE <http://www.myexperiment.org/>
PREFIX ore: <http://www.openarchives.org/ore/terms/>
PREFIX mepack: <http://rdf.myexperiment.org/ontologies/packs/>

SELECT ?pack
WHERE {
  ?pack a mepack:Pack ;
        ore:aggregates <workflows/181>, <workflows/246>
}
```

[Run]
[Hide Example Results]

pack
http://www.myexperiment.org/packs/1
http://www.myexperiment.org/packs/47

4.1. UNION

The *UNION* clause allows you to return results where you want to match multiple patterns. An example of this may be returning all comments and ratings for a particular workflow:

```
BASE <http://www.myexperiment.org/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX meannot: <http://rdf.myexperiment.org/ontologies/annotations/>
PREFIX mebase: <http://rdf.myexperiment.org/ontologies/base/>
SELECT ?annotation ?annotator
WHERE{
  ?annotation mebase:annotates <workflows/72> .
  { ?annotation rdf:type meannot:Comment }
    UNION { ?annotation rdf:type meannot:Rating } .
  ?annotation mebase:has-annotator ?annotator
}
```

[Run]
[Hide Example Results]

annotation	annotator
http://www.myexperiment.org/workflows/72/comments/33	http://www.myexperiment.org/users/18
http://www.myexperiment.org/workflows/72/ratings/63	http://www.myexperiment.org/users/283
http://www.myexperiment.org/workflows/72/comments/163	http://www.myexperiment.org/users/18
http://www.myexperiment.org/workflows/72/ratings/132	http://www.myexperiment.org/users/690

4.2. OPTIONAL

There may be ocassions where you want to include additional variables in your search but you still want to return results if there are not values for these variables. An example might be that you want to return the name, homepage and email address for users. Some users may have not set a homepage and/or email address but you still want to know their name:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX sioc: <http://rdfs.org/sioc/ns#>
PREFIX mebase: <http://rdf.myexperiment.org/ontologies/base/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?name ?homepage ?email
WHERE{
  ?user rdf:type mebase:User ;
    sioc:name ?name .
   OPTIONAL { ?user foaf:homepage ?homepage } .
    OPTIONAL { ?user foaf:mbox ?email }
}
```

[Run]
[Hide Example Results]

name	homepage	email
Workflowxdong		
Piculin		
Daniel Kornev	http://blogs.msdn.com/semantics/	mailto:daniel.kornev@microsoft.com
Ramesh kuc		
John locke		
Nikolas		
Nlynch		
kondas		
Jos?? Manuel Rodr??guez	http://www.inab.org	mailto:jmrodriguez@cnio.es
Babo1ug		
Funkyd101		mailto:d.woodhead@gmail.com
Hong harper		mailto:ihatewinter03@yahoo.com

4. WHERE Go Back to Contents Page

B.2.5 FILTER

5. FILTER

The *FILTER* clause is used within the curly parenthesis {} as a subclause of the WHERE clause. As its name suggests, it allows you to filter the results based on certain conditions. Most commonly you may want to filter on text but you can also filter on numbers and dates.

5.1. On Text

An example of a query where you want to filter on text is when you want to find all Taverna workflows (both 1 and 2):

```
PREFIX dcterms: <http://purl.org/dc/terms/>
PREFIX mebase: <http://rdf.myexperiment.org/ontologies/base/>
PREFIX mecontrib: <http://rdf.myexperiment.org/ontologies/contributions/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?workflow
WHERE{
  ?workflow rdf:type mecontrib:Workflow ;
    mebase:has-content-type ?ct .
  ?ct dcterms:title ?ct_title
  FILTER regex(?ct_title,'^taverna','i')
}
```

[Run]
[Hide Example Results]

workflow	ct_title
http://www.myexperiment.org/workflows/167	Taverna 1
http://www.myexperiment.org/workflows/932	Taverna 2 beta
http://www.myexperiment.org/workflows/193	Taverna 1
http://www.myexperiment.org/workflows/874	Taverna 1
http://www.myexperiment.org/workflows/519	Taverna 1
http://www.myexperiment.org/workflows/545	Taverna 1

The *regex* operand allows you to compare two text strings. In the example above this compares the value of ?ct_title with the text string 'taverna'. The caret (^) sign is used to indicate that the string for ?ct_title must start with 'taverna', not just have it somewhere within the string. The 'i' as the third parameter for the regex operand means that the regular expression is case insentive, if you wish it to be case sensitive only the first two parameters are required.

Sometimes you may want to pattern match to a variable that is a URI rather than a literal to do this you need to use the *str* operand to convert the URI into a string. An example of this may be to find all the accepted membership where a group asked a user to join:

```
PREFIX mebase: <http://rdf.myexperiment.org/ontologies/base/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?membership ?requester
WHERE{
  ?membership rdf:type mebase:Membership ;
    mebase:has-requester ?requester ;
    mebase:has-accepter ?accepter ;
    mebase:accepted-at ?accepted_at
  FILTER regex(str(?requester),'Group','i')
}
```

[Run]
[Hide Example Results]

membership	requester
http://www.myexperiment.org/users/30/memberships/96	http://www.myexperiment.org/groups/51
http://www.myexperiment.org/users/2646/memberships/825	http://www.myexperiment.org/groups/195
http://www.myexperiment.org/users/2213/memberships/884	http://www.myexperiment.org/groups/211
http://www.myexperiment.org/users/2611/memberships/745	http://www.myexperiment.org/groups/187
http://www.myexperiment.org/users/13/memberships/721	http://www.myexperiment.org/groups/194

5.2. On Numbers

In some cases you may want to query on a number being greater (or less than) a particular value. The FILTER clause allows inequalities (and equalities) to be defined as criteria for filtering on. An example of this may be to find all the workflows that have a rating that is 4 or greater:

```
PREFIX dcterms: <http://purl.org/dc/terms/>
PREFIX mebase: <http://rdf.myexperiment.org/ontologies/base/>
PREFIX meannot: <http://rdf.myexperiment.org/ontologies/annotations/>
PREFIX mecontrib: <http://rdf.myexperiment.org/ontologies/contributions/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT DISTINCT ?workflow ?ct_title
WHERE{
  ?workflow rdf:type mecontrib:Workflow ;
    mebase:has-content-type ?ct .
  ?ct dcterms:title ?ct_title .
```

```
  ?rating rdf:type meannot:Rating ;
    mebase:annotates ?workflow ;
    meannot:rating-score ?score
  FILTER (?score >= 4)
}
```

[Run]
[Hide Example Results]

http://www.myexperiment.org/workflows/244	Taverna 1
http://www.myexperiment.org/workflows/742	SimileXMLv3
http://www.myexperiment.org/workflows/19	Taverna 1
http://www.myexperiment.org/workflows/133	Taverna 1
http://www.myexperiment.org/workflows/735	Excel 2007 Macro-Enabled Workbook
http://www.myexperiment.org/workflows/90	Taverna 1
http://www.myexperiment.org/workflows/549	Chemistry Plan

You may want to filter on more than one criteria. For the above example you may only want Taverna 1 workflows that are rated at least 4 out of 5:

```
PREFIX dcterms: <http://purl.org/dc/terms/>
PREFIX mebase: <http://rdf.myexperiment.org/ontologies/base/>
PREFIX meannot: <http://rdf.myexperiment.org/ontologies/annotations/>
PREFIX mecontrib: <http://rdf.myexperiment.org/ontologies/contributions/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT DISTINCT ?workflow ?ct_title
WHERE{
  ?workflow rdf:type mecontrib:Workflow ;
    mebase:has-content-type ?ct .
  ?ct dcterms:title ?ct_title .
  ?rating rdf:type meannot:Rating ;
    mebase:annotates ?workflow ;
    meannot:rating-score ?score
  FILTER (?score >= 4 && regex(?ct_title,'^Taverna 1'))
}
```

[Run]
[Hide Example Results]

workflow	ct_title
http://www.myexperiment.org/workflows/173	Taverna 1
http://www.myexperiment.org/workflows/40	Taverna 1
http://www.myexperiment.org/workflows/66	Taverna 1
http://www.myexperiment.org/workflows/235	Taverna 1

5.3. On Dates

Often one of the most useful filters is for finding things that have been added/modified before or after a certain date. You may want to find all the workflows that have been added since the beginning of September 2009:

```
PREFIX mecontrib: <http://rdf.myexperiment.org/ontologies/contributions/>
PREFIX dcterms: <http://purl.org/dc/terms/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
SELECT ?workflow ?added
WHERE {
  ?workflow rdf:type mecontrib:Workflow ;
    dcterms:created ?added
  FILTER ( ?added >= xsd:dateTime('2009-09-01T00:00:00Z') )
}
```

[Run]
[Hide Example Results]

workflow	added
http://www.myexperiment.org/workflows/932	2009-10-20T17:00:59Z
http://www.myexperiment.org/workflows/952	2009-11-16T12:37:25Z
http://www.myexperiment.org/workflows/906	2009-09-08T15:12:57Z
http://www.myexperiment.org/workflows/955	2009-11-16T18:10:54Z
http://www.myexperiment.org/workflows/912	2009-09-15T11:32:52Z

B.2.6 GROUP BY

myexperiment How To SPARQL

Submit Feedback/Bug Report

6. GROUP BY Go Back to Contents Page

6. GROUP BY

The purpose of the *GROUP BY* clause is to allow aggregation over one or more properties. This is particularly useful when you want to use mathematical functions on variables in the SELECT clause. A good example is using COUNT to list how many workflows are owned by each user.

```
PREFIX sioc: <http://rdfs.org/sioc/ns#>
PREFIX mecontrib: <http://rdf.myexperiment.org/ontologies/contributions/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?user (COUNT(?workflow) AS ?no_workflows)
WHERE{
  ?workflow rdf:type mecontrib:Workflow .
  ?workflow sioc:has_owner ?user .
}
GROUP BY ?user
```

[Run] [Hide Example Results]

http://www.myexperiment.org/users/6890	3
http://www.myexperiment.org/users/36	2
http://www.myexperiment.org/users/10173	1
http://www.myexperiment.org/users/884	2
http://www.myexperiment.org/users/87	1
http://www.myexperiment.org/users/8674	1
http://www.myexperiment.org/users/12835	12
http://www.myexperiment.org/users/7486	1
http://www.myexperiment.org/users/4533	11
http://www.myexperiment.org/users/83	1

The *GROUP BY* clause can also be used with the SUM function to for example get the total number of downloads for all the workflows owned by each user.

```
PREFIX mecontrib:
PREFIX mevd:
PREFIX sioc:
PREFIX rdf:
SELECT ?user (SUM(?downloaded) AS ?total_downloads)
WHERE{
  ?workflow rdf:type mecontrib:Workflow ;
    sioc:has_owner ?user ;
    mevd:downloaded ?downloaded
}
GROUP BY ?user
```

[Run] [Hide Example Results]

user	total_downloads
http://www.myexperiment.org/users/6890	351
http://www.myexperiment.org/users/36	520
http://www.myexperiment.org/users/10173	359
http://www.myexperiment.org/users/884	792
http://www.myexperiment.org/users/87	667
http://www.myexperiment.org/users/8674	60

Again, the *GROUP BY* clause can also be used with the AVG, MAX and MIN functions to get for example the average, maximum and minium ratings of workflows.

```
PREFIX mebase: <http://rdf.myexperiment.org/ontologies/base/>
PREFIX meannot: <http://rdf.myexperiment.org/ontologies/annotations/>
PREFIX mecontrib: <http://rdf.myexperiment.org/ontologies/contributions/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?workflow (AVG(?rating_score) AS ?avg_rating) (MAX(?rating_score) AS ?max_rating)
  (MIN(?rating_score) AS ?min_rating)
WHERE {
  ?workflow rdf:type mecontrib:Workflow .
  ?rating rdf:type meannot:Rating ;
    mebase:annotates ?workflow ;
    meannot:rating-score ?rating_score
}
GROUP BY ?workflow
```

[Run] [Hide Example Results]

>

workflow	avg_rating	max_rating	min_rating
http://www.myexperiment.org/workflows/240	5	5	5
http://www.myexperiment.org/workflows/761	4.999999999999999	5	5
http://www.myexperiment.org/workflows/1402	4	4	4

http://www.myexperiment.org/workflows/20	3.9999999999999992	5	3
http://www.myexperiment.org/workflows/6	4.4999999999999991	5	4
http://www.myexperiment.org/workflows/688	4	4	4

N.B. Currently (March 9th 2011) 4Store does not properly handle floating point numbers properly so average results may need to be manually rounded to make for example 4.49999999999999991 into 4.5.

6. GROUP BY Go Back to Contents Page

B.2.7 ORDER BY

myexperiment How To SPARQL

Submit Feedback/Bug Report

◀◀ ◀ 7. ORDER BY Go ▶ ▶▶ Back to Contents Page

7. ORDER BY

The filter on dates example restricts the workflows returned, however you may want to go further, listing the most recent workflows first. The *ORDER BY* clause can be used to do this:

```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX dcterms: <http://purl.org/dc/terms/>
PREFIX mecontrib: <http://rdf.myexperiment.org/ontologies/contributions/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?workflow ?added
WHERE {
  ?workflow rdf:type mecontrib:Workflow ;
    dcterms:created ?added
  FILTER ( ?added >= xsd:dateTime('2009-09-01T00:00:00Z') )
}
ORDER BY DESC(?added)
```

[Run]
[Hide Example Results]

workflow	added
http://www.myexperiment.org/workflows/1008	2009-12-22T20:45:54Z
http://www.myexperiment.org/workflows/1005	2009-12-15T22:33:09Z
http://www.myexperiment.org/workflows/1004	2009-12-15T22:17:56Z
http://www.myexperiment.org/workflows/1003	2009-12-15T22:17:11Z
http://www.myexperiment.org/workflows/1002	2009-12-15T22:16:23Z

The *DESC* operand means order the results in descending order, in this case, the latest workflows first. If you wanted them ordered ascending you don't require any operand. If you want a second criteria to order on, you just add this after the first. An example of this might be order by workflow type with the latest of each type first:

```
PREFIX mebase: <http://rdf.myexperiment.org/ontologies/base/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX dcterms: <http://purl.org/dc/terms/>
PREFIX mecontrib: <http://rdf.myexperiment.org/ontologies/contributions/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?workflow ?added ?ct_title
WHERE {
  ?workflow rdf:type mecontrib:Workflow ;
    dcterms:created ?added ;
    mebase:has-content-type ?ct .
  ?ct dcterms:title ?ct_title
  FILTER ( ?added >= xsd:dateTime('2009-09-01T00:00:00Z') )
}
ORDER BY ?ct_title DESC(?added)
```

[Run]
[Hide Example Results]

workflow	added	ct_title
http://www.myexperiment.org/workflows/36	2010-01-12T13:58:46Z	Taverna 1
http://www.myexperiment.org/workflows/15	2009-12-04T16:04:38Z	Taverna 1
http://www.myexperiment.org/workflows/994	2009-12-04T10:47:04Z	Taverna 1
http://www.myexperiment.org/workflows/1005	2009-12-15T22:33:09Z	Taverna 2 beta
http://www.myexperiment.org/workflows/1004	2009-12-15T22:17:56Z	Taverna 2 beta

By default, *ORDER BY* will treat the field(s) it is ordering on as alphanumeric. In some cases it is necessary to treat a field as being numeric. An example of this is a list of the most downloaded workflows. Below is an example of how to treat *? downloaded* as a numeric field.

```
BASE <http://www.myexperiment.org/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX mevd: <http://rdf.myexperiment.org/ontologies/viewings_downloads/>
PREFIX sioc: <http://rdfs.org/sioc/ns#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX mecontrib: <http://rdf.myexperiment.org/ontologies/contributions/>

SELECT ?workflow ?downloaded
WHERE {
  ?workflow rdf:type mecontrib:Workflow .
  ?workflow sioc:has_owner <users/43> .
  ?workflow mevd:downloaded ?downloaded
}
ORDER BY DESC(xsd:nonNegativeInteger(?downloaded))
```

[Run]
[Hide Example Results]

Alphanumeric Ordering

workflow	downloaded
http://www.myexperiment.org/workflows/1392	99
http://www.myexperiment.org/workflows/12	988

Numeric Ordering

workflow	downloaded
http://www.myexperiment.org/workflows/140	7852
http://www.myexperiment.org/workflows/10	3250

http://www.myexperiment.org/workflows/95	985	http://www.myexperiment.org/workflows/16	2918
http://www.myexperiment.org/workflows/1215	98	http://www.myexperiment.org/workflows/72	2631
http://www.myexperiment.org/workflows/1105	98	http://www.myexperiment.org/workflows/15	2542
http://www.myexperiment.org/workflows/1224	98	http://www.myexperiment.org/workflows/124	2303
http://www.myexperiment.org/workflows/1246	98	http://www.myexperiment.org/workflows/154	2162
http://www.myexperiment.org/workflows/1315	98	http://www.myexperiment.org/workflows/158	2008
http://www.myexperiment.org/workflows/75	978	http://www.myexperiment.org/workflows/79	1788
http://www.myexperiment.org/workflows/39	976	http://www.myexperiment.org/workflows/19	1617

7. ORDER BY Go Back to Contents Page

B.2.8 LIMIT

myexperiment How To SPARQL

Submit Feedback/Bug Report

8. LIMIT Go Back to Contents Page

8. LIMIT

Sometimes you may not want all possible results. The *LIMIT* clause allows you to limit how many results are returned. In the examples used in FILTER and ORDER BY you may only want to the latest 10 workflows:

```
PREFIX dcterms: <http://purl.org/dc/terms/>
PREFIX mecontrib: <http://rdf.myexperiment.org/ontologies/contributions/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?workflow ?added
WHERE {
  ?workflow rdf:type mecontrib:Workflow ;
    dcterms:created ?added
}
ORDER BY DESC(?added)
LIMIT 10
```

[Run]
[Hide Example Results]

workflow	added
http://www.myexperiment.org/workflows/1010	2010-01-12T13:58:46Z
http://www.myexperiment.org/workflows/1009	2010-01-04T17:42:36Z
http://www.myexperiment.org/workflows/1008	2009-12-22T20:45:54Z
http://www.myexperiment.org/workflows/1005	2009-12-15T22:33:09Z
http://www.myexperiment.org/workflows/1004	2009-12-15T22:17:56Z
http://www.myexperiment.org/workflows/1003	2009-12-15T22:17:11Z
http://www.myexperiment.org/workflows/1002	2009-12-15T22:16:23Z
http://www.myexperiment.org/workflows/1001	2009-12-15T22:15:21Z
http://www.myexperiment.org/workflows/1000	2009-12-15T22:14:39Z
http://www.myexperiment.org/workflows/999	2009-12-15T22:13:44Z

8.1 OFFSET

Once you have got the first 10 workflows you might want to get the next 10. The *OFFSET* clause allows you to do this:

```
PREFIX dcterms: <http://purl.org/dc/terms/>
PREFIX mecontrib: <http://rdf.myexperiment.org/ontologies/contributions/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?workflow ?added
WHERE {
  ?workflow rdf:type mecontrib:Workflow ;
    dcterms:created ?added
}
ORDER BY DESC(?added)
LIMIT 10
OFFSET 10
```

[Run]
[Hide Example Results]

workflow	added
http://www.myexperiment.org/workflowss/998	2009-12-15T22:12:50Z
http://www.myexperiment.org/workflows/997	2009-12-15T22:12:00Z
http://www.myexperiment.org/workflows/996	2009-12-15T22:10:36Z
http://www.myexperiment.org/workflows/995	2009-12-04T16:04:38Z
http://www.myexperiment.org/workflows/994	2009-12-04T10:47:04Z
http://www.myexperiment.org/workflows/993	2009-12-01T06:56:54Z
http://www.myexperiment.org/workflows/992	2009-12-01T06:28:26Z
http://www.myexperiment.org/workflows/991	2009-12-01T06:24:54Z
http://www.myexperiment.org/workflows/990	2009-12-01T04:08:08Z
http://www.myexperiment.org/workflows/989	2009-12-01T03:01:28Z

If you then want the third set of 10 you can change to *OFFSET 20*. The OFFSET clause can be used without the LIMIT clause, by removing the LIMIT clause in the example above you will get all but the 10 latest workflows.

8. LIMIT Go Back to Contents Page

B.2.9 Troubleshooting

myexperiment **How To SPARQL**

Submit Feedback/Bug Report

9. Troubleshooting Go Back to Contents Page

9. Troubleshooting

When trying to execute a query you may get one or more warning/error messages. If you are using the query form with *HTML Table* results these messages will be shown in a red box just above the query text box. E.g.

parser warning: Variable q was selected but is unused in the query. at line 1

If you request raw SPARQL results XML these messages will appear as within a comment tag <!-- --> in the XML itself. Usually between the *head* and *results* tags.

```
<?xml version="1.0"?>
<sparql xmlns="http://www.w3.org/2005/sparql-results#">
  <head>
    <variable name="q"/>
  </head>
<!-- parser warning: Variable q was selected but is unused in the query. at line 1 -->
  <results>
  </results>
</sparql>
```

Generally these messages are fairly self explanatory but here are some tips for how you might go about resolving them.

8.1. Syntax Errors

Error messages are generally syntactical errors in the query itself. Commonly this is because the the clauses have not be defined in the right order. Thtype of error message you receive may look something like this:

parser error: syntax error, unexpected WHERE, expecting AS at line 6

Below is a rough guide for how the clauses that have been described in this tutorial should be ordered. (Other ordering may work):

```
BASE
PREFIX
SELECT
WHERE {
  UNION / OPTIONAL
  FILTER
}
GROUP BY
ORDER BY
LIMIT
OFFSET
```

A second common syntactical error is to fail to put a dot between triples in the WHERE clause. This is why it is often good practice to start a new line for each triple and ident appropriately when using the semi-colon (;) operator. E.g.

```
PREFIX dcterms: <http://purl.org/dc/terms/>
PREFIX mebase: <http://rdf.myexperiment.org/ontologies/base/>
PREFIX meannot: <http://rdf.myexperiment.org/ontologies/annotations/>
PREFIX mecontrib: <http://rdf.myexperiment.org/ontologies/contributions/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT DISTINCT ?workflow ?ct_title
WHERE{
  ?workflow rdf:type mecontrib:Workflow ;
    mebase:has-content-type ?ct .
  ?ct dcterms:title ?ct_title .
  ?rating rdf:type meannot:Rating ;
    mebase:annotates ?workflow ;
    meannot:rating-score ?score
  FILTER (?score >= 4)
}
```

SPARQL queries often use several levels of parentheses either curly parentheses {} in WHERE, OPTIONAL and UNION clauses or round parentheses () in FILTER clauses. If you get an error message like the one below you should check that all your parentheses pair up.

parser error: syntax error, unexpected $end, expecting '}' at line 11

8.2. Complexity Warnings

If you run a query that is fairly complex (e.g. lots of triples, many OPTIONAL / UNION clauses, complicated FILTER clauses, etc.) and is expected to return quite a lot of results, you may receive a complexity warning message like the one below:

warning: hit complexity limit 8 times, increasing soft limit may give more results

Usually you will receive a number of results but the query engine cannot guarantee it has returned all of them. This can usually be resolved by increasing the Soft Limit to 5%, 10%, 20%, etc. until you cease to get this warning message. Sometimes you may receive a complexity warning because there is a semantic error in your query, such as typo for one of your variable names which means your triples don't link together in the way you intended.

8.3. Parser Warnings

Parser warnings like the one shown at the top of this page, mean that the query was syntactically correct and able to execute but there may be some mistake in your query causing you not to get the results you expected. The error message below is a classic example of where a variable name in the SELECT clause does not match up with a variable in the WHERE clause:

parser warning: Variable nmae was selected but is unused in the query. at line 1

9. Troubleshooting Go Back to Contents Page

B.3 myExperiment Ontology Change Log

As of 16/10/2011.

12/10/2011 Changed mebase:User to be equivalent to a SIOC UserAccount rather than a SIOC User that no longer exists in the current specification. A new mebase:Person class and migration of person based triples to this class will be performed shortly.

03/10/2011 Corrected rdfs:label for Tagging as it was just set to Tag in Annotations module. Corrected wrong namespace for Entry when defining PackEntry as a subClassOf it in Packs module. Also modified all of the ontology module descriptors to include owl:versionInfo, so that dc:date captures the original modularization date of the ontology and owl:VersionInfo captures the last modification date of the ontology module.

02/09/2011 Corrected rdfs:label for RestrictedAccess as it was just set to Access.

18/07/2011 Replaced PackRelationship with RelationshipEntry as a more generic way of associating a Relationship with an Aggregation. Made Entry class to hang PackEntry and RelationshipEntry off as subclasses.

03/07/2011 Removed Concepts from packs module, predicates of relationships now defined as OWL ObjectProperties that make up user-defined ontologies. Vocabulary moved to annotations module to be used to represent at set of tags.

09/03/2011 Updated Packs module to create a separate Relationship class that has PackRelationship as an ore:Proxy to describe it in the context of the Pack. Fixed some other subclassing issues with PackEntry and Vocabulary.

10/02/2011 Added PackRelationship to Packs module to allow relationship between items in Packs to be defined. Moved Vocabularies from Contributions to Packs module to support PackRelationships. Added Concepts to Packs module, as a specific form of SKOS Concept to define the predicate of a PackRelationship.

25/11/2010 Added foaf:name to user as more commonly used than sioc:name and added restriction for sioc:has_member and sioc:member_of to Group and User respectively.

15/09/2010 Added DBPedia's residence property as potential field(s) for a User.

01/07/2010 Added various has-x properties to allow users in the Linked Data world to more easily navigate around myExperiment entities.

01/07/2010 Migrated RDF to use Linked Data Non-Information Resource URIs. This has also involved providing information about the RDF document itself and not just the entity it is describing.

01/07/2010 Consensus seems that xsd:anyURI should only be used if the resource is non-resolvable therefore have switched to unrestricted ObjectProperty where URIs can be resolved. Also removed mebase:human-start-page as it is covered by the foaf:homepage in the description of the graph.

19/05/2010 Added ranges to properties reused from non-myExperiment ontologies.

05/04/2010 Made Vocabulary a subclass of SKOS ConceptScheme and Tag a subclass of Concept changing it's dcterms property to skos:prefLabel.

31/03/2010 Removed accessed-from-site as this can be inferred from user-agent. Also removed retroactive assignment of human-start-page as a property restriction for Policy as there is currently no such thing exists

30/03/2010 Made Version a subclass of Interface rather than Contribution.

24/03/2010 Removed Use AccessType and associated Accesses from specific module as they are unused.

24/03/2010 Fixed syntax error changing foaf:based-near to foaf:based_near. Added Commentable as a subclass to File and Commentable, Favouritable and Taggable to Pack and by inheritance Experiment. Added Taggable and Commentable as a subclass to Group retroactively in the specific module.

22/03/2010 Removed cc:License definition in the specific module as these have been replaced by myExperiment's License class for which instances can be found at `http://rdf.myexperiment.org/Licenses`.

21/03/2010 Added dbpedia ontology's residence property to Users. This essentially uses DBPedia URIs rather than text strings and should iin time replace the foaf:based_near and mebase:country properties.

03/02/2010 Anonymised usage statistics (i.e. Viewings and Downloads) are to no longer be generated. This is due to the excessive amount of processor time taken to keep this up to date versus their actual usefulness.

12/01/2010 The Components module has been completly rewritten to encompass Dataflows and to represent extra data exposed through the new Taverna GEMs.

28/09/2009 Removed has-license property from Base module, cc:license should be used instead.

26/08/2009 To improve Linked Data compliance object properties have been changed to anyURI typed datatype properties where the object of these properties is not intended to resolve to RDF. Also a file extension has been added to the filename property for workflows and workflow versions.

22/06/2009 Converted mebase:uri from anyURI datatype property to an object property and replaced mepack:alternate-uri with rdfs:seeAlso.

19/06/2009 Upgraded Jobs to Contributions to support ORE Aggregation capability.

26/05/2009 Removed dcterms:type from Workflows, WorkflowVersions and Files to use mebase:has-content-type. ContentType contains dcterms:title as a human-readable label for the type and dcterms:format for the MIME type.

Glossary

4Store A triplestore application developed by Garlik. Originally developed as 3Store at the University of Southampton.

ABCDE Annotations, Background, Contribution, Discussion and Entities. A discourse model proposed by (de Waard and Tel, 2006).

AEDS Automated Experiment Driver Software. Software that allows a user to manage the analysis of data captured from a chemical sample.

AKT Advanced Knowledge Technologies. A six year multi-institution project to investigate technologies for the "capture, modelling, publishing, reuse and management of knowledge".

AllegroGraph A modern, high-performance, persistent RDF graph database, capable of scaling to billions of triples.

Angelfire A web-hosting service that facilitated the development of virtual communities.

ANNIE A Nearly-New Information Extraction system. A tool built using GATE by combining CREOLE modules.

Annotea A system that allows metadata annotations in RDF format to be made about a web document, external to the document itself.

AO Annotation Ontology. An ontology for annotating scientific documents on the web.

APACE Application Advanced Computing Exchange.

Apache A well-known HTTP (Web) server application.

ASP Active Server Pages. A Web-scripting language.

aTag Associative tags. "Snippets of HTML that capture the information that is most important to you in a machine-readable, interlinked format".

ATN Augmented Transition Network. A type of transformational grammar that is a modified non-deterministic pushdown automata for traversing finite-state grammars capable of supporting transformational rules.

Atom an XML-based document format that describes lists of related information known as 'feeds'.

BFO Basic Formal Ontology. Focused on the task of providing a genuine upper ontology which can be used in support of domain ontologies developed for scientific research.

BIBO Bibliographic ontology. An ontology for modelling bibliographic data.

Boundary Object From the field of Sociology. A means of sharing research entities amongst a heterogeneous group of researchers with different viewpoints and understandings.

CERN The European Organisation for Nuclear Research.

CiTO Citation Typing Ontology. An ontology for modelling bibliographic data.

Classmates.com An early US social networking site designed for rediscovering old school friends.

CoAKTinG Collaborative Advanced Knowledge Technologies in the Grid.

Comb*e*Chem One of the original UK e-Science platform projects. Intended primarily to support chemists manage the large amounts of data produced through combinatorial methods.

Corpus A collection of natural language text that can be parsed.

Creative Commons A project that provides "free licenses and legal tools" to allow content creators to licence their work "so others can share, remix and use commercially".

CREOLE Collection of REusable Objects for Language Engineering. A library that encapsulates pre-existing or user-defined tools and resources, so they can be used within GATE.

CWA Concept Web Alliance. A group "devoted to the Concept Web: a dynamic, interactive fabric of concepts and their relationships".

DAML DARPA Agent Markup Language. Has been amalgamated with OIL to produce OWL.

DBPedia A project with the aim of producing a machine-readable transcription of Wikipedia.

DCMI Dublin Core Metadata Initiative. An organisation set up to define standards, vocabularies and practices for metadata.

Deep Structure The most basic form of the sentence that effectively stores the semantic meaning. Can be used to generate more sophisticated sentences through the application of transformational rules.

Diaspora* An alternative to social networking sites like Facebook. Using a distributed network of Web server 'seeds', it aims to provide clear, contextual interfaces to give users complete control over who they share their content with.

DLG Directed Labelled Graph.

DoCO Document Component Ontology. An ontology "describing the component parts of a bibliographic document." Part of the SPAR ontologies suite.

Dublin Core A set of vocabularies for metadata, defined in RDF Schema.

e-Laboratories A number of VRE and similar projects run at the University of Manchester and/or University of Southampton.

eCrystals archive An archive for storing the results and analysis of chemical crystal samples.

EFO Experimental Factor Ontology.

EliCIT A web-based knowledge elicitation system from planned experiments. Part of the Comb*e*Chem.

ELIZA A fairly simple system written in Lisp designed to impersonate a psychiatrist in an attempt to pass the Turing test.

ELN Electronic Lab Notebook. Digital replacement for a paper-based lab book.

Facebook A social networking website, which as of 2009 held one of the greatest market shares.

Flash Mob An organised event where a large group of people suddenly assemble together in a public place to perform an unusual act before dispersing.

FOAF Friend Of A Friend. A project to define machine-readable profile documents, describing the person concerned and amongst other things the people they know.

Forum Also known as a message board. An online discussion site where people can hold conversations in the form of posted messages.

FRBR Functional Requirements for Bibliographic Records.

FriendFeed A social networking site that supports FOAF profile documents.

Friends Reunited One of the first social networking sites to become popular in the UK. Designed for rediscovering old school friends and based on the US site Classmates.com.

GATE General Architecture for Text Engineering. An architecture allowing resources for processing natural language to be pieced together to build a workflow that could be executed over some text.

Gem Packages of Ruby code that can package information along with the files to install.

GeoCities A web-hosting service that facilitated the development of virtual communities.

GINSENG Guided Input Natural language Search ENGine. A Question-Answering system that uses Semantic Web technologies.

GO The Gene Ontology. "A major bioinformatics initiative with the aim of standardizing the representation of gene and gene product attributes across species and databases".

GPSG Generalized Phrase Structure Grammar. An extension of Chomsky's Transformational Grammar with a greater emphasis on the semantics of the natural language.

Graphviz A open source piece of software for graph visualisation.

GUI Graphical User Interface.

HCLS Healthcare and Life Sciences. An Interest Group of the W3C.

HPC High Performance Computing. Systems capable of processing large amounts of data or running complex simulations.

HPC/NA High Performance Computing / Numerical Analysis.

HPSG A type of transformational grammar based on GPSG but provide a much simpler context free grammar, at the expense of a much more complex lexicon.

IANA Internet Assigned Numbers Authority. Responsible for the global coordination of the DNS Root, IP addressing, and other Internet protocol resources.

IDE Integrated Developer Environment.

IETF Internet Engineering Task Force. Has the goal of making the Internet work better.

in silico An adjective to describing a process or workflow that is performed on a computer.

in vitro Experimentation not carried out on living organism but within controlled conditions. Also known as ex vivo.

in vivo Experimentation within or upon a living organism.

IT Information Technology.

JAPE Java Annotation Patterns Engine. Rule builder for document annotation with tags.

Jena A triplestore application and Java RDF library.

JISC Joint Information Systems Committee. A UK research funding body.

JSON JavaScript Object Notation. "A lightweight text-based open standard designed for human-readable data interchange".

KEfED Knowledge Engineering for Experimental Design. A model that allows for the representation of the pieces of data that are part of a scientific experiment. Allowing relationships to be built up between the different pieces of data and the other components that make up the experiment.

Kepler A free and open source scientific workflow application.

Kowari A massively scalable "metastore" database for storing RDF and OWL metadata.

LFG Lexical Functional Grammar. A transformational grammar with two levels of syntactic description, constituent structures and functional structure. The former represent the phrase structure grammar, the latter the grammatical functions of the sentence using a feature structure that captures the phrasal category, plurality, person, agreement, subject, object, predicate and tense.

Linked Data A set of best practices for publishing and connecting structured data on the Web.

LiveJournal A social networking site that supports FOAF profile documents.

MAGE MicroArray and Gene Expression.

MCF Meta Content Framework. Part of the inspiration for RDF.

MGED Microarray Gene Expression Data.

MIBBI Minimum Information for Biological and Biomedical Investigations. A metadata standard.

Middleware A piece of software that allows two or more other software components to interact with each other.

Mongrel A HTTP (Web) server application for delivering Web pages written in Ruby-on-Rails.

MVC Model-View-Controller. A software design architecture.

^{my}**Grid** A project to produce a suite of tools designed to "help e-Scientists get on with science and get on with scientists." One of first UK e-Science platform projects.

MySpace A social networking website, which as of 2009 held one of the greatest market shares.

MySQL An implementation of a Relational DataBase Management System (RDBMS).

myTea A project that developed the common world analogy of putting together a jigsaw to help understand the work undertaken by bioinformaticians..

n-gram A sequence of n consecutive words.

N3 Notation3. A concrete syntax for RDF that is more compact and readable than XML.

Nano-publication The idea of publishing each research assertion as a separate machine-readable publication.

NCS National Crystallography Service. A UK service for the analysis of chemical samples based at the University of Southampton.

NeuroScholar A project for capturing neuroscience experiments. One of the first projects to define a machine-readable representation of experiments for use in scientific discourse.

NL Natural Language. Typical language used day-to-day for reading and writing, such as English, French, etc.

NLP Natural Language Processing. The process of parsing and evaluating natural language to produce machine-readable representation of grammatical structure and semantic content of the text..

NTriple An alternative concrete syntax to XML for RDF. A fixed subset of N3.

OAC Open Annotations Collaboration. A data model to allow annotations to be assigned to resources, drawing significantly on the Annotea model.

OAI-ORE Open Archives Initiative Object Reuse and Exchange protocol. Defines "standards for the description and exchange of aggregations of Web resources".

OAI-PMH Open Archives Initiative Protocol for Metadata Harvest.

OAuth A means for third party (consumer) applications to register and authenticate themselves with the primary application. Allowing them to make full use of the primary application's REST API.

OBI Ontology for Biomedical Investigations.

OBO Open Biological and Biomedical Ontologies.

OCML Operational Conceptual Model Language.

OGSA Open Grid Services Architecture. An architecture for describing service-oriented grid computing.

OIL Ontology Interchange Language. Has been amalgamated with DAML to produce OWL.

OpenID An open standard for supporting user authentication in a decentralised manner.

OpenWetWare An e-Science community wiki that mandates that all content deposited must be open to the whole community.

ORB Ontology of Rhetorical Blocks. An ontology module aligning SWAN with SALT, ABCDE and other models containing an element of coarse-grained rhetorical structure.

oreChem A project providing semantic representations of research, as well as developing and deploying infrastructure, services, and applications to support these representations.

OWA Open World Assumption. The concept that unless something is explicitly stated it must be assumed to both true and false. E.g. just because a query returns no results it does mean that data does not exist that fulfills the criteria of that query.

OWL The Web Ontology Language. An evolution of DAML+OIL, capable of representing logical relationships between different concepts.

OWL-S Web Ontology Language for Services. Language to semantically markup web services.

PCFG Probabilistic Context Free Grammar. A grammar that uses probabilistic values to determine the most likely syntactic structure of a sentence; in contrast to discrete parameter used by purely transformational grammars. Also known as Stochastic Context-Free Grammars (SCFGs).

PHP PHP Hypertext Preprocessor. A Web-scripting language.

Plugin A set of software components that adds specific capabilities to a larger software application.

POS Parts Of Speech. Such as nouns, verbs, adjectives, etc.

PostgreSQL An implementation of a Relational DataBase Management System (RDBMS).

Principle-based Grammars Type of transformational grammar. Includes Government and Binding (GB) and the Minimalist Program. Instead of using rules for how to precisely generate sentences, they use principles which specify well-formedness conditions.

PRISM Publishing Requirements for Industry Standard Metadata. A specification for managing bibliographic data.

Protégé A free, open source ontology editor and knowledge-base framework.

PSI Proteomics Standards Initiative.

Question-Answering (QA) System A system that allows the user to obtain answers to natural language questions they submit.

RDF Resource Description Framework. Conceptually a triple joining a subject to an object via a predicate.

RDF/XML RDF formatted in XML.

RDFS Resource Description Framework Schema.

RDQL Resource Description Query Language. A triplestore query language.

Research Object a means of publishing a machine-readable representation of a person's research output, as opposed to a human-readable one in the form of a conference paper of journal article.

REST REpresentational State Transfer. A software architecture for distributed hypermedia systems such as the Web.

RO See Research Objects.

RODS Research Object Domain Schema for defining domain-specific RO types, properties and Relationship predicates.

ROUM Research Object Upper Model for defining basic RO types, properties and Relationship predicates.

RQL Resource Query Language. A triplestore query language.

RST Rhetoric Structure of Text. A theory "intended to describe texts, rather than the processes of creating or reading and understanding them.".

SALT Semantically Annotated LaTeX. A discourse model proposed by (Groza et al., 2007).

sameAs A tool for looking OWL sameAs relationships for a particular URI.

SCARF SCientific ARticle of The Future. The ultimate goal of the alignment of the SWAN ontology with other ontologies in scientific discourse to provide a specification for a machine-readable and semantically rich version of the existing research publication.

ScholOnto An ontology-based hypertext argumentation system.

Schools Malaria project A collaboration between the Comb*e*Chem and the e-Malaria project.

Scientific Discourse The argumentation of competing claims and hypotheses to produce theories.

SCUFL Simple Conceptual Unified Flow Language. An XML format for defining Taverna workflows.

SeRQL Sesame's Resource Query Language.

Sesame A triplestore application that can support multiple back-end storage mechanisms.

Sindice The Semantic Web Index. A website that allows users to search over SW documents.

SIOC Semantically-Interlinked Online Communities. A project that aims to provide "the main concepts and properties required to describe information from online communities", including weblogs, messageboards and wikis.

Six Degrees One of the first social networking websites. Its model was based on the "six degrees of separation" concept first proposed by Frigyes Karinthy in 1929.

SKOS Simple Knowledge Organisation System. A schema designed to provide a means for representing various types of concept schemes in RDF.

SKUA Semantic Knowledge Underpinning Astronomy.

SmartTea A project to determine the best means of eliciting user requirements from the chemists for the ELN.

SNARM Simple Network Access Rights Management. An ontology written for myExperiment that allows access rights to be assigned based on the relationship between users in a simple, commonly social, network.

SOAP Simple Object Access Protocol. XML-based protocol for exchanging structured information.

SocialNet One of the first social networking websites.

Solr A Java application that uses the Lucene Java search library to provide full-text indexing and search.

SPAR Semantic Publishing And Referencing ontology suite.

SQL Structured Query Language. The standard language for query relational databases.

Surface Structure Derived from a deep structure through the application of transformational rules, allowing complex sentences to be defined.

SVG Scalable Vector Graphics. An XML-based file format for describing two dimensional vector graphics.

SWAN Semantic Web Applications in Neuromedicine.

SWRL Semantic Web Rule Language. The integration of the OWL Lite and OWL DL sublanguages with the Rules Markup Language (RuleML) to allow rules to be defined.

TAG Tree Adjoining Grammar. A mildly context-sensitive transformational grammar. Similar to a context-free grammar but instead of having production rules, it has initial and auxiliary trees. Grammatical representation of the sentence is achieved through adjoining auxiliary trees or substitution of non-terminal leaf nodes with initial or derived initial trees.

TAMBIS Transparent Access to Multiple Bioinformatics Information Sources. A precursor to the $^{my}Grid$ project.

Taverna An open source and domain independent Workflow Management System.

Taverna Workbench A free software tool that provides an environment for designing and executing workflows.

TeLQAS Telecommunication Literature Question-Answering System. An experimental domain-specific QA system developed for answering simple English questions in the telecommunication domain.

The Grid The electronic infrastructure required to "facilitate the increasing reliance on collaborative, multidisciplinary research" (Hey and Trefethen, 2002).

Transformational rules Rules that allow complex sentences (known as surface structures) with different tenses, cases or plurality to be derived from a basic sentence (known as a deep structure).

Triana A workflow development environment for problem solving.

Triplestore Semantic Web technologies equivalent of a relational database's data structure.

Tripod.com A web-hosting service that facilitated the development of virtual communities.

TURTLE Terse RDF Triple Language. An alternative concrete syntax to XML for RDF.

Twitter A social networking website, which as of 2009 held one of the greatest market shares.

Unification-based grammar A transformational grammar that takes feature structures and unifies if all corresponding register values in the two feature structures are the same. Starting with basic features this unification process is repeated until a single specific feature structure for the sentence is generated.

URI Uniform Resource Identifier. A standard way of defining character strings for resources when representing them on the Internet.

URN Uniform Resource Name. A non-resolvable URI.

Virtuoso A hybrid data server for relational, RDF-graph, and full text document data management.

VO Virtual Organisation. A dynamic set of individual and/or institutions defined around a set of resource-sharing rules and conditions.

VoID Vocabulary of Interlinked Datasets. a vocabulary for defining a file that contain all the salient information about the data being published by a project as Linked Data.

VRE Virtual Research Environment. A system that allows user to perform research in the electronic (virtual) rather than physical world.

W3C World Wide Web Consortium. An organisation that defines standards for the Web.

WebOnto An application for building ontologies.

Wiki A Web site that allows the easy creation and editing of any number of interlinked web pages via a Web browser using a simplified markup language.

Wordpress A piece of software for running a Web log (blog) server.

WS-BPEL Web Services - Business Process Execution Language. A orchestration language that is part of OASIS.

WSDL Web Services Description Language. XML-based language for describing services on the Web.

XML eXtensible Markup Language. Similar to HTML but for marking up real world objects and not just hypertext.

XQuery A language for querying XML documents.

Bibliography

B. Adida and M. Birkbeck. RDFa Primer. W3C Working Group Note, October 2008. `http://www.w3.org/TR/xhtml-rdfa-primer/` (Last Accessed: 07/04/2010).

M. J. Adler. *Dialectic*. Kegan Paul, Trench and Co Ltd., 1927.

I. Altintas, C. Berkley, E. Jaeger, M. Jones, B. Ludäscher, and S. Mock. Kepler: An Extensible System for Design and Execution of Scientific Workflows. In *SSDBM*, pages 21–23, 2004.

A. Alves, A. Arkin, S. Askay, B. Bloch, F. Curbera, Y. Goland, N. Kartha, C. K. Liu, D. König, V. Mehta, S. Thatte, D. van der Rijn, P. Yendluri, and A Yiu. Web Services Business Process Execution Language Version 2.0. Technical report, OASIS Committee, May 2006.

I. Androutsopoulos, G. D. Ritchie, and P. Thanisch. Natural Language Interfaces to Databases– an introduction. *Journal of Language Engineering*, 1(1):29–81, 1995.

M. Atwood, D. Balfanz, D. Bounds, R. M. Conlan, B. Cook, L. Culver, B. de Medeiros, B. Eaton, K. Elliott-McCrea, L. Halff, E. Hammer-Lahav, B. Laurie, C. Messina, J. Panzer, S. Quigley, D. Recordon, E. Sandler, J. Sergent, T. Sieling, B. Slesinsky, and A. Smith. OAuth Core 1.0 Revision A. Web page, June 2009. `http://oauth.net/core/1.0a/` (Last Accessed: 07/04/2010).

P. Atzeni, G. Mecca, and P. Merialdo. Semistructured and Structured Data in the Web: Going Back and Forth. *ACM SIGMOD Record*, 26(4):16–23, December 1997.

B. Au, D. Cutting, O. Gospodnetić, E. Hatcher, C. Hostetter, G. Ingersoll, M. Klaas, S. S. Mangar, R. McKinley, M. Miller, P. Noble, Y. Seeley, and K. Sekiguchi. Welcome to Solr. Web page, January 2010. `http://lucene.apache.org/solr/` (Last Accessed: 07/04/2010).

S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, and Z. Ives. DBpedia: A Nucleus for a Web of Open Data. In *In 6th International Semantic Web Conference, Busan, Korea*, pages 11–15. Springer, 2007.

M. Bachler, S. Buckingham Shum, J. Chen-Burger, J. Dalton, D. De Roure, M. Eisenstadt, J. G. Frey, J. Komzak, D. Michaelides, K. Page, S. Potter, N. R. Shadbolt, and A. Tate. Chain ReAKTing: Collaborative Advanced Knowledge Technologies in the CombeChem Grid. In S. Cox, editor, *Proceedings of the UK e-Science All Hands Meeting, Nottingham, Sep 2003*, page (6pp). EPSRC, 2004a.

M. Bachler, S. Buckingham Shum, J. Chen-Burger, J. Dalton, D. De Roure, M. Eisenstadt, J. Komzak, D. Michaelides, K. Page, S. Potter, N. R. Shadbolt, and A. Tate. Collaborative

Tools in the Semantic Grid. In Luc Moreau, editor, *GGF11 - The Eleventh Global Grid Forum*, Honolulu, Hawaii, USA, 2004b. Global Grid Forum.

A. Barabási. *Linked: The New Science of Networks*. Perseus Publishing, April 2002.

A. D. Baxevanis. The molecular biology database collection: an online compilation of relevant database resources. *Nucleic Acids Res.*, 28:1–7, 2000.

S. Bechhofer, J. Ainsworth, J. Bhagat, I. Buchan, P. Couch, D. Cruickshank, M. Delderfield, Dunlop I., M. Gamble, C. Goble, D. Michaelides, P. Missier, S. Owen, Newman D., D. De Roure, and S. Sufi. Why Linked Data is Not Enough for Scientists. In *Sixth IEEE e-Science conference (e-Science 2010)*, August 2010a.

S. Bechhofer, D. De Roure, M. Gamble, C. Goble, and I. Buchan. Research Objects: Towards Exchange and Reuse of Digital Knowledge. In *The Future of the Web for Collaborative Science (FWCS 2010)*, February 2010b. Co-located with WWW'10.

S. Bechhofer and R. Volz. Patching Syntax in OWL Ontologies. In *Proceedings of ISWC2004*, 2004.

D. Beckett. The Design and Implementation of the Redland RDF Application Framework. In *WWW10*, 2001.

D. Bedell. Meeting your new best friends Six Degrees widens your contacts in exchange for sampling Web sites. Electronic news article, October 1998. `http://www.dougbedell.com/sixdegrees1.html` (Last Accessed: 07/04/2010).

G. Bellinger, D. Castro, and A. Mills. Data, Information, Knowledge, and Wisdom. Web Page, 2004. `http://www.systems-thinking.org/dikw/dikw.htm` (Last Accessed: 07/04/2010).

T. Berners-Lee. Information Management: A Proposal. Technical report, CERN, March 1989.

T. Berners-Lee. The Semantic Web. W3C Presentation, March 2002. `http://www.w3.org/2003/Talks/03-pcforum-tbl/` (Last Accessed: 07/04/2010).

A. Bernstein, E. Kaufmann, C. Kaiser, and C. Kiefer. Ginseng: A Guided Input Natural Language Search Engine for Querying Ontologies. In *Proceedings of the 2006 Jena User Conference*, May 2006.

D. Berrueta, D. Brickley, S. Decker, S. Fernández, C. Görn, A. Harth, T. Heath, K. Idehen, K. Kjernsmo, A. Miles, A. Passant, A. Polleres, L. Polo, and M. Sintek. SIOC Core Ontology Specification. W3C Member Submission, June 2007. `http://www.w3.org/Submission/2007/02/` (Last Accessed: 07/04/2010).

J. Bhagat, F. Tanoh, E. Nzuobontane, T. Laurent, J. Orlowski, M. Roos, K. Wolstencroft, S. Aleksejevs, R. Stevens, S. Pettifer, R. Lopez, and C. Goble. BioCatalogue: a universal catalogue of web services for the life sciences. *Nucleic Acids Research*, May 2010.

N. Bilton. Price of Facebook Privacy? Start Clicking. Electronic news article, May 2010. `http://www.nytimes.com/2010/05/13/technology/personaltech/13basics.html` (Last accessed: 31/08/2010).

C. Bizer, R. Cyganiak, and T. Heath. How to Publish Linked Data on the Web. Web Page, July 2007. http://www4.wiwiss.fu-berlin.de/bizer/pub/LinkedDataTutorial/20070727/ (Last Accessed: 07/04/2010).

C. Bizer, T. Heath, and T. Berners-Lee. Linked Data - The Story So Far. *International Journal on Semantic Web and Information Systems (IJSWIS)*, page preprint, 2009.

F. S. Black. *A Deductive Question-Answering System.* PhD thesis, Harvard University, Cambridge, Mass., June 1964.

S. Boag, D. Chamberlin, M. F. Fernández, J. Robie, and J. Siméon. XQuery 1.0: An XML Query Language. W3C Recommendation, W3C, January 2007.

T. L. Booth. Probabilistic Representation of Formal Languages. In *Tenth Annual IEEE Symposium on Switching and Automata Theory*, pages 74–81. IEEE, 1969.

T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, and F. Yergeau. *Extensible Markup Language (XML) 1.0.* W3C, third edition edition, February 2004. W3C Recommendation.

D. Brickley and R. V. Guha. RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation, February 2004. http://www.w3.org/TR/rdf-schema/ (Last Accessed: 07/04/2010).

D. Brickley and L. Miller. FOAF Vocabulary Specification 0.97. Web Page, January 2010. http://xmlns.com/foaf/spec/20100101.html (Last Accessed: 07/04/2010).

E. Brill, S. Dumais, and M. Banko. An Analysis of the AskMSR Question-Answering System. In *Conference on Empirical Method in Natural Language Processing*, 2002.

J. Broekstra, A. Kampman, and F. van Harmelen. Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. In *ISWC2002*, 2002.

S. Buckingham Shum, E. Motta, and J. Domingue. ScholOnto: An Ontology-Based Digital Library Server for Research Documents and Discourse. *International Journal on Digital Libraries*, 3:237–248, 2000.

G. Burns and T. Russ. Biomedical Knowledge Engineering tools based on Experimental Design: a case study based on neuroanatomical tract-tracing experiments. In *KCAP 2009*, Long Beach, CA., 2009.

G.A. Burns. Knowledge management of the neuroscientific literature: the data model and underlying strategy of the NeuroScholar system. *Philos Trans R Soc Lond B Biol Sci*, 356 (1412):1187–208, 2001. ISSN 0962-8436.

D. Buytaert. About the Drupal project. Web Page, February 2009. http://drupal.org/History-mission-and-community (Last Accessed: 07/04/2010).

P. Carlson. Apache Lucene - Query Parser Syntax. Online PDF document, 2006. http://lucene.apache.org/java/2_9_1/queryparsersyntax.pdf (Last Accessed: 07/04/2010).

J. Carroll, I. Dickinson, C. Dollin, D. Reynolds, A. Seaborne, and K. Wilkinson. Jena: Implementing the Semantic Web Recommendations. In *WWW2004*, 2004.

F. Cerbah. Learning Highly Structured Semantic Repositories from Relational Databases - The RDBtoOnto Tool. In *Proceedings of the 5th European Semantic Web Conference (ESWC 2008)*, June 2008.

N. Chomsky. Three models for the description of language. *IEEE Transactions on Information Theory*, 2(3):113–124, September 1956.

P. Ciccarese, T. Clark, and M. Ocana. Semantic Web Applications in Neuromedicine (SWAN) Ontology. W3C Interest Group Note, October 2009. `http://www.w3.org/TR/hcls-swan/` (Last Accessed: 09/04/2010).

P. Ciccarese, M. Ocana, S. Das, and T. Clark. AO: An Open Annotation Ontology for Science on the Web. In *Bio-ontologies 2010*, 2010.

E. Clark. Friends Reunited comes of age. Electronic news article, January 2003. `http://news.bbc.co.uk/1/hi/business/2652959.stm` (Last Accessed: 27/08/2010).

G. R. Cochrane and M. Y. Galperin. The 2010 Nucleic Acids Research Database Issue and online Database Collection: a community of data resources. *Nucleic acids research*, 38(Database issue):D1–4, January 2010. ISSN 1362-4962.

P. R. Cohen. The Role of Natural Language in a Multimodal Interface. In *Proceedings of the 5th annual ACM symposium on User interface software and technology*, pages 143–149, New York, NY, USA, 1992. ACM Press.

S. Coles, J. G. Frey, M. Hursthouse, M. Light, L. Carr, D. De Roure, C. Gutteridge, H. Mills, K. Meacham, M. Surridge, E. Lyon, R. Heery, M. Duke, and M. Day. The 'end to end' crystallographic experiment in an e-Science environment: From conception to publication. `http://eprints.soton.ac.uk/17454/` (Last Accessed: 21/11/2010), September 2005.

M. J. Collins. A new statistical parser based on bigram lexical dependencies. In Arivind Joshi and Martha Palmer, editors, *Proceedings of the Thirty-Fourth Annual Meeting of the Association for Computational Linguistics*, pages 184–191, San Fransisco, California, US, 1996. Morgan Kaufmann Publishers.

comScore. The 2009 U.S. Digital Year in Review: A Recap of the Year in Digital Marketing. Online PDF Document, February 2010. `http://www.comscore.com/layout/set/popup/request/Presentations/2010/The_2009_U.S._Digital_Year_in_Review_PDF_Request?req=slides&pre=The+2009+U.S.+Digital+Year+in+Review` (Last Accessed: 07/04/2010).

S. Corlosquet. Drupal RDF Schema proposal. Web Page, March 2008. `http://groups.drupal.org/node/9311` (Last Accessed: 07/04/2010).

M. Courtot, W. Bug, F. Gibson, A. L. Lister, J. Malone, D. Schober, R. Brinkman, and A. Ruttenberg. The OWL of Biomedical Investigations. In *OWLED 2008*, 2008.

Creative Commons. What is Creative Commons. Online PDF Document, February 2008. `http://wiki.creativecommons.org/images/3/35/Creativecommons-what-is-creative-commons_eng.pdf` (Last Accessed: 07/04/2010).

D. Cruickshank. myExperiment Wiki - Developer:API. Wiki Page, May 2009. `http://wiki.myexperiment.org/index.php?title=Developer:API` (Last Checked: 07/04/2010).

N. Cullot, R. Ghawi, and K. Yetongnon. DB2OWL: A Tool for Automatic Database-to-Ontology Mapping. In *Proceedings of the 15th Italian Symposium on Advanced Database Systems (SEBD 2007)*, pages 491–494, June 2008.

H. Cunningham. *Information Extraction - a User Guide (Second Edition)*. Institute for Language, Speech and Hearing (ILASH) and Department of Computer Science, University of Sheffield, Sheffield, UK, 2 edition, April 1999.

H. Cunningham, D. Maynard, K. Bontcheva, V. Tablan, C. Ursu, M. Dimitrov, M. Dowman, N. Aswani, I. Roberts, Y. Li, and A. Shafirin. *Developing Language Processing Components with GATE Version 4 (a User Guide)*, 2007.

DCMI Usage Board. DCMI Metadata Terms. Web Page, January 2008. `http://dublincore.org/documents/dcmi-terms/` (Last Accessed: 07/04/2010).

D. De Roure. myExperiment Wiki - Developer:Licensing. Wiki Page, July 2009. `http://wiki.myexperiment.org/index.php?title=Developer:Licensing` (Last Accessed: 07/04/2010).

D. De Roure. myExperiment Wiki - About. Wiki Page, July 2010a. `http://wiki.myexperiment.org/index.php?title=About` (Last Accessed: 09/09/2010).

D. De Roure. Replacing the Paper: The Twelve Rs of the e-Research Record. Web page, November 2010b. `http://blogs.nature.com/eresearch/2010/11/replacing_the_paper_the_twelve_rs_of_the_e-research_record.html` (Last accessed: 28/11/2010).

D. De Roure, M. Borkum, P. Groth, D. Michaelides, and J. Bhagat. myExperiment Wiki - Developer:Projects. WIki page, January 2010a. `http://wiki.myexperiment.org/index.php?title=Developer:Projects` (Last Accessed: 07/04/2010).

D. De Roure, C. Goble, J. Bhagat, D. Cruickshank, A. Goderis, D. Michaelides, and D. R. Newman. myExperiment: Defining the Social Virtual Research Environment. In *4th IEEE International Conference on e-Science*, pages 182–189. IEEE Press, December 2008.

D. De Roure, C. Goble, and R. Stevens. Designing the myExperiment Virtual Research Environment for the Social Sharing of Workflows. In *e-Science 2007: Proceedings of the Third IEEE International Conference on e-Science and Grid Computing*, pages 603–610. IEEE Computer Society, 2007.

D. De Roure, C. Goble, and R. Stevens. The Design and Realisation of the myExperiment Virtual Research Environment for Social Sharing of Workflows. *Future Generation Computer Systems*, 25:561–567, February 2009.

D. De Roure, N. R. Jennings, and N. R. Shadbolt. Research Agenda for the Semantic Grid: A Future e-Science Infrastructure, December 2001.

David De Roure, Carole Goble, Sergejs Aleksejevs, Sean Bechhofer, Jiten Bhagat, Don Cruickshank, Paul Fisher, Nandkumar Kollara, Danius Michaelides, Paolo Missier, David Newman, Marcus Ramsden, Marco Roos, Katy Wolstencroft, Ed Zaluska, and Jun Zhao. The Evolution of myExperiment. In *Sixth IEEE e-Science conference (e-Science 2010)*, October 2010b.

A. de Waard and G. Tel. The ABCDE format - enabling semantic conference proceeding. In *1st Workshop: SemWiki2006 - From Wiki to Semantics*, Budva, Montenegro, 2006.

E. Deelman, D. Gannon, M. Shields, and I. Taylor. Workflows and e-Science: An overview of workflow system features and capabilities. *Future Generation Computer Systems*, 25(5): 528–540, 2009. ISSN 0167-739X.

A. Di Iorio, F. Vitali, and S. Peroni. The Pattern Ontology: Describing documents by means of their structural components. Web page, October 2010. `http://www.essepuntato.it/2008/12/pattern` (Last Accessed: 02/12/2010).

A. Dix, M. Blythe, K. Overbeeke, A. Monk, and P. Wright. *Funology: From Usability to Enjoyment*, chapter 13: Deconstructing Experience - pulling crackers apart, pages 165–178. Kluwer Academic Publishers, Dordrecht, 2003.

M. Duke. eBank UK Homepage. Web page, October 2009. `http://www.ukoln.ac.uk/projects/ebank-uk/` Last Accessed: 27/08/2010).

E. Dumbill. XML Watch: Finding friends with XML and RDF. Web page, June 2002. `http://www.ibm.com/developerworks/xml/library/x-foaf.html` (Last Accessed: 07/04/2010).

D. Dupplaw, D. Brunson, A.E. Vine, C. P. Please, S. M. Lewis, A.M. Dean, A. J. Keane, and M. J. Tindall. Web-based knowledge elicitation and application to planned experiments for product development. In *ASME Design Engineering Technology Conference*. ASME International, 2003. DETC2003/CIE-48275.

O. Erling. Implementing a SPARQL compliant RDF Triple Store using a SQL-ORDBMS. Technical report, OpenLink Software, 2009.

C. Fellbaum, G. A. Miller, R. Al-Halimi, R. C. Berwick, J. F. M. Burg, M. Chodorow, J. Grabowski, and S. Harabagiu. *WordNet: An Electronic Lexical Database*. MIT Press, 1998.

D. Field, S. Sansone, A. Collis, T. Booth, P. Dukes, S. K. Gregurick, K. Kennedy, P. Kolar, E. Kolker, M. Maxon, S. Millard, A. Mugabushaka, N. Perrin, J. E. Remacle, K. Remington, P. Rocca-Serra, C. F. Taylor, M. Thorley, B. Tiwari, and J. Wilbanks. 'Omics Data Sharing. *Science*, 326(5950):234–236, October 2009. ISSN 1095-9203.

D. Florescu, A. Y. Levy, and A. O. Mendelzon. Database Techniques for the World-Wide Web: A Survey. *SIGMOD Record*, 27(3):59–74, 1998.

FOAF project team. DataSources - FOAF Wiki. Wiki page, December 2009. `http://wiki.foaf-project.org/w/DataSources` (Last Accessed: 07/04/2010).

Franz Inc. AllegroGraph Introduction. Web page, March 2010. `http://www.franz.com/agraph/support/documentation/current/agraph-introduction.html` (Last Checked: 07/04/2010).

J. G. Frey, R. Gledhill, S. Kent, B. Hudson, and J. Essex. Schools Malaria Project. CombeChem Project Output, September 2006.

G. Gazdar, E. Klein, G. K. Pullum, and I. A. Sag. *Generalized Phrase Structure Grammar*. Basil Blackwell Publisher Ltd., 108 Cowley Road, Oxford, UK, 1 edition, 1985.

GeoCities. 29 Neighborhoods. Web Page, 1995. `http://web.archive.org/web/19961219233921/www.geocities.com/homestead/homedir.html` (Last Checked: 07/04/2010).

T. Geoghegan. What's the ideal number of friends? Electronic news article, March 2009. `http://news.bbc.co.uk/1/hi/magazine/7920434.stm` (Last accessed: 27/08/2010).

F. Giasson and B. D'Arcus. Bibliographic Ontology Specification. Web Page, November 2009. `http://bibliontology.com/specification` (Last Checked: 09/04/2010).

A. Gibson, R. Stevens, R. Cooke, S. Brostoff, and m. c. schraefel schraefel. myTea: Connecting the Web to Digital Science on the Desktop. Technical report, ECS, University of Southampton, October 2005.

H. Glaser, A. Jaffri, and I. Millard. Managing Co-reference on the Semantic Web. In *WWW2009 Workshop: Linked Data on the Web (LDOW2009)*, April 2009.

C. Goble and D. De Roure. myExperiment: social networking for workflow-using e-scientists. In *Proceedings of the 2nd workshop on Workflows in support of large-scale science*, pages 1–2. ACM, June 2007.

C. Goble, R. Stevens, G. Ng, S. Bechhofer, N. W. Paton, P. Baker, M. Peim, and A. Brass. Transparent access to multiple bioinformatics information sources. *IBM Systems Journal*, 40 (2), 2001.

C. Goble, C. Wroe, and R. Stevens. The myGrid project: services, architecture and demonstrator. In *All Hands Meeting*, pages 595–603, September 2003.

B. F. Green Jr., A. K. Wolf, C. S. Chomsky, and K. Laughery. Baseball: An Automatic Question-Answerer. *Computers and Thought*, pages 207–216, 1963.

R. Grishman. TIPSTER Architecture Design Document Version 2.3. Technical report, DARPA, 1997.

T. Groza, S. Handschuh, T. Clark, S. Buckingham Shum, and A. de Waard. A Short Survey of Discourse Representation Models. In *Semantic Web Applications in Scientific Discourse (SWASD) ISWC2009 Workshop*, 2009.

T. Groza, S. Handschuh, K. Möller, and S. Decker. SALT - Semantically Annotated $LaTeX$ for Scientific Publications. In *ESWC '07: Proceedings of the 4th European conference on The Semantic Web*, pages 518–532, PBerlin, Heidelberg, 2007.

R. V. Guha and T. Bray. Meta Content Framework Using XML. W3C Note, June 1997. `http://www.w3.org/TR/NOTE-MCF-XML/` (Last Checked: 07/04/2010).

S. Harabagiu, D. Moldovan, M. Pasca, R. Mihalcea, M. Surdeanu, R. Bunescu, R. Gîrju, V. Rus, and P. Morarescu. FALCON: Boosting Knowledge for Answer Engines. In *In the Proceedings of Text REtrieval Conference (TREC-9), 2000*, 2000.

F. A. P. Harmsze. *A modular structure for scientific articles in an electronic environment.* PhD thesis, University of Amsterdam, 2000.

S. Harris, N. Lamb, and N. R. Shadbolt. 4store: The Design and Implementation of a Clustered RDF Store. Technical report, Garlik Ltd., 2009.

HCLS Scientific Discourse Sub Task Group. HCLSIG/SWANSIOC/Actions/RhetoricalStructure/alignment/mediumgrain. Wiki page, December 2010. `http://esw.w3.org/HCLSIG/SWANSIOC/Actions/RhetoricalStructure/alignment/mediumgrain` (Last Accessed: 07/12/2010).

M. R. Hejazi, M. S. Mirian, K. Neshatian, A. Jalali, and B. Ofoghi. TelQAS: A Telecommunication Literature Question Answering System Benefits from a Text Categorization Mechanism. In *IKE*, pages 500–504, 2003.

G. G. Hendrix, E. D. Sacerdoti, D. Sagalowicz, and J. Slocum. Developing a Natural Language Interface to Complex Data. *ACM Transactions on Database Systems*, 3(2):105–147, June 1978.

A. J. G. Hey and A. E. Trefethen. The UK e-Science Core Programme and the Grid. *Future Generation Computer Systems*, 18(8):1017–1031, September 2002.

E. W. Hinrichs. Tense, Quantifiers, and Contexts. *Computational Linguistics*, 14(2), June 1988.

I. Horrocks and P. F. Patel-Sch nei der. Reducing OWL Entailment to Description Logic Satisfiability. In *Proc. of the 2nd International Semantic Web Conference (ISWC)*, 2003.

I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean. SWRL: A Semantic Web Rule Language Combining OWL and RuleML. W3C Member Submission, May 2004. `http://www.w3.org/Submission/SWRL/` (Last Checked: 07/04/2010).

I. Horrocks, P. F. Patel-Schneider, and F. van Harmelen. Reviewing the design of DAML+OIL: an ontology language for the semantic web. In *Eighteenth national conference on Artificial intelligence*, pages 792–797, Menlo Park, CA, USA, 2002. American Association for Artificial Intelligence. ISBN 0-262-51129-0.

G. Hughes, H. Mills, D. De Roure, J. G. Frey, L. Moreau, m. c. schraefel schraefel, G. Smith, and E. Zaluska. The Semantic Smart Laboratory: A system for supporting the chemical eScientist. *Org. Biomol. Chem.*, 2:1–10, 2004. DOI: 10.1039/b410075a.

IDEAlliance. PRISM: Publishing Requirements for Industry Standard Metadata (Version 2.1), 2009.

International Federation of Library Associations and Institutions. Functional Requirements for Bibliographic Records: Final Report, 1998.

A. Isaac and E. Summers. SKOS Simple Knowledge Organization System Primer. Web Page, August 2009. `http://www.w3.org/TR/skos-primer` (Last Checked: 07/04/2010).

A. W. Johnson and T. K. Earle. *The evolution of human societies: from foraging group to agrarian state.* Stanford University Press, Stanford, Calif., 1987. ISBN 0804713391 0804715157.

A. K. Joshi, Y. Schabes, G. Rozenberg, and A. Salomaa. *Handbook of Formal Languages*, chapter Tree-Adjoining Grammars, pages 69–124. Springer, Berlin, New York, 1997.

J. Kahan, M. R. Koivunen, E. Prud'Hommeaux, and R. R. Swick. Annotea: an open RDF infrastructure for shared Web annotations. *Computer Networks*, 39(5):589 – 608, 2002. ISSN 1389-1286.

R. M. Kaplan and J. Bresnan. *The Mental Representation of Grammatical Relations*, chapter Lexical-Functional Grammar: A Formal System for Grammatical Representation, pages 173–281. MIT Press, 1982.

G. Karvounarakis, S. Alexaki, V. Christophides, D. Plexousakis, and M Scholl. RQL: A Declarative Query Language for RDf. In *WWW2002*, May 2002.

B. Katz, S. Felshin, D. Yuret, A. Ibrahim, J. Lin, G. Marton, A. J. McFarland, and B. Temelkuran. Omnibase: Uniform Access to Heterogeneous Data for Question Answering. In *The 7th International Workshop on Applications of Natural Language to Information Systems (NLDB 2002)*, 2002.

J. R. Kelly, B. Canton, and R. P. Shetty. OpenWetWare:Mission. Wiki Page, April 2008. `http://openwetware.org/wiki/OpenWetWare:Mission` (Last Checked: 07/04/2010).

D. Klein and C. D. Manning. Fast Exact Inference with a Factored Model for Natural Language Parsing. In *Advances in Neural Information Processing Systems15*, pages 3–10. MIT Press, 2003.

C. A. Knoblock, S. Minton, J. L. Ambite, N. Ashish, I. Muslea, A. G. Philpot, and S. Tejada. The Ariadne approach to Web-based information integration. *IEEE Intelligent Systems*, 10 (1-2):145–169, 2001.

J. Krause, D. Lusseau, and R. James. Animal social networks: an introduction. *Behavioral Ecology and Sociobiology*, 63(7):967–973, May 2009.

C. Lagoze. The oreChem Project: Integrating Chemistry Scholarship with the Semantic Web. In *WebSci'09: Society On-Line*, 2009.

C. Lagoze, H. Van de Sompel, P. Johnston, M. Nelson, R. Sanderson, and S. Warner. ORE User Guide - Primer. Web Page, October 2008. `http://www.openarchives.org/ore/1.0/primer.html` (Last Accessed: 07/04/2010).

C. Lagoze, H. Van de Sompel, M. Nelson, and S. Warner. The Open Archives Initiative Protocol for Metadata Harvesting (Version 2.0). Web page specification, June 2002. `http://www.openarchives.org/OAI/openarchivesprotocol.html` (Last Accessed: 13/09/2010).

O. Lassila. Web Metadata: A Matter of Semantics. *IEEE Internet Computing*, 2(4):30–37, July/August 1998.

O. Lassila and R. R. Swick. Resource Description Framework (RDF) Model and Syntax Specification. Technical report, W3C, February 1999. W3C Recommendation.

R. Leigh. Videos: Police step in to prevent Facebook flash mob events. Electronic news article, May 2008. http://www.mirror.co.uk/news/weird-world/2008/05/19/videos-police-step-in-to-prevent-facebook-flash-mob-events-115875-20423056/ (Last accessed: 27/08/2010).

Y. Li, H. Yang, and H. V. Jagadish. NaLIX: an Interactive Natural Language Interface for Querying XML. In *Proceedings of SIGMOD 2005*, 2005.

D. Lin. Dependency-based Evaluation of MINIPAR. In *Workshop on the Evaluation of Parsing Systems*, May 1998.

F. Lisacek, C. Chichestera, A. Kaplan, and A. Sandor. Discovering Paradigm Shift Patterns in Biomedical Abstracts: Application to Neurodegenerative Diseases. In *1st International Symposium on Semantic Mining in Biomedicine (SMBM)*, 2005.

m. c. schraefel, G. Hughes, H. Mills, G. Smith, and J. G. Frey. Making Tea: Iterative Design through Analogy. In Proceedings of Designing Interactive Systems. In *2004 conference on Designing interactive systems: processes, practices, methods, and techniques*, pages 49–58, Cambridge Mass, USA, 2004a. ACM Press.

m. c. schraefel, G. Hughes, H. Mills, G. Smith, T. Payne, and J. G. Frey. Breaking the Book: Translating the Chemistry Lab Book into a Pervasive Computing Lab Environment. In *Conference on Human Factors in Computing Systems (CHI)*, pages 25–32, Vienna, Austria, 2004b. ACM Press.

m. c. schraefel, D. A. Smith, A. Russel, A. Owens, C. Harris, and M. L. Wilson. The mSpace Clasical Music Explorer: Improving Access to the Classical Music Space for Real People. In *MusicNetwork Open Workshop: Integration of Music in Multimedia Applications*, Vienna, Austria, 2005.

D. M. Magerman. Parsing as Statistical Pattern Recognition. Technical report, IBM T. J. Watson Research Center, January 1995.

J. Malone, T. F. Rayner, X. Z. Bradley, and H. Parkinson. Developing an application focused experimental factor ontology: embracing the OBO Community. In *ISMB2008*, 2008.

F. Manola, E. Miller, and B. McBride. RDF Primer. W3C Recommendation, February 2004. `http://www.w3.org/TR/rdf-primer/` (Last Checked: 07/04/2010).

M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):315–330, 1994.

V. M. Markowitz. Heterogeneous Molecular Biology Databases. *Computational Biology*, 2(2): 537–538, 1995.

D. Martin, A. Ankolekar, M. Burstein, G. Denker, D. Elenius, J. Hobbs, L Kagal, O. Lassila, A. McDermott, D. McGuinness, S. McIlraith, M. Paolucci, B. Parsia, T. Payne, T. Sabou, C. Schlenoff, E. Sirin, M. Solanki, N. Srinivasan, K. Sycara, and R. Washington. OWL-S: Semantic Markup for Web Services. DAML Technical Overview Web Page, November 2004. `http://www.daml.org/services/owl-s/1.1/overview/` (Last Checked: 07/04/2010).

P. Mika. Ontologies are us: A unified model of social networks and semantics. In *ISWC 2005*, 2005.

M. Minsky. A Framework for Representing Knowledge. Technical report, Massachusetts Institute of Technology, Cambridge, MA, USA, 1974.

Y. Mizuka, A. Korhonen, T. Mullen, and N. Collier. Zone analysis in biology articles as a basis for information extraction. *International Journal of Medical Informatics*, 75:468–487, 2006.

B. Mons and J. Velterop. Nano-Publication in the e-science era. In *Semantic Web Applications in Scientific Discourse*, 2009.

D. Naber. OpenThesaurus: Building a Thesaurus with a Web Community. Online PDF Document, June 2004. `http://www.openthesaurus.de/download/openthesaurus.pdf` (Last Checked: 07/04/2010).

D. R. Newman, S. Bechhofer, and D. De Roure. myExperiment: An ontology for e-Research. In *Semantic Web Applications in Scientific Discourse*, October 2009.

M. Nottingham and R. Sayre. The Atom Syndication Format. Request For Comments, December 2005. `http://tools.ietf.org/html/rfc4287` (Last Checked 07/04/2010).

P. J. Nürnberg, J. J. Leggett, and E. R. Schneider. As We Should Have Thought. In *Eighth ACM conference on Hypertext*, 1997.

T. Oinn, M. Greenwood, M. Addis, M. N. Alpdemir, J. Ferris, K. Glover, C. Goble, A. Goderis, D. Hull, D. Marvin, P. Li, P. Lord, M. R. Pocock, M. Senger, R. Stevens, A. Wipat, and Chris Wroe. Taverna: lessons in creating a workflow environment for the life sciences. *Concurrency Computat.: Pract. Exper*, 18:1067–1100, 2006.

A. Passant, P. Ciccarese, J. G. Breslin, and T. Clark. SWAN/SIOC: Alignment Between the SWAN and SIOC Ontologies. W3C Interest Group Note, October 2009. `http://www.w3.org/TR/hcls-swansioc/` (Last Accessed: 09/04/2010).

A. V. Phillips. A Question-Answering Routine. Memo 16, MIT, Cambridge, Mass., May 1960.

S. Phillips. A brief history of Facebook. Electronic News Article, July 2007. `http://www.guardian.co.uk/technology/2007/jul/25/media.newmedia` (Last Checked 07/04/2010).

A. Powell, M. Nillson, A. Naeve, P. Johnston, and T. Baker. DCMI Abstract Model. Web Page, June 2007. `http://dublincore.org/documents/abstract-model/` (Last Checked: 07/04/2010).

E. Prud'hommeaux and A Seaborne. SPARQL Query Language for RDF. W3C Candidate Recommendation, April 2006. `http://www.w3.org/TR/rdf-sparql-query/` (Last Checked: 07/04/2010).

D. Quick. Diaspora – a distributed, open source, secure social network with Facebook in its sights. Electronic News Article, May 2010. `http://www.gizmag.com/diaspora-open-source-social-network/15098/` (Last accessed 31/08/2010).

A. Rodriguez. RESTful Web services: The basics. Online PDF document, November 2008. `http://download.boulder.ibm.com/ibmdl/pub/software/dw/webservices/ws-restful/ws-restful-pdf.pdf` (Last Checked: 07/04/2010).

G. R. Sampson. *English for the Computer*. Oxford University Press, 1995.

M. Samwald and H. Stenzhorn. Simple, ontology-based representation of biomedical statements through fine-granular entity tagging and new web standards. In *Bio-Ontologies 2009*, June 2009.

R. Sanderson and H. Van de Sompel. Open Annotation: Alpha3 Data Model Guide. Web page, October 2010. http://www.openannotation.org/spec/alpha3/ (Last accessed: 12/12/2010).

F. Schlesinger. Swivelchair activism. Electronic news article, December 2007. http://www.guardian.co.uk/education/2007/dec/11/students.studentpolitics (Last accessed: 27/08/2010).

A Seaborne. RDQL - A Query Language for RDF. W3C Member Submission, January 2004. http://www.w3.org/Submission/RDQL/ (Last Checked: 07/04/2010).

B. Shneiderman. Natural vs. precise language for human operation of computers. In *ACL Proceedings, 18th Annual Meeting*, pages 139–141, 1980.

D. Shotton. CiTO, the Citation Typing Ontology, and its use for annotation of reference lists and visualization of citation networks. In *Bio-Ontologies 2009, a Special Interest Group meeting at ISMB 2009*, 2009. Image Bioinformatics Research Group, Department of Zoology, University of Oxford, Oxford OX1 3PS, UK.

D. Shotton and S. Peroni. The Document Components Ontology (DoCO). Web page, November 2010. http://sempublishing.svn.sourceforge.net/viewvc/sempublishing/DoCO/doc/doco.html (Last Accessed: 07/12/2010).

B. Smith. The Basic Tools of Formal Ontology. In *Formal Ontology in Information Systems*, 1998.

S. L. Star and J. R. Griesemer. Institutional Ecology, 'Translations' and Boundary Objects: Amateurs and Professionals in Berkeley's Museum of Vertebrate Zoology, 1907-39. *Social Studies of Science*, 19(3):387–420, 1989.

R. Stevens, C. Goble, P. Baker, and A. Brass. A classification of tasks in bioinformatics. *Bioinformatics*, 17(2):180–188, 2001.

R. Stevens, C. Goble, N. W. Paton, S. Bechhofer, G. Ng, P. Baker, and A. Brass. Complex Query Formulation Over Diverse Information Sources in TAMBIS. In Zoe Lacroix and Terence Critchlow, editors, *Bioinformatics: Managing Scientific Data*. Morgan Kaufmann, May 2003. ISBN 1-55860-829-X.

P. J. Stone, R. F. Bayles, J. Z. Namerwith, and D. M. Ogilvie. The General Inquirer: A Computer System for Content Analysis and Retrieval Based on the Sentence as a Unit of Information. *Behavioural Science*, 7(4):1–15, 1962.

M. Taboada and W. C. Mann. Rhetorical Structure Theory: looking back and moving ahead. *Discourse Studies*, 8:423–459, 2006.

J. Tennison. Linked Open Data in a Changing World. Web Page Blog, July 2009. http://www.jenitennison.com/blog/node/108 (Last Accessed: 27/04/2010).

S. Teufel and M. Moens. Summarizing Scientific Articles: Experiments with Relevance and Rhetorical Status. *Computational Linguistics*, 28, 2002.

D. Thakker, T. Osman, and P. Lakin. *GATE JAPE Grammar Tutorial Version 1.0*, February 2009.

D. Thomas, D. H. Hansson, L. Breedt, M. Clark, J. D. Davidson, J. Gehtland, and A. Schwarz. *Agile Web Development with Rails*. Pragmatic Bookshelf, 2nd edition, 2007.

S. E. Toulmin. *The Uses of Argument*. Cambridge University Press, 1969.

A. M. Turing. Computing machinery and intelligence. *Mind*, 59:433–460, 1950.

J. Weizenbaum. ELIZA - A Computer Program For the Study of Natural Language Communication Between Man And Machine. *American Journal of Computational Linguistics*, 9(1): 36–45, January 1966.

D. Wood, P. Gearon, and T. Adams. Kowari: A Platform for Semantic Web Storage and Analysis. In *WWW2005*, 2005.

W. A. Woods. Transition network grammars for natural language analysis. *Communications of the ACM*, 13(10):591–606, October 1970.

W. A. Woods, R. M. Kaplan, and B. N. Webber. *The Lunar Sciences Natural Language Information System: Final Report*. Woods, W. A. and Kaplan, R. M. and Webber, B. N., Cambridge, Massachusetts, 1972.

J. Yousefi and L. Kosseim. Automatic Acquisition of Semantic-Based Question Reformulations for Question Answering. In A. Gelbukh, editor, *CICLing 2006*, pages 441–452. Springer-Verlag Berlin Heidelberg, 2006.

J. Zhao, K. Alexander, M. Hausenblas, and R. Cyganiak. VoID Guide - Using the Vocabulary of Interlinked Datasets. Web Page, December 2008. `http://rdfs.org/ns/void-guide` (Last Checked: 07/04/2010).