The SWO Project: A Case Study of Applying Agile Ontology Engineering Methods in Community Driven Ontologies

Maria Copeland [1], Andy Brown [1], Helen Parkinson [2], Robert Stevens [1] and James Malone [2]*

[1]Department of Computer Science, University of Manchester, Manchester, UK
[2]European Bioinformatics Institute, Cambridge, UK

ABSTRACT

The Software Ontology Project (SWO) is a community effort to build an ontology that models software used in the generation and analysis of data for curation and preservation purposes in areas such as biomedicine. In community driven efforts, requirements are elicited from the members of these communities to help ensure the ontology is fit for purpose. This requires methods which are able to engage with users with a wide range of expertise, allow close collaboration between developers and users and that are able to respond rapidly to changing knowledge. We describe an Agile Ontology Engineering method for developing ontologies, adapted from modern Agile software engineering methods. The approach was applied within the SWO project and demonstrated promising results in engaging a diverse set of community representatives and an objective measure of ontology success, of much relevance to the active community of bio-ontology developers and users.

1 INTRODUCTION

Community driven development of ontologies have become the norm in biomedicine (Garcia *et al.*, 2010; The Gene Ontology Consortium, 2010). Engaging with a diverse set of users from a community presents challenges; some of these challenges are technical, such as when multiple ontology authors need to synchronously or asynchronously edit from multiple sites, and to allow users to view ontologies and directly comment on the artefacts (Alexander *et al.*, 2011). For such challenges, tools such as CollaborativeProtégé (Tudorache *et al.*, 2008) (CP) and resources such as BioPortal (Noy *et al.*, 2009) have been developed, and versioning tools can be used to support the use of code bases in a similar way as for software projects. Tools such as CP offer technical solutions for the technically savvy, but methods for collaboration need to be much broader than addition of axioms (a necessary, but not sufficient task for ontology building). Some of these challenges are more about the process used and are sociological in nature, as much as technical (Randall *et al.*, 2011). They include, but are not limited to: working with people that are not ontology engineers; the ability to appropriately capture requirements from a diverse set of users; prioritising those requirements by seeking group consent and resolutions; judging when an ontology meets a community's requirements; responding to issues, requests and shifts in domain knowledge; and take into account and predicting the various types of the ontologyś users.

We used the Software Ontology (SWO) Project to explore ways of including a broad range of user backgrounds into a community building process. The SWO is a community driven, collaborative project with the aim of producing an ontology that captures formal descriptions of software used in the production and analysis of data for curation and preservation purproses, in order to promote standardisation and reusability of knowledge (Brown, 2011). This need is particularly pertinent in the computational biology field, where an understanding of how data were analysed is an important part of the scientific record that allows proper re-interpretation and re-use of the discipline's data.

There is a need in many Ontology Engineering efforts to engage with a community of domain experts that may not be ontology or knowledge representation experts. This presents a challenge of how to engage an audience without having to learn their domain knowledge or for them to become ontology engineers. In addition to this challenge, there are difficulties of keeping the community engaged, the ontology up-to-date and responsive to the customers. Some of these challenges in community ontology development are similar to those seen in Software Engineering.

One popular method in software engineering for tackling these issues is Agile Software Development (Martin, 2002) which is a set of software engineering methods that drives the development effort around the requirements, use cases, and continuous participation with users. Agile methods require that iterations are frequent and that software is released often with an emphasis on collaboration between developers, especially with regards customer requirements. Typically through these short iterations, the whole life cycle is reproduced, including some requirements analysis, testing and product delivery. Some of the advantages of this approach are that changing requirements are considered part of the normal life cycle and that the users are more 'involved' with the development process, making this approach more able to respond to change as determined by the customer. There is also an emphasis on working code and test-driven development, so the software works for each of the features added at each iteration.

In community ontology projects like the SWO, the community should act as the customers from which requirements are elicited and used to help make the ontology fit for purpose. Unsurprisingly then, ontology engineering methods have been developed that aim to tackle similar aspects of the development process as in Software Engineering. In a review of ontology engineering methods, Tempich and colleagues (Tempich *et al.*, 2006) were critical of many methods for their lack of consideration for evolution of ontologies, thus indicating a failure in continuing a collaborative relationship with the community of users as part of the ontology's evolution. A wider study surveyed 148 ontology engineering projects in academia and industry, concluding that ontology engineering has become

*To whom correspondence should be addressed: malone@ebi.ac.uk an important discipline, though there is still work to be done in the area, such as the need for better documentation of decisions taken during the ontology engineering process (Simperl *et al.*, 2010).

In response to such needs, some aspects of Agile methods are beginning to be adopted by the Ontology Engineering community. One such method is the RapidOWL method (Auer and Herre, 2007), where they claim that the adoption of agile methods for Ontology Engineering is greatly benefited when these methods are carried out in a unified process and with the continuous involvement of the user community. This is a positive step in community ontology development, however more analysis, case studies, and empirical evaluations are needed in order to recommend and refine the application of Agile Ontology Development.

In this paper we present a case study in which an Agile Ontology Development method was applied to the SWO project. We use the SWO as a case study of how an agile software approach can be adapted to the development of an ontology. We also reflect on what we did in our *agile* method and what we think could and should be changed to make it better.

2 SWO 'S AGILE METHOD

Eliciting requirements from and engaging with the life science community is important to ensure ontologies meet the needs of the stakeholders. Given a broad similarity of activities within both software and ontology development (requirements gathering, evaluation, design, implementation, testing, publishing, maintaining, etc.), moving methods from one to the other has an *a priori* appeal. The organisers of the SWO project approached the ontology building task by adapting agile methods from software development into the engineering process for ontologies. SWO focused on the following agile principles (Martin, 2002):

- The introduction of requirements gathering and ontology modelling sessions as iterative and incremental activities;
- That requirements evolve throughout the engineering cycle;
- The encouragement of self-organised and cross-functional teams;
- The provision of rapid and responsive ontology development;
- That domain experts, users, and ontology engineers are all active contributors throughout the process;
- The use of a test driven approach to development;
- The provision of regular and frequent builds to the participants for discussion, testing, refinement, and agreement.

These methods have several events whose activities are planned to deliver information to other events in a cyclic manner. Events can, however, sometimes run in parallel. The Agile Ontology Engineering Method is summarised as (see also Figure 1):

- **Requirements Gathering Event:** This event focuses on the capture of requirements by identifying competency questions and desirable features for the ontology. These activities are use case or scenario driven.
- **Requirements Prioritisation Event :** This event has two parts, both adapted from Agile Software Engineering techniques. The first requires participants to estimate the complexity required to implement a requirement based on 'planning poker' (Grenning, 2002). The second part has participants rank features collected in the requirements gathering phase by importance by 'bidding' for individual features based on the 'Buy a Feature' method (Kirk, 2011).
- **Implementation of Top Requirements Event:** The modelling and coding of the ontology takes place, focusing on the features 'bought' from the previous step. Modular development is used to allow concurrent development from co-located or distributed developers. Content is also gathered by participants completing templates taken from the implemented ontology.
- **Evaluation Event:** Evaluation of the ontology is done by all participants of the development process, thus helping competency questions to be satisfied. Testing is conducted with defined classes acting as queries based on the competency questions against the ontology. This is disseminated to the stakeholders who evaluate the ontology against the requirements.

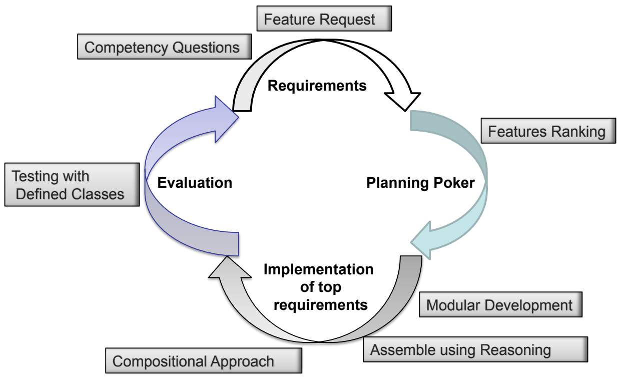

Fig. 1. SWO 's Ontology Development Flow of Events.

Requirements Gathering Event The following ground-rules were used in the SWO workshops:

- Presenting the community of users and contributors by introducing stakeholders and typical users of the ontology;
- 'No death by PowerPoint'; the workshops are hands on events where results and instant feedback will be the focus of activity;
- No up-front 'Ontologising' in the workshops; the resolution of the pattern of axioms for representing the user 's needs is vital, but does not need to be done interactively in this setting, especially with those that are not familiar with ontologies;
- Everyone participates. Activities were designed such that everyone could join in and contribute and that 'powerful' voices could not dominate.

These rules of engagement were overseen by a moderator to help keep the level of details and discussion appropriate for the activities. The universe of discourse was created by asking participants to organise into groups and write on 'sticky-notes' the information they wanted to record about software. The notes were then clustered according to similarity by the participants, invoking more discussion. A similar exercise was used for gathering competency questions. Both sets of clusters were reviewed for correspondence between features and questions and any gaps highlighted by one cluster in the other set of clusters were discussed and addressed.

The validation of the clusters and understanding of competency questions were conducted by the next activity which instructed users to create personas. A persona represents an individual user of the ontology with specific characteristics that provide context to the interactions and needs of that particular user. In contrast to use cases, personas define a specific individual of a user group with a detailed 'story' of the user and an example of usage of the ontology. The moderator asked for each persona to give: their age, background, dress code, favourite food, work, detailed task that include the competency question in the form of a story. Persona are meant to motivate participants to have some basis for the decisions they make, rather than making decision based on personal preference. The use of personas is a requirements gathering method that aims to reduce the effect of sample bias in participant events.

Requirements Prioritization Event - Planning Poker: The set of clusters, questions, and usage examples were prioritised by adopting the Agile technique of Planning Poker (Grenning, 2002). The SWO project 's version of this technique was divided into two voting activities. In the first voting activity participants were asked to rate the difficulty of describing the clusters by casting a vote using 'voting' cards with '?', '0', '1', '2', '3', '5', '8', '13', '20', '40', and '100'; where zero is the easiest and 100 is the hardest. Discussion was encouraged by asking participants to contribute their views when their vote was not close to the average vote for that cluster. The consensus or mean was used to derive a cost for the feature in question.

In the second voting activity users were given 100 points to spend on the clusters they wanted to be represented using the 'Buy a Feature' game. In this activity voting and polling drives the prioritisation; if a discussion about a cluster was prolonged a re-vote was taken to allow participants the opportunity to change their investment after the discussion. The buying phase results in a set of features that the participants most wish to have in the ontology. Such prioritisation events should be iterative over the entire life of the ontology.

Implementation of Top Requirements Event: Implementation of the requirements was conducted in a modular development approach using normalisation Rector (2003). The requirements indicated the main modules to be made (for example *data* in a data OWL module). Classes of Software are then described in terms of these feature modules via restrictions, and defined classes of software establish the hierarchy of software. In addition to versioning standards, other standards such as coding standards, URI naming conventions, and labelling standards were chosen or devised, and implemented by the ontology engineers. The Implementation Event ran concurrently with the Evaluation Event in order to adhere to the Agile principle of continuous and gradual testing and validation.

In the Implementation Event, user contribution continued by allowing users to populate templates of software descriptions as part of the development process. These templates, based on the axiom patterns in SWO, were used to both gather input for software descriptions and to test the ability of the SWO to enable descriptions. The spreadsheet had a software entity in the first column and subsequent columns represented the prioritised features for software descriptions. Participants either filled in from the existing SWO 's software descriptions or provided their own new terms.

Evaluation Event: Evaluation of the ontology was conducted by combining the information from the Requirement Gathering Event, namely software clusters and their descriptions, with test cases. The SWO ontology engineers produced a series of software description test cases based on the competency questions and samples supplied by the participants during the two workshops, the email lists and blog posts. All commonly used test cases were covered without spending too much time on rarely used software descriptions not covered in the test cases. The ontology was tested with the use of personas and competency questions through the participation of all SWO project members throughout all the events in the development cycle.

Table 1. Features that were initially identified, then given complexity estimate (cost) and then later prioritized through a 'bidding' process. *indicate features that were bought (i.e. their cost was met)

Feature	Cost	Total Bid
Algorithms*	75	75
Architecture	87	0
Capability	254	247
Configure/run parameters	542	174
Cost of ownership	295	70
Data*	300	300
Dependencies	276	91
Function*	44	44
Interface*	74	74
Licenses*	38	38
Life cycle*	188	188
Platform	169	50
Source code location*	25	25
Supplier*	14	14
Version*	110	110

3 RESULTS

The SWO project had a six-month duration. The Requirements Gathering Events took place during the first face-to-face workshop (WS1) and then again in an iteration four months later in a second workshop (WS2); there were 18 participants in WS1 and 13 participants in WS2. Seven of these people participated in both workshops. The first set of requirements were produced following the activities described in section 2. This resulted in 17 clusters of features (formed by clustering possible features of interest) and 91 competency questions aligned to these features. The matching of competency questions to feature clusters provided interesting validation regarding what 'can be said' as opposed to what is 'desirable to ask' about a feature. For instance, one of the main clusters identified was software 'platform'. Platform had 10 sticky notes about information that can be recorded about a platform's software requirements (e.g. will it run on Linux, Android etc.); however, when questions were solicited from participants, only 4 questions were asked. The moderator highlighted this discrepancy and explained the usage of a cluster is as important as the detailed recorded in the cluster.

Following the initial gathering activities, the Requirements Prioritization Event took place. Table 1 presents a summary of the features bought by the method described in section 2. It should be noted that the 'cost' of each feature does not translate directly to a real effort or cost (e.g. days or money). The metric is meant as a comparable measure within features only.

Table 2. Examples of competency questions for some of the features and how they were met in the ontology, evaluated using an OWL defined class, which equated to the competency question. The example answers are actual answers given from the ontology (not exhaustive).

Feature	Competency Question	Manchester OWL Test Question	Example Answer
Data	Which software has MAGE tab input?	has_specified_input some (data and has_format_specification some 'MAGE tab format')	ArrayExpress Bioconductor
Function	What software peforms molecular sequencing analysis?	achieves_objective some 'molecular sequencing analysis'	EMBOSS
Algorithm	Which software implements a Bayesian Model	implements some 'Bayesian Model'	GeneSelector
Interface	What software has command line interface?	has_interface some 'Command Line Interface'	DROID
Version	Which version Microsoft Excel came after 2007?	'Microsoft Excel' and (has_version some ('version name or number' and (preceeded_by value 'Microsoft 2007 version')))	Microsoft Excel 2010

Participants continued to contribute to the population of descriptions and clusters by providing examples of usage, software descriptions, and new entities via the emailing list and blog post.

After four months WS2 was held at which progress was evaluated. An initial exercise of asking the participants to each describe software, following the requirements set out previously, was conducted. Then, a re-prioritisation occurred, with the intention of finding out if, given the current ontology and experience of describing software, the current set of priorities were still relevant. Some new features that emerged from this included the need to capture software suites or packages, and software documentation. the platform upon which the software runs was seen to be a more important feature in WS2, although it was not bought in WS1. Axiom patterns changed according to feedback and the descriptions captured in the spreadsheets used to evaluate SWOS's ability to describe software were added to SWO. These spreadsheets captured extensions to various parts of the SWO, so all parts of the SWO's descriptions of software features changed. Three iterations occurred, resulting in three releases during the six-months. The final ontology after iteration 3 had 903 classes, 101 individuals, and 34 properties.

4 DISCUSSION

In our experience of building SWO, the use of Agile methods in ontology development appeared to have several strengths that should be of relevance to the biomedical community. There is, of course, no formal evaluation of the process; what we report here is reflection on what we did with the SWO and in our involvement in other ontology development efforts. During the SWO project the link between participation of its members and its corresponding effects on actions in the development cycle were self-evident. The focus of development was the community of users rather than on a particular ontology technology, formal ontology or philosophical paradigm. The 'ontologising' is, of course, important, but we treated it only as a means to an end. Also, the process we describe is neutral to ontological paradigm.

This emphasis on community and commitment to Agile principles appears to us to have produced the following benefits:

- An open and transparent process, present throughout the development cycle of the ontology. This seemed to have encouraged participation and a sense of membership to the SWO project as a whole rather than unconnected contributions to prescribed phases.
- Users' competency questions, personas, and test cases were important drivers and validators of the development process.
- The resulting ontology covered the knowledge needed by the community as communicated through concrete examples, users' descriptions, and competency questions.

The approach enabled the SWO developers to engage with users with a varied background and experience in ontology building. Around 80% of those who contributed to the workshops and requirements had no experience of ontologies or OWL, yet all participants actively contributed to the ontology. The SWO 's approach relied on the leadership of the moderators and organisers of the project during meetings with users. The organisers were familiar with, but not experts in, Agile techniques; this suggests that the techniques may be easily adopted.

An element of the workshops that appeared to be particularly successful was that of conflict resolution with regards to estimating the effort required to add features. One of the benefits of the planning poker game was that it appeared to allow for both an independent estimate of effort and then, if estimates were individually far apart, discussion as a group to resolve those discrepancies. Several such discrepancies occurred during the initial SWO workshop, often as a result of disagreement in exactly what adding such a feature would require. Discussion then helped to come to a common understanding and a re-estimate was taken, often with closer agreement.

Deciding what was to be included in the SWO was a collaborative effort using the buy a feature game. The vast majority of features were not able to be bought by any one individual and thus co-operation in bidding was required. There were some features on which much of the 'money' spent on that feature was done so by one individual in an attempt to have it 'accepted', despite no one else bidding for that feature. Some features may be important for one domain but less so for another; such features may be 'showstoppers' for that community and this could be taken into account in the process. An example of this was the 'algorithm' feature which was bid for by one person with a large amount of money, but not bought outright as it was deemed less important to almost all other users.

This feature buying process helped to reduce development time on areas that were collectively deemed unimportant. This is of particular importance in the life science domain, in which the scope is vast. Prioritising areas of importance should provide a cost benefit.

Although one of the Agile principles was to encourage self organised and cross-functional teams, this would not have been possible without the input from the project 's organizers. Skills in group organisation, team building, and responsive feedback were as important as the method implemented. Experience of community ontology building was also a factor that allowed the organizers to prevent and resolve conflict during the requirements gathering. The cyclical nature of the method allowed for a continuous feedback process to all participants and presented working versions of the ontology as requirements and competency questions were refined.

One area of the method that was under employed was the 'persona'. Personas are used in Software Engineering with the purpose of validating specific user interacting features such as user interfaces. Requirements not only cover features but also cover the needs of stakeholders that may or may not be explicitly known to the users. SWO could have increased the scope of the ontology by combining use cases and personas, and by utilising both as assets to the different phases of the project such as during Planning Poker.

Consistency of modelling was achieved in SWO by the establishment and enforcement of standards, the ontology engineers' communication and participation in requirements gathering and prioritising sessions, and effectively using communication tools such as the SWO blog to resolve modelling conflicts and questions. These observations suggest that participants, users, and engineers valued open communication and collaboration throughout the project and benefited from a mutual understanding of each other's background and motivation. The commitment to shared values and trust in a methodology are social questions that are hard to quantify and reproduce in all projects, but they appeared to help make this instance of the agile method work.

5 CONCLUSION

This paper presented a reflection on one projects experience of applying an agile method to ontology development in the Software Ontology (SWO). The success or failure of the adoption of Agile methods in Ontology Engineering have much to do with the delivery of such methods under an organised, flexible, responsive and collaborative social and technical environment. The technical tools employed to deliver collaborative ontologies must be complemented with organisation, responsive feedback, and transparency of process. The SWO project should be judged not only on the coverage and consistency of its knowledge representation, but on the active participation of its members in making it a maintainable ontology grounded in its communitys needs. Such methods should be important to the bio-ontology community, where user participation is key in building community driven ontologies that are relevant, adaptable to change and therefore more likely to be used. In addition, the agile method provides objective, documented evidence that an ontology meets a users needs. Requirements are more than capturing the entities and their relationships that exist in a given domain, since for most domains, this is not feasible and can introduce ontologies that are unnecessarily complex. The features of interest should be prioritised with the users needs in mind, but also with the developing team in mind, since person power and money are limited; setting out to represent all of reality is infeasible in most cases and often unnecessary. Our experience from SWO is that an agile approach to ontology authoring could deliver what a biomedical community needs by making ontology authoring agiley responsive to users.

6 ACKNOWLEDGEMENTS

This work was funded by JISC. We would like to thank all the participants who gave their time to the SWO project.

REFERENCES

Alexander, P., Nyulas, C., Tudorache, T., Whetzel, P., Noy, N., and Musen, M. (2011). Semantic infrastructure to enable collaboration in ontology development. In *2011 International Conference on Collaboration Technologies and Systems (CTS)*, pages 423–430.

Auer, S. and Herre, H. (2007). Rapidowl an agile knowledge engineering methodology. In I. Virbitskaite and A. Voronkov, editors, *Perspectives of Systems Informatics*, volume 4378 of *Lecture Notes in Computer Science*, pages 424–430. Springer Berlin / Heidelberg.

Brown, A. (2011). An overview of swo. http://softwareontology.wordpress.com/2011/02/23/an-overview-of-sword/.

Garcia, A., ONeill, K., Garcia, L. J., Lord, P., Stevens, R., Corcho, O., and Gibson, F. (2010). Developing ontologies within decentralised settings. In H. Chen, Y. Wang, and K.-H. Cheung, editors, *Semantic e-Science*, volume 11 of *Annals of Information Systems*, pages 99–139. Springer US.

Grenning, J. (2002). Planning poker or how to avoid analysis paralysis while release planning.

Kirk, G. (2011). Democracy unleashed: Bringing agility to citizen engagement. In *AGILE Conference 2011*, pages 209–215.

Martin, R. (2002). *Agile Software Development, Principles, Patterns and Practices*. Prentice Hall.

Noy, N. F., Shah, N. H., Whetzel, P. L., Dai, B., Dorf, M., Griffith, N., Jonquet, C., Rubin, D. L., Storey, M.-A., Chute, C. G., and Musen, M. A. (2009). BioPortal: ontologies and integrated data resources at the click of a mouse. *Nucleic Acids Research*, **37**(suppl 2), W170–W173.

Randall, D., Procter, R., Lin, Y., Poschen, M., Sharrock, W., and Stevens, R. (2011). Distributed ontology building as practical work. *Int. J. Hum.-Comput. Stud.*, **69**, 220–233.

Rector, A. L. (2003). Modularisation of domain ontologies implemented in description logics and related formalisms including owl. In *Proceedings of the 2nd International Conference on Knowledge Capture*.

Simperl, E., Machol, M., and Burger, T. (2010). Achieving maturity: the state of practices in ontology engineering in 2009. *Int Journal of Computer Science and Applications*, **7**, 4565.

Tempich, C., Pinto, H. S., and Staab, S. (2006). Ontology engineering revisited: an iterative case study with diligent. In *In Proc. of the 3rd European Semantic Web Conference (ESWC 2006*, pages 110–124.

The Gene Ontology Consortium (2010). The Gene Ontology in 2010: extensions and refinements. *Nucleic Acids Research*, **38**, D331–D335.

Tudorache, T., Vendetti, J., and Noy, N. (2008). Web-Proíegé: A Lightweight OWL Ontology Editor for the Web. *Fifth OWLED Workshop on OWL: Experiences and Directions*.