

Description of Research

1. Introduction

1.1 Background

Metadata and ontologies are widely identified as key technologies for E-Science. Ontologies are being developed and applied in domains as diverse as molecular biology (GO¹/GONG²) astronomy (AstroGrid³) and aircraft engineering (Geodise⁴). Ontologies and metadata are the foundation of the Semantic Web⁵/Grid⁶ and intelligent web services. They provide the frameworks for describing the meaning of resources and services in terms that software agents and other services can understand and manipulate.

Expertise in ontology development is one of the UK and Manchester's major strengths. Manchester's skills range over:

- a) Languages for ontology representation – it is a prime mover in the new Web Ontology Language being developed by W3C, OWL⁷ (previously known as DAML+OIL), and was the developer of the first practical reasoner for the expressive description logics which underpin OWL and related ontology languages;
- b) Tools for ontology development, ranging from the popular OilEd⁸ (Bechhofer, Horrocks et al. 2001), a notepad for simple ontology sketching, to full blown ontology development environments in the GALEN projects (Rector, Zanstra et al. 1999; Rector, Wroe et al. 2001) and the WonderWeb⁹ project;
- c) Experience in developing real, large scale ontologies in collaborative environments and challenging scientific domains – GONG (Wroe, Stevens et al. 2003), the UK Drug Ontology (Solomon, Wroe et al. 1999), and the *OpenGALEN* Common Reference Model (Rogers and Rector 1996).

This experience has led to the observation that current ontology development practice is at a similar stage to software development two decades ago. It presumes that each ontology is started from scratch, and it approaches ontology development more as a craft than as a principled engineering discipline. Examples are often small scale 'toys' for well understood domains. This has lulled the ontology community into a false confidence and led to knowledge engineers building ontologies on behalf of domain experts rather than enabling domain experts to develop their own ontologies for themselves. Tools such as OilEd have thus been developed primarily for ontology experts rather than domain experts.

Realistically, for ontologies to be developed and widely accepted in established domains as proposed for Grid – *e.g.* molecular biology, astronomy, fluid dynamics, medicine, etc. – the ontologies must be developed and 'owned' by the domain experts themselves. Our experience has shown that, given appropriate tools and guidance, interested domain experts can build ontologies for their own disciplines more readily than can ontology experts from outside those disciplines. However, to do so they require tools oriented to their needs rather than to those of ontology experts.

The key to success for tools for domain experts is that they allow the domain expert to concentrate on domain issues. Domain experts should not have to become logicians to use logic based tools.

Hence we regard the underlying logic based language – be it OWL, SHIQ/FaCT, GRAIL, or some other – as an "assembly code" to be seen only by ontology experts. Domain experts interact with "high level languages" or "ontology views". In line with common practice in the knowledge acquisition community, we refer to the views used for creating or populating ontologies as "Intermediate Representations" (Gaines and Boose 1988), while those designed to visualise and use the ontology in applications we refer to simply as "views" by analogy to the usage of the database community. Both require transformation mechanisms sensitive to the ontological classification. Such transformations constitute one of the two key research issues for this proposal.

The power of logic based ontologies is the ability of their reasoners to infer classifications (subsumption) and recognise inconsistencies. However, even ontology experts can find it difficult to work out why inferences were made or not made – *i.e.* to "debug" the ontology. Furthermore, the more powerful the reasoner, the more likely it is to make non-obvious inferences. Most modern reasoners which underpin OWL implementations, *e.g.* FaCT (Horrocks 1998), use algorithms based on tableaux calculus. Unfortunately, such reasoners transform problems internally to such an extent that explaining the resulting inferences directly is, to date, an unsolved problem.

However, experience suggests a range of heuristic techniques which help in most cases. Hence the second focus of the proposal is to develop debugging heuristics for ontologies expressed in logic-based languages such as OWL. The results of these heuristics must then be linked back through the view mechanism so as to be easily understood by domain expert users. If the project succeeds, it should be rare that a domain expert stares at the screen in bemused puzzlement over how the reasoner has re-arranged the ontology or why large numbers of the concepts in the ontology have suddenly been inferred to be inconsistent.

¹ <http://gong.man.ac.uk/>

² <http://www.geneontology.org/>

³ <http://www.astrogrid.org/>

⁴ <http://www.geodise.org/>

⁵ <http://www.semanticweb.org>

⁶ <http://www.semanticgrid.org>

⁷ <http://www.w3.org/2001/sw/WebOnt/>

⁸ <http://oiled.man.ac.uk>

⁹ <http://wonderweb.semanticweb.org/>

These two issues require fundamental research. The companion proposal submitted to JISC, CO-ODE, will develop the flexible plug and play framework within which to implement the results of this research. The two projects seek to take advantage of a rare opportunity for synchronised funding on the two sides of the Atlantic. The results will be integrated into a combined open user oriented ontology development environment based on the integration of the UK developed OilEd and the Stanford developed Protégé¹⁰, the most widely used frame-oriented knowledge acquisition environment.

The project will extend over three years with intermediate deliverables and will result in:

- a) Principled mechanisms for ontology views, intermediate representations, and associated transformations integrated into an environment for large scale ontology engineering;
- b) A set of debugging/explanation tools based on heuristics integrated into the same environment;
- c) A demonstration of that environment in the migration of the Gene Ontology to OWL in house and in ontologies for Cancer Therapeutics and Comparative Vertebrate Anatomy with collaborating projects.

The Gene Ontology (GO) was originally developed by hand with minimal support. The GONG¹¹ project is reformulating it in OWL with support from the description logic reasoner (Wroe, Stevens et al. 2003). The Gene Ontology is large scale, heavily used world wide in its existing form, and increasingly a keystone in the global genomics movement with users throughout the world. The reformulation is already underway, successful, but not sustainable and difficult to disseminate without more domain user oriented tools. It therefore presents an ideal test case for proposed methods.

1.2 Context, users, and use cases

This project forms part of an extended programme of work on ontologies and ontology development environments spanning the range from fundamental research on description logic classifiers (CAMELOT¹²), to middle ware oriented projects (^{my}Grid, WonderWeb), to user oriented tools (GOAT¹³) to the development and migration of large ontologies such as the Gene Ontology and *OpenGALen* Common Reference Model¹⁴. to practical applications of ontologies (PEN&PAD (Nowlan 1994; Kirby and Rector 1996), Tambis¹⁵ and COHSE¹⁶). The proposed project thus reflects many years' experience in dealing with users and developers of ontologies in a variety of environments.

From this experience we have identified four broad classes of users:

- a) Users needing to build small, specialist ontologies for a specific use. The goal for these users is to make simple things simple and help them avoid obvious pitfalls.
- b) Users needing to extend or adapt existing ontologies for a specific application. Often the existing ontologies will be very large and generic, but the users need only a small part and to make minimum extensions. We expect this class of users to increase rapidly. Hopefully it will come to be much more common than de novo development. The goal is to make it easier to adapt an existing, sharable, ontology than to build one from scratch.
- c) Developers of large scale shared reference ontologies such as the Gene Ontology, ontologies of Anatomy, Diseases, Drugs, Stars, Plants, etc. For this type of user the issue is how to improve productivity and accuracy. Developing ontologies is notoriously time consuming and expensive
- d) Applications developers who wish to use ontologies.

Across the different types of users we have identified seven key tasks.

- 1) Getting started
- 2) Developing the ontology using familiar intuitive notions and constructs including contingent information
- 3) Searching and navigating the ontology
- 4) Understanding and debugging the ontology
- 5) Extending, evolving, and maintaining the ontology.
- 6) Access to external resources indexed by or linked to the ontology
- 7) Using the ontology to build applications

This project is aimed primarily at large scale developers (c) and tasks 2) and 4). The companion proposal to the JCSR addresses the more straightforward developments required for smaller scale users (a-b) and applications builders (d) with respect to 1), 3), 6), 7) and the basic architecture to support 2).

In addition to the Gene Ontology/GONG projects, the proposed research will take into account requirements and experience from ongoing and related projects – the development of a cancer therapeutics ontology (CLEF), radiological images in MIAS-GRID and to the broad range of ontologies being developed in the E-Science frameworks both through ^{my}Grid and more widely. New ontology developments, e.g. on comparative mammalian anatomy, appear likely to emerge from independent initiatives and provide further opportunities for testing.

¹⁰ <http://protégé.stanford.edu>

¹¹ <http://www.gong.man.ac.uk>

¹² <http://www.cs.man.ac.uk/~horrocks/Camelot/> (EPSRC GR/L54516, successor in preparation)

¹³ <http://goat.man.ac.uk/> (Gene Ontology Annotation Tool)

¹⁴ <http://www.opengalen.org>

¹⁵ <http://imgproj.cs.man.ac.uk/tambis/>

¹⁶ <http://cohse.semanticweb.org/>

2. Proposed Approach

Paradoxically, the advent of more expressive ontology languages and re-usable ontology libraries makes ontologies less rather than more intuitive to understand and more rather than less difficult to debug. In simpler systems, choices are constrained and inferences limited and local. By contrast, when using an expressive ontology formalism such as OWL to develop a re-usable ontology:

- *Inferences can be complex and non-local.* Axioms can have wide-ranging effects which are hard to predict. Debugging is therefore difficult. Even experienced users are often perplexed by results.
- *There are usually several ways to express the same notion.* Expressions which lead to efficient computations are frequently unintuitive.
- *Careful choice of style is required to allow the ontology to be modularised* – otherwise it becomes analogous to an early ‘spaghetti’ programs in which ‘goto’ statements were used indiscriminately.
- *Generic reusable constructs tend to be more complex than application specific constructs.* Application specific ontologies can leave implicit information which generic ontologies must make explicit. However, most applications and users wish to leave implicit those aspects of the ontology which they regards as ‘obvious’. Hence application specific ‘views’ are required to reconcile the conflict.
- *No mechanism exists for the tableaux reasoners which underpin modern ontologies to ‘explain’ or ‘justify’ their conclusions.*

This proposal will develop two related mechanisms – heuristic debugging and ontology views – to provide a major step towards combining the ease of use of simpler representations with the power of expressive ontology formalisms.

2.1 Debugging and explanation tools

The proposed approach to debugging is heuristic, although it will draw on whatever improvements in tracking and explaining the underlying tableaux algorithms arise in parallel research in the course of the project (Borgida, Franconi et al. 2000). The heuristic approach is based extensive experience in building and debugging ontologies in OWL and in training others to do so. It is not our purpose to address the problem of providing explanations of larger problem solving tasks as in (Shadbolt and O'Hara 1997) or (Cawsey 1992). Rather, we propose to provide tools to make it possible to isolate the reason for specific inferences – to find errors in the ontology quickly.

Our starting point is a set of assumptions about ontology development each of which leads to a set of heuristics for debugging. From this base we will explore further methods and mechanisms.

- Users should express themselves in ways that reflect their intention.* For example, it is often possible to get logically equivalent results from domain and range constraints, from axioms, or from definitions. However, if expressed as a domain/range constraint, the user’s intention is probably that any violation is a simple error. Hence, domain and range constraints should be checked early in debugging or prechecked before classification.
- Users can be constrained to express most facts in ways that are relatively local.* For example, in expressive formalisms, axioms can be arbitrary. However the environment can constrain (or at least encourage) axioms to be ‘object oriented’ – *i.e.* to limit antecedents to single concepts and to limit consequents to characteristics (‘restrictions’) of those concepts. The effects of such ‘object oriented axioms’ are easy to trace. Similarly users can be constrained to maintain primitives as disjoint trees which can then be separated into modules. Little generality is lost by either restriction, and users find them intuitively reasonable (Rector 2002).
- Changes in inferences must arise from changes in the knowledge base;* hence keeping track of changes focuses the search for the cause of errors. The theoretical point is obvious; the practical implementation of its ramifications is non-trivial. It requires that the changes made by the user and the inferences made by the reasoner be tracked separately and in detail, that it be possible to compare the previous and current state meaningfully, and that it be easy to check-point a state and restore it when things go wrong subsequently.
- Most errors are simple and local;* therefore surface examination of the definitions and restrictions will usually show up the error quickly. Where the problem is not obvious, we will experiment with using a simple graph-based classification engine to retest the satisfiability or subsumption in question. Although logically incomplete, such classifiers catch most simple errors and their behaviour is relatively easy to explain (McGuinness and Borgida 1995).
- Most errors are single, even if they result in many concepts becoming unsatisfiable.* Any concept which involves an existential restriction whose value is inconsistent will itself be inconsistent. Therefore, a single error in a densely interconnected ontology can make large swathes of the ontology unsatisfiable. Tracing the network to find the common source unsatisfiable concept is an obvious task for debugging heuristics.
- Where none of the above succeed on their own, *the tableaux reasoner can be used as an ‘oracle’* and definitions or axioms simplified or augmented and then submitted to the ‘oracle’ until the problem has been found. Although obviously combinatorially explosive in the worst case, in most ontologies the number of possibilities is small enough to make such procedures manageable if guided heuristically.

2.2 Views, Intermediate Representations and Transformations

Our overall approach is that representations such as OWL should be viewed analogously to “assembly languages”. Domain experts should deal with “views” analogous to high level programming languages. We term views used for

building ontologies “Intermediate Representations” in deference to the knowledge acquisition community. Whichever name is used, views/intermediate representations serve at least five functions:

- 1) For accessibility, they *improve usability and productivity and reduce training time* – in past experience from months to days. Not only are the views simpler than the underlying representation, they are more uniform. Distinctions arising from the logical representation rather than the domain can be hidden. Furthermore, intermediate representations can be used to encourage standard styles and give domain experts a “place to start”.
- 2) In ontology development, they provide *a persistent representation of the domain experts’ intentions* which is independent of the precise formulation in the underlying logic. The underlying representation may be changed either for efficiency or to take advantage of new features in the underlying formalism without reformulating the knowledge at the level of the domain expert. This level of persistence is also important in change management to distinguish changes in users’ intentions from changes in technical implementation.
- 3) In ontology migration and adaptation, they provide a useful *intermediate target* and allow issues of domain knowledge to be separated from issues of implementation of that knowledge. This is particularly important to the Gene Ontology/GONG development.
- 4) In ontology use, they provide *a filtered presentation appropriate to individual applications*. This is particularly important for users needing to make modest extensions to large, complex ontologies.
- 5) In ontology formulation they provide *means of extending the expressiveness of the language* by allowing axiom schemas to be compiled to sets of explicit axioms and other meta information to be added to the lattice calculated by classifier. The classifier itself is, of course, restricted strictly to (a subset of) first order logic.

Our approach to views consists of three layers as shown in Figure 1. Working from the bottom up:

- a) Explicit representation of all context and content in the “assembly language” – *i.e.* the underlying generic reference ontology expressed in the logical formalism – OWL or a related description logic.
- b) Application specific “high level languages” – *i.e.* “views” or “intermediate representations” which simplify generic representations to fit specific needs and allow much context which is ‘obvious’ to users to be left implicit.
- c) Localised Presentations using phases and images appropriate to circumstances and preferences.

The abstract layers are shown schematically in figure 1a below with a concrete example in Figure 1b. Note that not only in the “View” level less complex than the “Assembly” level, but that it is more uniform. In the underlying representation, the expression would depend on whether “normal human hand” was implemented as a primitive or defined concept. Domain experts find such formal distinctions irrelevant to their knowledge and unintuitive.

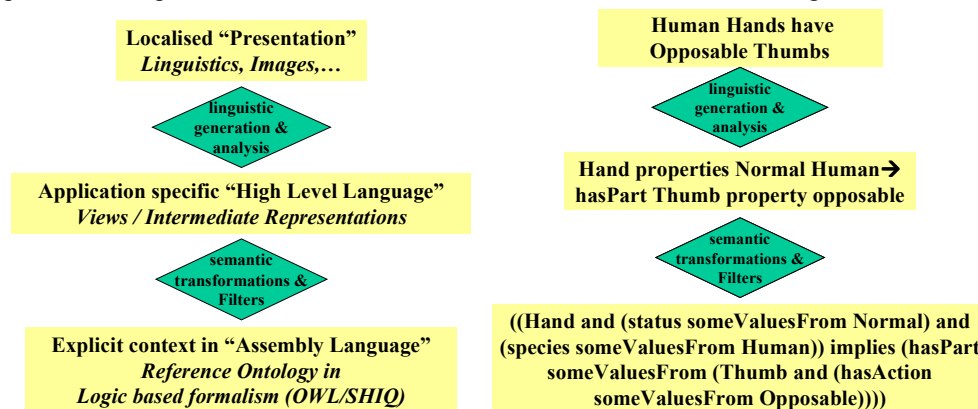


Figure 1a: Schematic structure of Views

Figure 1b: Example of structure of view mechanism showing simplification from bottom to top

In addition to the three levels shown in Figure 1, users often need to view an ontology in the light of the pragmatics of its use rather than its logical meaning – *e.g.* to navigate through the official reporting categories for regulatory agencies. Such legacy requirements are best dealt with by separate sets of mappings orthogonal to the three layers in Figure 1.

A system of “ontology views” for OWL therefore consists of five parts plus the environment to manage them:

- a) *The underlying representation in OWL making context explicit* – see (Rector, Rogers et al. 2002) and the metadata needed to indicate the information needed in each view.
- b) *Formal transformations from the underlying representation to the views*. Extending and formalising GALEN’s methods which use the class structure of the underlying representation to constrain the transformation rules.
- c) *Rewriting schemas for important notions which cannot be captured directly in OWL* because they are outside the description logic fragment of first order logic. These include axiom schemas which must be exploded as sets of axioms and mechanisms to deal with part-whole relations (Hahn, Schulz et al. 1999; Rector 2002). Details of the representation can have a dramatic effect on performance and inferences which can be drawn. The ability to experiment with the alternative rewriting schemas with different classifiers is expected to be important to making large ontologies scale computationally. The possibility of using hybrid rule systems such as those described by Rousset (Levy and Rousset 1996; Goasdoué and Rousset 2002) will also be explored.

- d) *The metadata schemas for provenance linking the three layers of representation.* The linking is required both for provenance and for providing effective reverse translations where the transformations are not uniquely reversible. It is anticipated that this will be developed as a set of RDF Schemas.
- e) *The mappings between the ontology and pragmatic views to deal with legacy requirements.*;

The proposed project concentrates on the bottom two layers as shown in Figure 1 plus the orthogonal pragmatic mappings. It will take as its starting point the techniques pioneered informally in the *OpenGALEN* (Rector, Zanstra et al. 1999) and *PEN&PAD* projects (Kirby and Rector 1996). Rather than being developed in the proposed project, the presentation and language layer will draw on “What you see is what you meant” interfaces developed by Scot (Bouayad-Agha, Scott et al. 2000; Scott and Power 2001) and will be funded separately.

3. Development Workpackages and Deliverables (See also attached Gantt chart)

3.1 Debugging and Explanation tools

- *“Checking” of constraints and restricted user interface* to constrain users to checkable paradigms (2.1-a,b). The first phase of ‘prechecking’ offers a large initial increment in usability at modest cost and can be combined with the companion project’s integration of the interface plug and play environment and tracking of simple ‘views’. A second phase will attempt more sophisticated checks. (Initial version pm 6; extended version pm 18)
- *Change tracking and dynamic comparisons of versions.*(2.2-c,d) The key problem for change tracking is ‘ontological comparison’. Preliminary experiments suggest that simple techniques offer large benefits as debugging aids and will give practical experience in working towards the more comprehensive comparison mechanisms needed for ontology integration. (Initial version pm 12; extended version pm 27).
- *Use of alternative incomplete reasoners and querying the reasoner as an ‘oracle’*(2.1-e). This is the most speculative of the techniques and the one which will benefit most from advances in reasoners allowing them to provide more information about the reasons for their inferences. It will therefore be investigated after other techniques with the use of the reasoner as oracle partly decoupled. (initial version pm 18; with oracle pm 21; extended version pm 30, with oracle pm 33)

3.2 Views Intermediate Representations and Transformations

- *Metadata and provenance schemas and software for linking levels, layers, and views.* (2.2-d). The basic metadata and provenance mechanisms need to be developed early in collaboration with the companion project so as to be integrated into the overall architecture and then revised in the light of experience. (Initial version pm 6; revised version pm 24)
- *Underlying representations in OWL.* (2.2-a). Initial versions sufficient for testing exist. However, experiments with the scalability of alternatives are required which will be much easier once transformation software is in place. (Preliminary versions pm 3; experiments with alternatives pm 24; final versions pm 33).
- *Formal transformations and the editing environment to manage them* (2.2-b,f,g) This is the heart of the project. The work requires a) formalisation of the transformation and rewriting mechanisms, starting with the *OpenGALEN* mechanisms for which prototypes exist, and b) iterative implementations and usability experiments which will use the Gene Ontology (GO) project as its primary testbed (Initial transformations; pm 6; iterative development of user interfaces and prototype conversions from GO pm 7-27 API pm 12; extended version pm 30)
- *Pragmatics layer and mappings* (2.2-e) This is a well understood problem from *GALEN*’s experience with classifications. It is expected to be implemented in a relatively short period (pm 18)

3.3 Requirements, Evaluation, Dissemination & Collaborations

Requirements analysis, evaluation and dissemination will run throughout the project and draw heavily on the collaborating projects. The primary evaluation will be in the use of the environment to improve productivity in the Gene Ontology (GO) funded through the myGrid project. Secondary evaluation will be through other associated E-Science projects requiring ontologies – CLEF, myGrid and Geodise based in Manchester and others such as AstroGrid and with the Digital Library community. In addition, the commercial collaborators CSW and Sun and the National Electronic Library for Health have agreed to provide inputs to the requirement and testing stages of this proposal and its companion project, and discussions are underway with pharmaceutical companies who collaborate in myGrid and CLEF. The project will be closely linked to its companion JISC funded project CO-ODE and will take advantage of its programme of user consultation, requirements analysis, and dissemination workshops. The tools developed will be open source, and dissemination will be through the E-Science programme and centres for which Manchester is ideally placed, as well as through the established user groups for both OilEd and Protégé which are involved in both this and the companion project.

The project is part of an ongoing collaboration with the Protégé team at Stanford University, and will take advantage of the established Protégé and OilEd user communities of several thousand each. In addition to Stanford University, international links include the Mayo Clinic, standards bodies, and the funding bodies the National Library of Medicine (NLM) and the National Cancer Institute Center for BioInformatics (NCICB).

4. Justification of Resources

A research fellow is requested to develop the view, intermediate representation, and rewriting mechanisms. A PhD studentship is requested to develop the heuristic debugging techniques. Both are requested over three years.

The project is part of a major collaboration with Stanford University and support is requested for one trip per year to Stanford to participate in joint workshops (a second workshop each year will be held in Manchester.) In addition support is requested for each member of staff and the principle investigator to attend one semantic web workshop or conference each year. It is intended that the team be integrated with other E-Science activities in Manchester, but the tasks proposed are distinct from any others currently funded. Sufficient resources to support four workshops per year per member of staff plus participation in the user requirements studies of the companion project are therefore requested. Major computing infrastructure will be supplied by the department, but support is requested for one PC each for the Research Associate and Research Student plus one laptop for the project for demonstrations and a printer for the workgroup. Under consumables, DLT back-up tapes has been requested to ensure the secure back-up of all data, and a small additional amount has been included to cover sundry items such as software, books etc, which experience shows will be required during the course of such a research project. Manchester University Computer Science Department runs a dedicated Research Office, providing substantial project management and secretarial support to relieve the investigators of routine, non-technical project management tasks. The level of support goes far beyond what is normally covered by indirect overheads. The secretarial and administrative charges support this facility at a level commensurate with the size of the research project in question. The system management and technical staff provide the support to enable networking and central computing resources to be used effectively as part of the university support for research.

5. References

- Bechhofer, S., I. Horrocks, C. Goble and R. Stevens (2001). "OilEd: a Reason-able Ontology Editor for the Semantic Web". *KI2001, Joint German/Austrian conference on Artificial Intelligence*, Vienna, Springer-Verlag: 396--408.
- Borgida, A., E. Franconi and I. Horrocks (2000). "Explaining ALC subsumption". *14th European Conf on Artificial Intelligence (ECAI-2000)*, IOS Press: 209-213.
- Bouayad-Agha, N., D. Scott and R. Power (2000). "Integrating content and style in documents: a case study of patient information leaflets." *Information Design Journal* 9(2-3): 161-176.
- Cawsey, A. (1992). *Explanation and Interaction: the Computation of Explanatory Dialogues*. Cambridge, MA, MIT Press.
- Gaines, B. and J. Boose (1988). *Knowledge Acquisition for Knowledge-Based Systems*. New York, Academic Press.
- Goasdoué, F. and M.-C. Rousset (2002). "Compilation and Approximation of Conjunctive Queries by Concept Descriptions". *Knowledge Representation Meets Databases (KRDB-2002)*, Toulouse, France, CEUR Workshop Proceedings.
- Hahn, U., S. Schulz and M. Romacker (1999). "Part-whole reasoning: a case study in medical ontology engineering." *IEEE Intelligent Systems and their Applications* 14(5): 59-67.
- Horrocks, I. (1998). "Using an expressive description logic: FaCT or Fiction". *Principles of Knowledge Representation and Reasoning: Proc Sixth Int Conf on Knowledge Representation (KR 98)*, San Francisco, CA, Morgan Kaufmann: 634-647.
- Kirby, J. and A. L. Rector (1996). "The PEN&PAD Data Entry System: From prototype to practical system". *AMIA Fall Symposium*, Washington DC, Hanley and Belfus, Inc: 709-713.
- Levy, A. Y. and M.-C. Rousset (1996). "CARIN: A Representation Language Combining Horn Rules and Description Logics". *European Conference on Artificial Intelligence*: 323-327.
- McGuinness, D. L. and A. Borgida (1995). "Explaining subsumption in Description Logics". *International Joint Conference on Artificial Intelligence (IJCAI-95)*: 816-821.
- Nowlan, W. A. (1994). "Clinical workstation: Identifying clinical requirements and understanding clinical information." *International Journal of Bio-Medical Computing* 34: 85-94.
- Rector, A. (2002). "Analysis of propagation along transitive roles: Formalisation of the GALEN experience with MedicalOntologies". *2002 International Workshop on Description Logics (DL2002)*, Toulouse France, CEUR-Proceedings 53: <http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-53/Rector-DL2002-propagates-via-final.ps>.
- Rector, A. (2002). "Normalisation of ontology implementations: Towards modularity, re-use, and maintainability". *Workshop on Ontologies for Multiagent Systems (OMAS) in conjunction with European Knowledge Acquisition Workshop*, Sigüenza, Spain.
- Rector, A., J. Rogers, A. Roberts and C. Wroe (2002). "Scale and Context: Issues in ontologies to link health- and bio-Informatics". *AMIA Fall Symposium*, Austin Texas, Hanley and Belfus, Philadelphia: 642-646.
- Rector, A., C. Wroe, J. Rogers and A. Roberts (2001). "Untangling taxonomies and relationships: Personal and practical problems in loosely coupled development of large ontologies". *Proceedings of the First International Conference on Knowledge Capture (K-CAP 2001)*, Victoria, BC, Canada, ACM: 139-146.
- Rector, A. L., P. E. Zanstra, et al. (1999). "Reconciling Users' Needs and Formal Requirements: Issues in developing a Re-Usable Ontology for Medicine." *IEEE Transactions on Information Technology in BioMedicine* 2(4): 229-242.
- Rogers, J. and A. Rector (1996). "The GALEN ontology". *Medical Informatics Europe (MIE 96)*, Copenhagen, IOS Press: 174-178.
- Scott, D. and R. Power (2001). Generating textual diagrams and diagrammatic texts., *Cooperative Multimodal Communication*. H. Bunt and R.-J. Beun. Berlin/Heidelberg, Springer-Verlag.
- Shadbolt, N. and K. O'Hara (1997). "Model-Based Expert Systems and the Explanation of Expertise". *Expertise in Context*, AAAI Press: 315-37.
- Solomon, W., C. Wroe, J. E. Rogers and A. Rector (1999). "A reference terminology for drugs." *Journal of the American Medical Informatics Association*((Fall Symposium Special Issue)): 152-155.
- Wroe, C., R. Stevens, C. A. Goble and M. Ashburner (2003). "An Evolutionary Methodology To Migrate The GeneOntology To A Description Logic Environment Using DAML+OIL". *Proc 8th Pacific Symposium on Biocomputing (PSB)*, Hawaii: in press.

Outline Gantt Chart of Project Plan

	Year 1				Year 2				Year 3			
	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12
Initiation, Requirements & alignment with companion projects												
Project initiation and planning												
Requirements confirmation ¹⁷												
Debugging and Explanation Tools												
Checking constraints & restrictions on expression												
Change tracking & dynamic comparison of versions												
Use of alternative graph reasoners												
Use of additional heuristics and “Reasoner as Oracle”												
Views, Intermediate Representations & Transformations												
Metadata & provenance schemas for linking views & formalism												
Experiments with underlying representations in OWL ¹⁸												
Formal transformations specification												
Formal transformation implementation ^{19,20}												
Pragmatics layer												
Evaluation and Dissemination												
Evaluation with local projects ¹												
Dissemination in E-Science Community ¹												
Final Completion												
Write up period for PhD Student												
Final documentation and reports												

¹⁷ A process of iterative requirements refinement will run throughout the project, but much of the primary analysis is already performed but must be confirmed in collaboration with companion project. Final evaluation will again be in conjunction with the companion project both locally and in the E-Science community.

¹⁸ Initial versions exist sufficient for other testing

¹⁹ First phase required sufficient for experiments with specification. Subsequent phases iterate with Formal transformations specification & with experiments with underlying representation.

²⁰ API specification Month 12.