Putting OWL into production at the European Bioinformatics Institute

James Malone, Tony Burdett, Jon Ison, Simon Jupp, Drashtti Vasant, Dani Welter, and Helen Parkinson

European Bioinformatics Institute, Cambridge, CB10 1SD, UK

Abstract. The Experimental Factor Ontology (EFO) is an OWL based ontology of experimental variables used in a wide range of biomedical studies. EFO has been in use since 2007 and was the first OWL ontology to be used in production ready data services at the EBI. As the ontology content, application requirements and the OWL language have evolved over time, this has presented several challenges in how to develop EFO and how best to deploy this within the applications. In this paper we describe our experiences of using OWL and the challenges we have faced which include; managing imports from multiple ontologies, optimising axiomatisation for performance, displaying the ontology in domain specific views, incorporating the ontology into a continuous integration framework and deploying the ontology in applications and curation tools. We also describe some of the tools we have developed and applications EFO is now used in and, finally, discuss our future challenges.

1 Introduction

The European Bioinformatics Institute (EBI) provides freely available data from life science experiments to help perform research in computational biology. One of the domains for which EBI provides data is in functional genomics. At EBI, two databases exist in this area; ArrayExpress Archive [1] and the Expression Atlas [2]. The ArrayExpress Archive is a database of functional genomics experiments that are either directly submitted by individuals or groups or imported from the Gene Expression Omnibus. ArrayExpress is curated manually by a team of highly trained experts to ensure a level of quality and consistency in the data and the annotations describing them. The Expression Atlas uses a subset of ArrayExpress to produce a meta-analysis about which genes are expressed under which conditions (commonly called experimental factors).

A number of use cases were elicited which supported the development and deployment of an ontology for these resources. These were as follows: To encourage consistency in the annotation of these data; Integration across like experimental conditions for performing meta-analysis; query support to allow simple subclass and partonomy query expansion such that a query for brain returns experiments for brain parts; visualization, to facilitate data exploration; consistency checking to ensure otherwise disjoint concepts were not used in an inconsistent manner such as marking the same sample as healthy and cancerous.

To support these we developed the Experimental Factor Ontology (EFO) [3]. EFO is an OWL based application ontology in that it is built towards a specific application's use cases distinct from reference ontologies which are considered to be the de facto point of reference for a given domain. ArrayExpress and Atlas contain a considerable diversity of variables, touching upon many difference biomedical domains, as is summarised in Table 1. In order to provide coverage of these variables, EFO consumes fragments of multiple reference ontologies and enriches them by providing additional axiomatisation to satisfy specific queries. For instance, EFO imports a class *epithelial cell* from the Cell Ontology [4] and *Homo sapiens* from the NCBI Taxonomy and uses these in class descriptions for a cell line which is derived from a human epithelial cell. These additional axiomatisations do not exist elsewhere.

Table 1. Annotations within ArrayExpress and Atlas

Annotations	ArrayExpress	Atlas
Species	1,821	73
Samples	1,126,456	98,631
Annotations on sample	7,672,825	463,797
Unique sample annotations	223,650	22,339
Assay	965,638	100,654
Annotations on assay	3,248,298	265,971
Unique assay annotation	189,381	19,136

The development of EFO and deployment into applications has brought with it a series of challenges for which we have had to develop new approaches and tools. In this paper we highlight our experiences of building and deploying OWL ontologies in real-world, high-usage biomedical applications. We outline the continuous integration system we use to ensure applications do not fail, the compromises which we have had to face due to technology limitations and thoughts on how our experience can lead to more adoption of OWL ontologies in applications.

2 Consuming the delicious OWL

Several applications now consume EFO in OWL. Both ArrayExpress and Atlas use the ontology as a search index mechanism. No real time reasoning is undertaken; instead an inferred version of the ontology is created and traversed to browse the subsumption hierarchy to expand queries, as are classes used in *part of* restrictions to fetch partonomies. This also uses annotation properties to expand over synonyms. In Atlas, the ontology tree is displayed along the subclass axis to visualise the experimental variables and can be visually expanded and explored by the user. Another application is the National Human Genome Research Institute Catalog of Published Genome-Wide Association Studies (or GWAS Catalog) which provides a quality controlled, manually curated, collection of published GWA studies [5]. The studies in the Catalog cover a wide range of often context-dependent traits and phenotypes, diseases and clinical measurements. These traits are annotated to EFO to improve the querying capabilities and concepts in the Catalog, such as SNP, study and trait, and the relationships between them are modeled in an OWL knowledge base which is used as the back end for this application. This provides advanced querying capabilities and also drives the new GWAS diagram.

3 Developing an OWL ontology for applications

The software engineering paradigm used within our developer team is that of Agile Software Development. One of the underlying methods of Agile development is that requirements can change over time and should be responded to in a prompt fashion. The continuous integration and test-driven development methods enable working version of software to be produced in iterative cycles, whilst keeping user needs at the fore. When developing EFO we have adopted the same approach and this has necessitated that we treat the OWL ontology as a component of software, much like any other in our development framework.

3.1 OWL supporting tools, methods and version control

To actually populate the ontology we have developed several pieces of software that support our production process. The first is an OWL importer[1] which is an implementation of the MIREOT specification [6]. For creating new classes native to EFO, we have developed URIGen[2] for controlling the minting of new URIs. URIGen is a client server tool, available as a Protege plugin that serialises the creation of a URI to ensure that multiple users, editing in Protege across multiple machines, do not produce URI duplications. We have also introduced our own bespoke rules for creating views of EFO due to the lack of a formal mechanism for this in OWL. Because of the multiple applications which consume the ontology, we have required a mechanism to produce different views to suit the different user groups. Several mechanisms are used; an annotation property flagging if the class should be included within a view set, inclusion of an alternative label to be used for a given view set and an annotation property used as a flag for classes which we wish to hide (such as upper level classes).

EFO is produced and edited by a multi-developer team, and as with any other multi-developer project a version control system is critical to ensure developers can synchronize their changes. We use a Subversion repository to store both our current development version of EFO and archive previous releases. OWL does not lend itself to analysis with conventional "diff" tools, since there is no ordering requirement to each class description within a file. Nevertheless, understanding the difference between one version of an ontology and another is of much use since it can help debug an ontology when something has broken (such as the ontology

[1] http://www.ebi.ac.uk/fgpt/sw/efoimporter
[2] http://www.ebi.ac.uk/fgpt/sw/urigen

becoming inconsistent) and this also helps us to produce release notes for our users so they can also be aware of changes. We developed and use the Bubastis tool[3] to perform a syntactic diff between named class descriptions within the OWL to highlight axiomatic changes, deletions and additions of named entities.

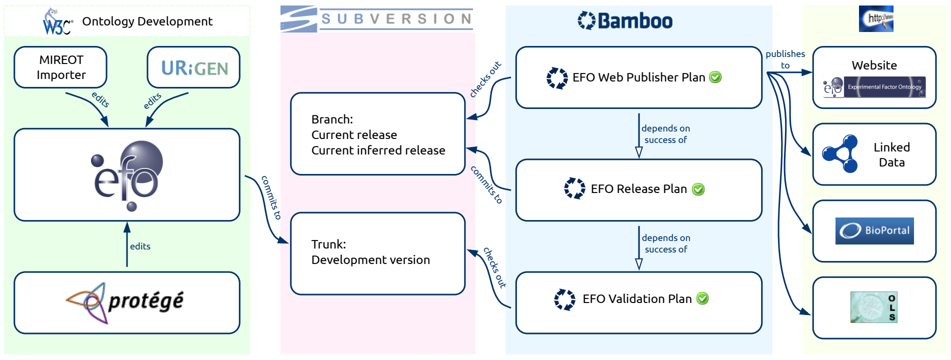

Fig. 1. The development and deployment process of EFO in an agile framework.

3.2 OWL in an Agile framework

There are several aspects of OWL ontologies that lend themselves well to software engineering approaches. The first is the use of test driven development. In a continuous integration framework, it is necessary to test each commit of code to ensure that it does not break previously working components and introduce new bugs and we treat OWL with the same respect. We have developed a series of automated tests using Bamboo[4] that the ontology is ran against after each commit which performs checks such as for: invalid namespaces; IRI fragments outside accepted conventions; duplicate labels between different classes; synonyms duplicated between classes; obsolete classes used in axiomatisation; unit tests for expected class subsumption (e.g. cancer should be subclass of disease).

Another aspect is performance and the OWL DL profile we restrict to. In order to fully exploit the querying power of the ontology, we use reasoning to infer various hierarchies of interest, such as classifications of cell lines by disease and species, and we need this to happen in a time that is responsive. There are several methods we use to ensure this remains the case. The first is the use of design patterns. We restrict axiomatisation to a set of patterns that we have developed to answer our priority competency questions. The second is to disallow the addition of new object properties and characteristics on those properties. The third is to classify the ontology on every commit.

[3] http://www.ebi.ac.uk/efo/bubastis
[4] http://www.atlassian.com/software/bamboo/

We also employ an automated release cycle to release a new version of EFO monthly, in order to best coordinate with our application needs. The release is programmatically performed using a Bamboo build plan which performs tasks such as creating the inferred version of the ontology, converting the ontology to OBO format, publishing files to the web, building the EFO website and creating URLs for classes in the EFO namespace to ensure that concepts described in EFO fully dereference.

4 Challenges and limitations of using OWL

Although we have managed to use OWL in several applications for a number of years, we have had to make several compromises. One of the primary challenges is in producing an ontology which relies on several external ontologies. OWL is designed to import external resources, however there are implications in doing so. Importing full ontologies into EFO such as the Gene Ontology would bloat EFO and introduce redundancy. The MIREOT method [6] aims to bypass some of these issues but ensures that the richness of axiomatisation is lost. OWL module extraction can import fragments but it also suffers from bloat; a potentially small number of imported classes can increase by ten fold or more. It is apparent there is no simple solution to this approach. If we consider an example in EFO - importing hierarchy from the NCBI Taxonomy, the subclass tree between *Eukaryota* and *Homo sapiens* is 28 deep yet our needs are to import only a tiny fragment of this. In one sense, importing all of this is not an OWL problem - it is an artifact of the ontology, however it speaks to the issue of how one utilises such ontologies within applications that a user will see. Producing views of OWL is a much over-looked requirement and we believe the lack of an agreed specification for generating them is a limitation. One proposal is to use SKOS [7] to indicate subsets for creating user friendly visualisations.

Reasoners and serving queries such that they respond in subsecond times still presents limitations for applications with large data stores such as GWAS. Due to the size of the GWAS knowledge base (more than 18600 individuals in the current release), and the strong reliance on data properties to represent variables such as author name and publication date for studies or p-value for SNP-trait associations, reasoning over the GWAS knowledge base can be slow. While large reasoning times can be considered an acceptable cost at initial start-up, this also affects querying and multi-factor DL queries are often unscalable. In order to retain the power provided by reasoner inference while mitigating scalability issues, we are currently exploring a hybrid strategy using the GWAS knowledge base both in its original OWL format and as an RDF triple store.

5 Conclusion

The use of OWL in a production environment requires similar considerations to using software components. One of the most challenging is in the coupling of an application to imported ontologies. If we are to utilise the growing number of

OWL ontologies in areas such as biomedicine, then we need ways of coping with external change which is outside the control of consumers. A recent study [8] found that a large number of ontologies frequently delete named classes from their ontologies which could have detrimental effects to applications relying on them. This is a strong disincentive for application developers to reusing such ontologies. Treating OWL classes as dynamic resources may be true to the nature of the language, but it is a barrier to use in biomedicine where provenance and persistence is of great importance. Similarly, methods for rendering OWL for different applications is an under-developed area. A mechanism or guidance for generating view sets would help in deployment of OWL into applications.

6 Acknowledgments

Funding: EMBL and The National Center for Biomedical Ontology, one of the National Centers for Biomedical Computing supported by the NHGRI, the NHLBI, and the NIH Common Fund under grant U54-HG004028.

References

1. Rustici, G., Kolesnikov, N., Brandizi, M., Burdett, T., Dylag, M., Emam, I., Farne, A., Hastings, E., Ison, J., Keays, M., Kurbatova, N., Malone, J., Mani, R., Mupo, A., Pedro Pereira, R., Pilicheva, E., Rung, J., Sharma, A., Tang, Y.A., Ternent, T., Tikhonov, A., Welter, D., Williams, E., Brazma, A., Parkinson, H., Sarkans, U.: Arrayexpress updatetrends in database growth and links to data analysis tools. Nucleic Acids Research **41**(D1) (2013) D987–D990
2. Kapushesky, M., Adamusiak, T., Burdett, T., Culhane, A., Farne, A., Filippov, A., Holloway, E., Klebanov, A., Kryvych, N., Kurbatova, N., Kurnosov, P., Malone, J., Melnichuk, O., Petryszak, R., Pultsin, N., Rustici, G., Tikhonov, A., Travillian, R.S., Williams, E., Zorin, A., Parkinson, H., Brazma, A.: Gene Expression Atlas update - a value-added database of microarray and sequencing-based functional genomics experiments. Nucleic Acids Research (2011)
3. Malone, J., Holloway, E., Adamusiak, T., Kapushesky, M., Zheng, J., Kolesnikov, N., Zhukova, A., Brazma, A., Parkinson, H.: Modeling sample variables with an Experimental Factor Ontology. Bioinformatics **26**(8) (2010) 1112–1118
4. Meehan, T., Masci, A., Abdulla, A., Cowell, L., Blake, J., Mungall, C., Diehl, A.: Logical development of the cell ontology. BMC Bioinformatics **12**(1) (2011) 6
5. Hindorff, L.A., Sethupathy, P., Junkins, H.A., Ramos, E.M., Mehta, J.P., Collins, F.S., Manolio, T.A.: Potential etiologic and functional implications of genome-wide association loci for human diseases and traits. Proceedings of the National Academy of Sciences **106**(23) (2009) 9362–9367
6. Courtot, M., Gibson, F., Lister, A.L., Malone, J., Schober, D., Brinkman, R.R., Ruttenberg, A.: Mireot: The minimum information to reference an external ontology term. Appl. Ontol. **6**(1) (January 2011) 23–33
7. Jupp, S., Gibson, A., Malone, J., Stevens, R.: Taking a view on bio-ontologies. In: ICBO. (2012)
8. Malone, J., Stevens, R.: Measuring the level of activity in community built bio-ontologies. Journal of Biomedical Informatics **46**(1) (2013) 5 – 14