Test-Driven Development for Ontology Engineering

Literature Review

Kieren Davies
University of Cape Town
kieren@kdavi.es

ABSTRACT

Ontologies are a subject of active research, yet they see little adoption in business and industry. A contributing factor is that ontology engineering methodologies have not reached a state of maturity. Test-Driven Development (TDD) is a software engineering methodology with proven success, in which new code is written only when an automated test fails. We propose that TDD should be incorporated into ontology engineering methodologies. In this paper we review prior work on methodologies, testing ontologies, and applying TDD to ontologies. We identify that much future work is needed, particularly in analysing performance of different test implementations, and formalising a new methodology and evaluating its effectiveness.

CCS Concepts

•**Computing methodologies** → **Ontology engineering;** •**Software and its engineering** → *Software development methods; Software testing and debugging;*

Keywords

Ontology engineering, test-driven development, development methodology

1. INTRODUCTION

Ontologies, and ontology engineering, have become increasingly relevant in the past decade. They are regarded as a critical component of the Semantic Web [2], and have been employed successfully in fields ranging from genetics [10] to news and broadcasting [20].

The value of ontologies is seen by considering their use cases, which generally fall into three categories [24]:

- Communication. An ontology serves as a shared understanding of some system, be it software or a domain, with assumptions made explicit. This allows different people in the same organisation, or working on the same project, to better understand each other and make more consistent decisions. The field of enterprise modelling concerns itself with this problem, but the tools and techniques currently in use have known shortcomings which are addressed by ontologies [13].

- Interoperability. Ontologies can be used as a translation layer which allows different pieces of software to integrate with each other. This is especially relevant because businesses often use software from different vendors who do not agree on terminology and do not use any common protocol or API. They can also be applied to data in heterogeneous representations to achieve a unified view.

- Systems engineering. When building a new system or application, an ontology can serve as a specification, which can aid in requirements gathering or even declaratively specify the design of the system. An ontology which describes an application can be used to manually or automatically find internal inconsistencies, thereby improving reliability. It can also be applied to ensure data integrity, in the application or between it and any data sources.

Despite this, they have so far seen disappointingly little adoption within business and industry [5, 13]. This is clearly a problem with complex causes, but in this paper we will focus on one contributing factor: a lack of mature engineering practices.

In Section 2 we examine the methodologies popularly used to author ontologies, and where they fall short. In Section 3 we discuss the software engineering practice of test-driven development and how it applies to ontology engineering. In Section 4 we discuss the established practice of competency-question-driven ontology authoring, and how it can be combined with test-driven development. Finally we conclude in Section 5 with a summary of areas identified for future work.

2. ONTOLOGY ENGINEERING METHODOLOGIES

Adoption of methodologies has been crucial to the commercial success of software engineering. As understanding of the software development process has improved, methodologies have been established to address many known of the known risks. This has improved software reliability and allowed the scope and scale of software projects to grow enormously.

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

Just like software engineering, ontology engineering is a difficult and cumbersome process. It seems natural that some approaches to problems should be applicable to both domains, and so methodologies have been designed for and applied to ontology engineering.

In ontology engineering, a methodology typically describes

- a set of tools and techniques to use,
- stages in which they should be applied, and
- goals which should be achieved at each stage.

The earliest fully-fledged methodologies arose out of the Toronto Virtual Enterprise (TOVE) Project [11] and the Enterprise Project [25], based on the personal experience of the creators. They were shortly followed by many others, notably including METHONTOLOGY [8] which remains popular nearly twenty years later. A recent analysis and comparison of methodologies [12] has identified a total of fifteen which are currently in use, excluding the NeOn methodology framework [23].

According to the aforementioned analysis, none of the existing methodologies are mature. That is, each of them is in some way inadequate for widespread adoption. METHONTOLOGY is the only one that provides sufficient details of the techniques involved for a user to clearly understand and apply them, but it particularly falls short by not accounting for collaboration. The Ontolingua Server [7], for example, supports collaboration by providing an online development environment which allows an ontology to be modified by multiple users simultaneously, and also by providing a repository of ontologies for reuse.

It is clear that the shortcomings of current methodologies need to be addressed, and this needs to be supported by new tools and techniques. We propose that one aspect that should be integrated is Test-Driven Development, introduced in Section 3.

2.1 User Testing

The use of methodologies is generally accepted as a good idea, but no work has been done on verifying their effectiveness with user tests.

Related testing is scarce in the literature. One experiment was done on GINO [3], a system for using natural language to modify or query an ontology. The focus was usability of the interface, so it timed users as they completed predetermined tasks and afterwards performed the SUS usability test [4]. When considering methodologies, we are primarily interested in the quality of the resultant ontology, so it is not useful to apply these same methods—at least not until the effect of methodologies on quality is understood.

Another experiment was done on the effectiveness of foundational ontologies [14]. Participants were gathered from a series of courses on introductory ontology engineering, totalling 52 postgraduate students, lecturers, and researchers. They were instructed on the purpose and usage of foundational ontologies, then split into 18 groups and tasked with developing a new ontology in the span of 24 hours, with the choice of using a foundational ontology or not. Comparison was done quantitatively by counting the number of entities and classes added, and qualitatively by manually inspecting for certain patterns of mistake.

This kind of experiment can be simply adapted to test methodologies. An effective methodology should improve efficiency and reduce mistakes, so the same measurements are applicable. A suitable experiment would be structured as follows:

1. Divide participants into test groups for each methodology under study, and a control group.
2. Train the test groups on the importance and usage of the assigned methodology.
3. Instruct every participant:
   (a) Develop a new ontology in a specified domain.
   (b) Submit it at the end of a specified time frame.
   (c) Make use of provided knowledge sources for elicitation, if appropriate.
4. Evaluate the ontologies by counting the number of classes, properties, and axioms created, and inspecting for identifiable mistakes.

The biggest concern is the difficulty of finding participants. The experiment failed to achieve statistical significance with four out of its five variables, so perhaps many more than 50 participants are required. They must also all be of a novice level so that members of the control group do not inadvertently apply any methodologies.

The domain in which participants are instructed to build an ontology must be chosen carefully. If it is very familiar to many or all participants, they will likely consider themselves domain experts and ignore knowledge elicitation components of the methodologies. If it is too foreign and difficult to understand, too much time may be lost trying to understand the domain instead of applying methodological techniques and building the ontology.

Despite the difficulties, this is a valuable avenue for future work.

3. TEST-DRIVEN DEVELOPMENT

Test-Driven Development (TDD) is a software development methodology which was popularised by Kent Beck in 2003 [1], although it can be traced in some form back to at least 1957 [17, p. 159–160]. Its essence is two rules:

- Write new code only if an automated test has failed.
- Eliminate duplication.

This induces a "red-green-refactor" pattern of development: first write a new test which fails, then write code which makes it pass with minimal effort, then remove resultant duplication and restructure if necessary. The process is usually facilitated with a test harness which runs tests automatically and generates reports.

The rationale behind TDD is primarily that it engenders trust in the quality of the software. Observing that the entire test suite has passed provides assurance that the software does what it should, ostensibly without defects. The hope is that this results in increased programmer morale, less time spent correcting defects, and therefore increased productivity and code quality.

A meta-analysis has shown that TDD improves code quality, especially in more complex projects, but does not significantly affect productivity [19].

A related methodology is Behaviour-Driven Development (BDD) [6], which differs principally in what constitutes a test. It permits individual tests which cover many components of the system, thereby testing "behaviours" rather than the conventional small "units". It also advocates that they be given natural-language names which describe the behaviour they test. Lastly, it suggests that any gathered requirements be directly recorded as tests.

3.1 Ontology Unit Testing

An ontology is a "white box"—that is, all of its internals are visible—so it may seem odd to employ automated tests at all. However there are cases where an author, especially an experienced one, may easily make a mistake without noticing. Consider an author who creates the following classes:

$$\texttt{Giraffe} \sqsubseteq \texttt{Herbivore} \sqsubseteq \texttt{Mammal} \sqsubseteq \texttt{Animal}$$

The author then realises that not all herbivores are mammals, so changes `Herbivore` to be a subclass only of `Animal`. But now `Giraffe` is no longer a subclass of `Mammal`, and an application which uses this ontology to enumerate mammals would erroneously miss giraffe. This mistake could be caught by a suitable test.

From this follows another question: why add a test which asserts an inferred axiom, instead of just adding that axiom directly to the ontology? Doing so introduces redundancy, making modification of the ontology more difficult, and in some circumstances increases the complexity of reasoning [26].

Very little work has been done on unit testing ontologies. Vrandečić and Gangemi [26] propose it be accomplished by accompanying the ontology under test with two others, the "positive" and "negative" test ontologies. The positive test ontology should contain axioms which must be derivable from the ontology under test, and the negative should contain axioms which must not be. There is no discussion of how such a mechanism might be implemented.

Warrender and Lord [28] present a tool called Tawny-OWL. It uses the OWL API to provide tests which can query the TBox, either directly or with a reasoner, and make changes to the ontology before querying and revert them afterwards. It can only test knowledge in the TBox. It has the disadvantage of requiring tests to be written in Clojure, a programming language with little adoption.

Ławrynowicz and Keet [16] present TDDOnto, a plugin for Protégé with three kinds of tests:

- TBox query with OWL-BGP through SPARQL,
- TBox query with OWL API, and
- ABox query with "mock individuals" with OWL API.

OWL-BGP relies on HermiT, and the other two on any standard reasoner. The paper notes the possibility of TBox queries with SPARQL-DL. A performance analysis was done but results are not clear. Each of the types of test performed well in different circumstances.

The following areas are open to future work:

- Alternative test implementations.
- Thorough comparison of performance between existing implementations.
- Implementation of non-functional tests, such as reasoning time.
- Assessment and improvement of interfaces.

3.2 Applying TDD to Ontology Engineering

Since there is so little literature on testing ontologies, it is not surprising that there is even less on applying TDD—we could find only one paper which directly addresses the matter.

Keet and Ławrynowicz [15] investigate how TDD can be transferred from software to ontology engineering, and suggest a sequence of high-level steps which should form part of a methodology. The methodology is described in very little detail, so considerable refinement is necessary before this produces a methodology which can be considered mature.

A new tool in development, named Scone [18, 22], provides a framework and accompanying methodology for BDD of ontologies using natural-language specifications. Again, the methodology is not described in sufficient detail.

As of yet, there is no accepted set of tools for testing ontologies, which is a major obstacle to promotion of TDD.

Future work is needed in the following areas:

- Implementation and assessment of a full toolchain to support TDD.
- Formalisation of a methodology which incorporates TDD.
- Evaluation of such a methodology with user tests.

4. COMPETENCY-QUESTION-DRIVEN AUTHORING

A Competency Question (CQ) is a natural-language question which represents a query, or set of queries, an ontology is expected to be able to answer [11]. CQ-Driven Authoring (CQDA) is a proposed methodology in which CQs are identified and used to produce tests, ideally automatically [21]. Knowledge should then only be added to the ontology in order to make a test pass. The methodology has not been fully developed and formalised.

The chief difficulty in this is automatically translating a natural-language CQ into a SPARQL query. Some work has been done on this [9, 27], but there are many competing technologies which should be assessed for their suitability.

Another problem is the constraint on the kind of test which can be automatically produced from a CQ. Such a test cannot, for example, assert that a query answer contains a given element or subset.

There is plenty of room for future work on this topic:

- Formalisation and evaluation of a CQ-driven methodology.
- Evaluation of technologies for translating natural language to SPARQL.
- Development of a framework similar to Tawny-OWL or Scone which allows CQs to be combined with more expressive assertions.

5. CONCLUSION

Test-Driven Development seems to be a promising solution to the immaturity of ontology engineering methodologies. We have identified future work which should be done in assessment of ontology engineering methodologies, unit testing, and application of TDD. We believe the most pressing need is for a mature TDD toolchain which can be easily adopted by many users, and a full formalisation of a methodology based on TDD.

6. REFERENCES

[1] K. Beck. *Test-Driven Development: By Example*. Addison-Wesley, 2003.

[2] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 284(5):34–43, 2001.

[3] A. Bernstein and E. Kaufmann. GINO—a guided input natural language ontology editor. In *The Semantic Web - ISWC 2006*, volume 4273 of *Lecture Notes in Computer Science*, pages 144–157. Springer, 2006.

[4] J. Brooke. SUS: A 'quick and dirty' usability scale. In *Usability Evaluation in Industry*, pages 189–194. Taylor & Francis, 1996.

[5] J. Cardoso. The semantic web vision: Where are we? *IEEE Intelligent Systems*, 22(5):84–88, 2007.

[6] D. Chelimsky, D. Astels, B. Helmkamp, D. North, Z. Dennis, and A. Hellesoy. *The RSpec Book: Behaviour Driven Development with RSpec, Cucumber, and Friends*, chapter Behaviour-Driven Development. Pragmatic Bookshelf, 2010.

[7] A. Farquhar, R. Fikes, and J. Rice. The Ontolingua Server: a tool for collaborative ontology construction. *International Journal of Human-Computer Studies*, 46(6):707–727, 1997.

[8] M. Fernández, A. Gómez-Pérez, and N. Juristo. Methontology: From ontological art towards ontological engineering. In *Ontological Engineering*, Spring Symposium Series. Association for the Advancement of Artificial Intelligence, 1997.

[9] S. Ferré. Squall: A controlled natural language as expressive as sparql 1.1. In *Natural Language Processing and Information Systems*, volume 7934 of *Lecture Notes in Computer Science*, pages 114–125. Springer, 2013.

[10] Gene Ontology Consortium. Gene ontology consortium: going forward. *Nucleic Acids Research*, 43(D1):D1049–D1056, 2015.

[11] M. Grüninger and M. S. Fox. Methodology for the design and evaluation of ontologies. In *Workshop on Basic Ontological Issues in Knowledge Sharing*, International Joint Conference on Artificial Intelligence, 1995.

[12] R. Iqbal, M. A. A. Murad, A. Mustapha, and N. M. Sharef. An analysis of ontology engineering methodologies: A literature review. *Research Journal of Applied Sciences, Engineering and Technology*, 6(16):2993–3000, 2013.

[13] M. Kaczmarek. Ontologies in the realm of enterprise modeling—a reality check. In *Formal Ontologies Meet Industry*, volume 225 of *Lecture Notes in Business Information Processing*, pages 39–50. Springer International Publishing, 2015.

[14] C. M. Keet. The use of foundational ontologies in ontology development: An empirical assessment. In *The Semantic Web: Research and Applications*, volume 6643 of *Lecture Notes in Computer Science*, pages 321–335. Springer, 2011.

[15] C. M. Keet and A. Ławrynowicz. Test-driven development of ontologies. In *13th Extended Semantic Web Conference*, Lecture Notes in Computer Science. Springer, 2016.

[16] A. Ławrynowicz and C. M. Keet. The TDDonto tool for test-driven development of DL knowledge bases. In *29th International Workshop on Description Logics*, number 1577 in CEUR Workshop Proceedings, 2016.

[17] D. D. McCracken. *Digital Computer Programming*. John Wiley & Sons, 1957.

[18] F. Neuhaus. Scenario-based ontology evaluation (SCONE) user guide. https://bitbucket.org/malefort/scone/raw/default/documentation/SCONEUserGuide.pdf, 2015. Accessed 9 May 2016.

[19] Y. Rafique and V. B. Mišić. The effects of test-driven development on external quality and productivity: A meta-analysis. *IEEE Transactions on Software Engineering*, 39(6):835–856, 2013.

[20] D. Ramsden. What did linked data ever do for us anyway? http://www.bbc.co.uk/academy/technology/software-engineering/semantic-web/article/art20130720153136618, 2013. Accessed 4 May 2016.

[21] Y. Ren, A. Parvizi, C. Mellish, J. Z. Pan, K. van Deemter, and R. Stevens. Towards competency question-driven ontology authoring. In *The Semantic Web: Trends and Challenges*, volume 8465 of *Lecture Notes in Computer Science*, pages 752–767. Springer, 2014.

[22] Scone project. https://bitbucket.org/malefort/scone. Accessed 9 May 2016.

[23] M. C. Suárez-Figueroa, A. Gómez-Pérez, and M. Fernández-López. The NeOn methodology for ontology engineering. In *Ontology Engineering in a Networked World*, pages 9–34. Springer, 2012.

[24] M. Uschold and M. Gruninger. Ontologies: Principles, methods and applications. *The Knowledge Engineering Review*, 11:93–136, 1996.

[25] M. Uschold and M. King. Towards a methodology for building ontologies. In *Workshop on Basic Ontological Issues in Knowledge Sharing*, International Joint Conference on Artificial Intelligence, 1995.

[26] D. Vrandečić and A. Gangemi. Unit tests for ontologies. In *On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops*, volume 4278 of *Lecture Notes in Computer Science*, pages 1012–1020. Springer, 2006.

[27] C. Wang, M. Xiong, Q. Zhou, and Y. Yu. PANTO: A portable natural language interface to ontologies. In *The Semantic Web: Research and Applications*, volume 4519 of *Lecture Notes in Computer Science*, pages 473–487. Springer, 2007.

[28] J. D. Warrender and P. Lord. How, what and why to test an ontology. In *Bio-Ontologies 2015*, 2015.