# In-house Developed Tools for Ontology Engineering Education (Demo)

C. Maria Keet[1]

[1]*Department of Computer Science, University of Cape Town, 18 University Avenue, Rondebosch 7701, Cape Town, South Africa*

**Abstract**

Both good ontology development and ontology engineering are considered to be advanced topics in computing curricula. They draw on knowledge from different specialisations and need to strike a balance between theory and skills. This may cause a steep learning curve, but one where tools may assist with the learning process. This work-in-progress demo paper, presents the ongoing development of a set of in-house developed tools that assist in the various learning activities in different ways. They were motivated in part by the ontology engineering course taught by the author and cover a spectrum of tasks for learning about ontology development.

**Keywords**

Ontology Development Tools, Ontology Engineering, Education

## 1. Introduction

University-level courses on ontologies may have various aims, depending on the students' background and the degree programme it is part of. This may focus on: domain ontology development only; also foundations for ontology research that may cover processes of development as well; logics and languages; or the development of ontology-driven information systems. An example of streaming of prospective types of learners is the online OBOOK for OBO Foundry ontologies that is currently under development [1]. Most such courses are advanced topics in computing curricula, building upon undergraduate content and drawing in concepts from different specialisations. They operate at levels 4 and 5 in Bloom's taxonomy and need to strike a balance between new theory and new skills to apply the theory. This may cause a steep learning curve in general and even more so if it relies solely on scientific articles. For instance, students need to learn about Competency Questions (CQs) for ontologies, which were first proposed in 1996 [2], yet guidance for writing appropriate CQs was proposed only in 2019 [3]. Assistive tooling is a well-known educational approach to try to lessen the difficulties of learning content [4] or to turn passive knowledge transfer into active learning. It does raise the question: once determined what they should learn, then where are pain points where tools may assist, rather than distract from, the task?

This is also the case for the ontology engineering course at the University of Cape Town,

*FOIS'23 Demonstrations: July 17-20, 2023, Sherbrooke, Canada*

mkeet@cs.uct.ac.za (C. M. Keet)

0000-0002-8281-0853 (C. M. Keet)

where the author teaches ontology engineering. Besides using so-called 'dual use' tools—for regular use and for teaching—I am, and have been, involved in developing tools that are either explicitly intended for education or sufficiently dual use to be used educationally. In this work-in-progress demo paper, I report on the ongoing development of a set of in-house developed tools that assist in the learning processes in various ways. They are aimed at honours (4th-years) and postgraduate-level course on ontology engineering for computer science students. The tools are available through the author's ontology engineering textbook's website[1], including listing supporting material, such as descriptions, tutorials, screencasts, and code.

In the remainder of this paper, I first describe several design considerations for the creation of a 'toolbox' (Section 2) and subsequently summarise the current set of in-house developed tools (Section 3). It closes with concluding remarks in Section 4.

## 2. Toolbox design considerations

Educational tool selection and creation for an ontology engineering course depend in part on the content that will be covered and the tasks that a lecturer expects a student to be able to complete by the end of the course. The current state of tooling for ontology education is not as mature to already have a complete workflow to map to learning outcomes (see, e.g., [5]), although learning outcomes for the author's course have been defined (see page xi of [6]). Other typical considerations include the detection of difficult concepts to grasp or skills to master that students grapple with and where software support may alleviate or resolve the bottleneck. Within the broader context of transformation of education in the world, including in South Africa, there are calls for greater student involvement in course content design and to have their voices heard [7, 8]. I will elaborate on these topics in the remainder of this section.

### 2.1. Course topics and transitions

The content of the open textbook [6] for the course is divided into three parts. Block I consists of logic foundations, covering First Order Predicate Logic, Description Logics, OWL, and automated reasoning. Block II focuses on developing good ontologies with methods and methodologies, the top-down approach with foundational ontologies and mereology, and bottom-up approaches on extracting content from non-ontological resources. Block III consists of a selection of advanced topics, being Ontology-Based Data Access, the interaction between ontologies and natural languages, modularisation of ontologies, and advanced modelling with more language features, of which one is chosen each year in the course I teach. There are exercises, an assignment on domain ontology development and a mini-project. A list of requirements emanating from the ordering of the topics and expected learning outcomes and observation of over 10 years of teaching the course (with some variations in blocks 2 and 3), there are multiple options:

1. Logic: autocomplete function and a syntax checker, a model builder to test a theory, and interactive tableau.
2. Transitioning from one logic to another and from theory to praxis. Specific learning hurdles and (perceived to be) 'annoyances' include:

[1]The tools are listed with further information at https://people.cs.uct.ac.za/~mkeet/OEbook/OEsoftware.html.

    - The DL-to-OWL notation mappings and the DL or OWL to Ontology Development Editor (ODE, e.g., Protégé) keywords, such as from the "∃" to "`ObjectSomeValuesFrom`" to "some".
    - Getting to grips with the many OWL 'species', which species has which features, and which species an ontology is in and why not another.
    - Why the reasoner outputs what it does, i.e., explanations of the deductions.
3. Tool support for methodologies. Since 'information systems style' methodologies (like waterfall, agile) adapted to ontology development is not a favourite topic outside information systems, software-supported workflows may assist to get students to follow a procedure. Anecdotally, NeON [9] is perceived to have too many routes, yet Methontology [10] has too few steps. There is no solution yet; for the duration, this is divided into:
    - Tools for component tasks of the procedure. Among others, how to devise good competency questions, adding axioms in a guided way, ontology alignment, modularisation, etc.
    - Tools for quality assessments. They can take the form of metrics, of canned explanations for certain axiom types or missing content (e.g., missing annotations, orphan classes), or ontological reasoning services or consistent use of a modelling style.
4. Foundational ontologies. Based on the core challenge of understanding what the ontology's content means, there are two key learning hurdles:
    - Selection of a foundational ontology among the set of ontologies available.
    - Alignment of a foundational ontology to one's domain ontology, which also needs guidance at least for computer science students.
5. Bottom-up strategies, for which many techniques exist, including from spreadsheets, conceptual data models, thesauri, and ontology learning from text.
    - It needs to overcome a learning hurdle of grasping the subtle differences between ontologies and similar-looking artefacts that are non-ontological resources.
    - Support for the task, guiding the process on the differences and adding explanations.
6. Auxiliary assistive tooling for ontology development, such a natural language renderings of the axioms in one's language of choice and diagrammatic renderings.
7. Auxiliary assistive tooling and related specifications and guidance for ontology tool development, for both ontology engineering and ontology-driven information systems.

## 2.2. Students' vantage point

Recent years has seen a theoretical shift in pedagogies from the 'sage on the stage' to other ways and modes of learning, including active learning, co-creation of course content, and students as constructors of knowledge jointly with the lecturer [7, 8]. For the course set-up, potential gaps and hurdles in the learning process and co-creation may be observed in the mini-project that is conducted in teams of 2-4 students. Typically 15-20 topics are proposed and students can propose one. They may be theoretical (e.g., a literature review, mapping to another language), experimental (e.g., comparing reasoner performance), or hands-on (developing a tool, setting up an OBDA system). What students do is, in part, motivated by what they wanted there to be. Examining 10 years of projects of the semester/term-based course offerings, there were 60 projects of which 17 involved tool development, which is summarised in Table 1. What

**Table 1**
10 years of mini-projects of the ontology engineering course at Honours (4th year) and Masters level, when ran over a term (5-6 weeks) or semester (12 weeks).

| Year | No. of projects | Tools | Tool topics |
|---|---|---|---|
| 2011 | 7 | 3 | UML to OWL; OWL profile checker; verbalizer (English) |
| 2012 | 9 | 3 | Verbalizer (isiZulu); ontology quality; OWL profile checker |
| 2013 | 4 | 1 | Verbalizer (isiZulu); (OWL species selection on paper) |
| 2014 | 6 | 1 | OWL profile recommender |
| 2015 | 6 | 3 | Verbalizer (Norwegian, isiZulu, Sepedi) |
| 2016 | 10 | 2 | Verbalizer (multilingual [English, Setswana, Afrikaans]), OWL 2 classifier |
| 2017 | 11 | 3 | Verbalizer (simplified English), Manchester Syntax/Protégé localization (Afrikaans), DL axiom renderer plugin |
| 2019 | 3 | 0 | |
| 2020 | 1 | 1 | Verbalizer (multilingual [Afrikaans and ciShona]) |
| 2021 | 3 | 0 | (BFO classifier on paper) |

stands out are the new tools for coming to grips with the many OWL species early on and the verbalisers to ease access to the logic later on. There were also a number of projects on ontology quality as experiment, several comparisons of visualisers, a few comparing OntoClean implementations, and a few OBDA instantiations.

## 3. A set of tools

To assist with learning about ontology engineering, we have been working towards a so-called 'toolbox' of software tools to augment the theory, both in the sense of illustration of the theory as well as assistance in putting it into practice, in order to support the specified aims and learning objectives. Besides availing of extant auxiliary tools developed elsewhere, such as Protégé as ODE [11] that also computes explanations [12] and OOPS! for basic quality checking [13], we're working on a set of in-house developed tools towards filling various gaps, and will continue to do so. Some of them were developed as part of the mini-projects and that students in later cohorts found useful, which has served as an impetus for students in successive years to also make a useful tool that may be listed on the textbook website. This section reports on the current state of the tools. It is divided into three main categories: logics support, methods support, and natural language tools support.

### 3.1. Logics support

Since there are well-known tools for First Order Predicate Logic (e.g., Tarski's World [14]) and interactive tableau basics (e.g., the Treeproof generator[2]), and it is not the the core of the course, the focus has been on DLs and OWL. The tool that is chronologically used first in the course is the DL axiom renderer (a Protégé 5.5 plugin), which renders class expressions in DL notation

[2]https://www.umsu.de/trees/

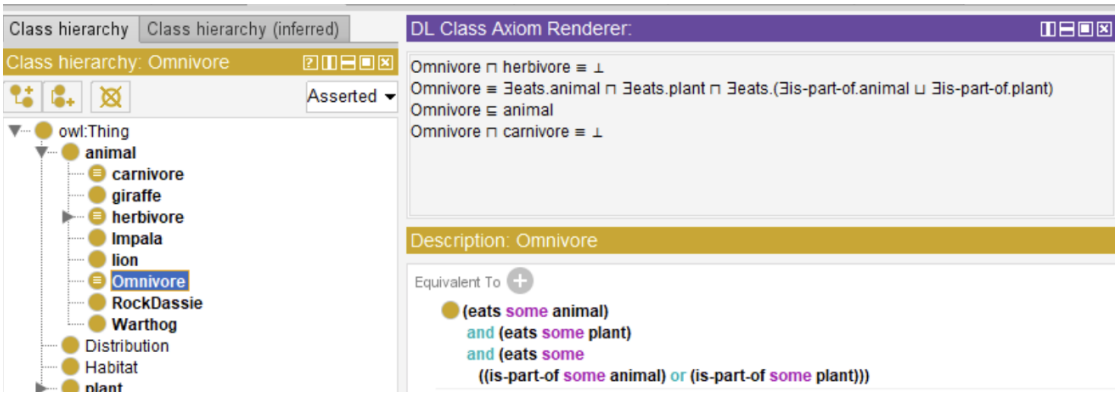


**Figure 1:** Screenshot of the DL axiom renderer. Bottom: the Protégé class expression pane where content can be edited; Top: the axioms rendered in DL notation.

next to the Protégé notation, as illustrated in Figure 1. This assist with transitioning between the theory (logic notation in DL) to the tool's rendering in praxis. Once that transition is made, it is generally not used further.

The second, and continuously most popular, tool is the OWL Classifier standalone tool (jar file). It determines the OWL species of the ontology and which axioms violate the other species or profiles, if any. It extracts and lists those axioms for the user. This also functions as a way to learn what is, and is not, in a particular OWL species. It incorporates two OWL APIs, so that it can distinguish between the three OWL species and all five OWL 2 species. It also lists the DL fragment by their letters, with a brief explanation. A screenshot is included in Figure 2.

The OWL Classifier is meant as an educational tool, but practically has gained a 'dual use' function, because it also can aid the entry-level ontology developer and experienced developer when a project imposes an expressiveness limitation on the ontology and one needs to pinpoint and negotiate among profile-violating axioms [15].

## 3.2. Tools for methods for ontology authoring

We have developed several tools for methods for ontology authoring, some of which have found their way into educational use, and two for education that can be of general use.

**Teaching-focussed, but expected to be dual use** The main teaching-focussed tool in this category is the BFO Classifier (standalone jar file) to align an ontology to BFO v2.0 [16]. It was developed partly because BFO is used most often among the top-level ontologies and partly because the D3 for DOLCE as MoKI plugin [17] is defunct since MoKI is. The questions in the decision diagram and the examples provided present the new terms that the students have to learn about in a different, more accessible way. As it is a new extension of the toolset, its impact on the students is yet to be determined. In addition, since practitioners face problems aligning their ontology to BFO [18], the tool also has the potential to become part of regular tool use among ontology developers.

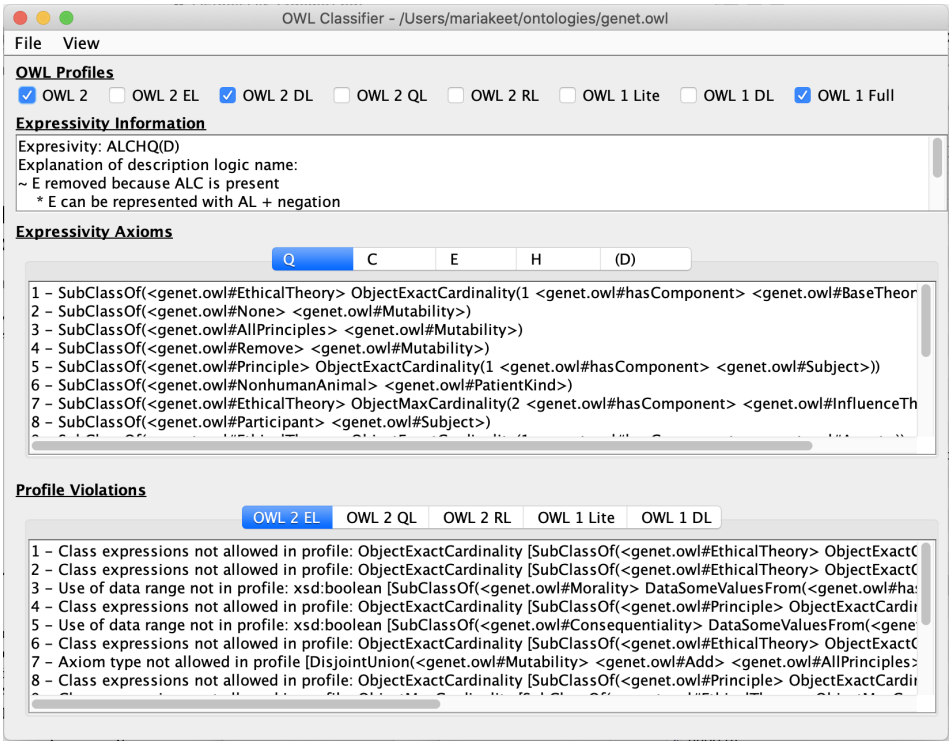


**Figure 2:** Screenshot of the OWL Classifier. Top: the OWL language(s) and the DL that the ontology is expressed in; middle: which axioms are responsible for it; bottom: why it is not another OWL species.

**Entry-level research tools in teaching** The following four tools were developed to solve a research problem in ontology development. This may be traced back to them not having been part of the educational canon until recently, and thus filling a gap in skills development. They include support for authoring competency questions with CLaRO (standalone jar file) [3], such that it steers them away from question formulations that are inapplicable, the foundational ontology selection tool ONSET (standalone jar file) [19], OntoPartS (standalone .exe) to select the appropriate part-whole relation [20], and the TDDonto2 Protégé 5.5 plugin as axiom authoring support in Protégé [21]. ONSET also provides one-liner explanations of terms from ontology and it provides an explanation of its output on why one foundational ontology was chosen over another, or why it is a draw due to too many conflicting requirements.

**Advanced topics possible 'dual use' tools** These tools are useful for the Block III extra topics of the course and for some mini-projects. They include the web-based ROMULUS foundational ontology repository with detailed comparisons and alignments of the selected foundational ontologies [22], and two tools related to the modularisation chapter of v1.5 of the textbook, which is taught only every few years, being TOMM [23] to compute module metrics and NOMSA [24] to modularise an ontology (both standalone jar files).

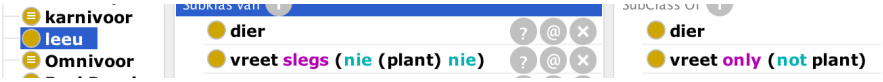


**Figure 3:** An attempt at double negation (*nie ... nie*) with a fixed string, for the *leeu* 'lion' that does not eat plants with the Afrikaans plugin in the centre and the default Protégé rendering on the right.

### 3.3. Natural language tools

Generally, tools involving natural language are limited despite the continuous interest of students, which is largely because it looks easy but is hard to do well, and even more so for Southern African indigenous languages due to a range of non-trivial grammatical features. Our current tools are relevant specifically to Chapter 9 in Block III of the textbook and focus on introducing and illustrating issues on the interaction of ontologies with natural languages.

**Teaching-focussed, but may be dual use** The changes in Protégé from v3.x to v4.x and later have not all been perceived to be good. In particular, v3.x had ample use of DL notation whereas this changed in later versions to *ad hoc* and untested new keywords in English. This frustrates rendering of the axioms for anyone who is developing an ontology in a language other than English; e.g., "jirafa come only (oja or ramita)" ('giraffe eats only (leaf or twig)') in Spanish-English code switching in the class expression editor is not conducive to ontology authoring. This is addressed by the DL axiom renderer and by a Protégé 5.x localisation plugin. A set of keywords were translated and a plugin developed to render class expressions in the selected language. It was done for Afrikaans and Spanish, but one may proceed likewise with another language: create a new XML file with translations and recompile the plugin. It also brings afore additional limitations of Protégé, such as multi-word expressions, the double negation in Afrikaans that it was initially not equipped for (see also Figure 3), and that not all terms have a 1:1 fixed-word translation.

**The 'dual use' tools for advanced topics** Currently, there are two dual use tools to demonstrate both advanced topics in knowledge-based natural language generation and a broader range of techniques for tool development. First, a template-based approach for ontology verbalisation is pitted against a grammar-based approach with Grammatical Framework with the Afrikaans verbaliser [25]—the verbalisers listed in Table 1 are template-based augmented with grammar rules. Second, while java and the OWL API [26] are well known, an alternate programming ecosystem in Python exists with Owlready [27] that has been shown to work for the isiZulu verbaliser [28], which has led to better design considerations in the mini-projects.

## 4. Closing remarks

The paper presented a set of in-house developed tools that assist with ontology engineering education activities. As the field progresses, so will any set of tools.

Tool development choices face a trade-off between ease of use, such as a Protégé plugin being easier when one already uses that ODE, versus broader usability beyond that specific

infrastructure, for which there is no good solution other than duplicating the work. Moreover, all tools, be they for teaching or research and in-house developed or developed elsewhere, are forced to having to grapple with the 'forced obsolescence' when dependencies—be it the ODE, the JRE or JDK, Maven, Python, the operating system, APIs—update their version and discontinue the version the tool was developed for rather than offering backward compatibility. The lack of guarantee that all tools still work also next year holds for almost all freely available ontology engineering tools, and the ones presented in this paper are no exception to that.

## Acknowledgments

The author is grateful for the time, effort, and expertise of the collaborators, scientific programmers, and student-programmers having designed and/or contributed to the tools mentioned in the paper: Michael Harrison and Larry Liu (DL axiom render), Brian McGeorge, Aashiq Parker, and Muhummad Patel (OWL Classifier), Kieren Davies and Agnieszka Lawrynowicz (TDDOnto), Zola Mahlaza (CLaRO), Zubeida Khan (ONSET, ROMULUS, TOMM, NOMSA), Chiadika Emeruem and Steve Wang and Zubeida Khan and Toky Raboanary (BFO Classifier), Francis Fernández, and Annette Morales (OntoPartS), Musa Xakaza and Langa Khumalo (isiZulu verbaliser), Lauren Sanby and Ion Todd (Afrikaans verbaliser), and Adam Kaliski, Casey O'Donnell, Michael Harrison, and Toky Raboanary, and Tommie Meyer and Pablo Fillottrani (ODE localiser). The DOT4D project funded in part the development or improvement of the DL axiom renderer, the BFO Classifier, the CLaRO tool, and the ODE localiser as part of the Ontology Engineering open textbook grant. The isiZulu verbaliser was funded in part, and based on research supported in part, by the National Research Foundation of South Africa (Grant Number 93397).

## References

[1] OBOOK Authors, Open biological and biomedical ontologies organized knowledge, 2022. URL: https://oboacademy.github.io/obook/.

[2] M. Uschold, M. Gruninger, Ontologies: principles, methods and applications, Knowledge Engineering Review 11 (1996) 93–136.

[3] C. M. Keet, Z. Mahlaza, M.-J. Antia, CLaRO: a controlled language for authoring competency questions, in: Proc. of MTSR'19, volume 1057 of *CCIS*, Springer, 2019, pp. 3–15.

[4] S. Stamenković, N. Jovanović, M. C. Bojan Vasović, Z. Jovanović, Software tools for learning artificial intelligence algorithms, Artificial Intelligence Reviews in print (2023).

[5] F. Berardinucci, G. Colombo, M. Lorusso, M. Manzini, W. Terkaj, M. Urgo, A learning workflow based on an integrated digital toolkit to support education in manufacturing system engineering, Journal of Manufacturing Systems 63 (2022) 411–423.

[6] C. M. Keet, An introduction to ontology engineering, College Publications, UK, 2018.

[7] E. Dunne, R. Zandstra, Students as change agents: new ways of engaging with learning and teaching in higher education, ESCalate (Education Subject Centre of the Higher Education Academy) University of Exeter, UK, 2011. URL: https://dera.ioe.ac.uk/id/eprint/14767.

[8] J. Charteris, D. Smardon, Democratic contribution or information for reform? prevailing and emerging discourses of student voice, Australian J. of Teacher Education 44 (2019).

[9] M. C. Suarez-Figueroa, G. A. de Cea, C. Buil, et al., NeOn Methodology for Building Contextualized Ontology Networks, NeOn Deliverable D5.4.1, NeOn Project, 2008.
[10] M. Fernández, A. Gómez-Pérez, A. Pazos, J. Pazos, Building a chemical ontology using METHONTOLOGY and the ontology design environment, IEEE Expert 14 (1999) 37–46.
[11] J. H. Gennari, et al., The evolution of Protégé: an environment for knowledge-based systems development, International Journal of Human-Computer Studies 58 (2003) 89–123.
[12] M. Horridge, B. Parsia, U. Sattler, Laconic and precise justifications in OWL, in: Proc. of ISWC'08, volume 5318 of *LNCS*, Springer, 2008.
[13] M. Poveda-Villalón, M. C. Suárez-Figueroa, A. Gómez-Pérez, Validating ontologies with OOPS!, in: Proc. of EKAW'12, volume 7603 of *LNAI*, Springer, 2012, pp. 267–281.
[14] J. Barwise, J. Etchemendy, The language of first-order logic, 3rd ed., Stanford, USA: CSLI Lecture Notes, 1993.
[15] C. M. Keet, R. Grütter, Toward a systematic conflict resolution framework for ontologies, Journal of Biomedical Semantics 12 (2021) 487–512.
[16] C. Emeruem, C. M. Keet, Z. C. Khan, S. Wang, BFO Classifier: Aligning domain ontologies to BFO, in: Proc. of FOUST-VI, part of JOWO'22, volume 3249 of *CEUR-WS*, 2022, p. 13p.
[17] C. M. Keet, M. T. Khan, C. Ghidini, Ontology authoring with FORZA, in: Proc. of CIKM'13, ACM proceedings, 2013, pp. 569–578.
[18] R. Stevens, P. Lord, J. Malone, N. Matentzoglu, Measuring expert performance at manually classifying domain entities under upper ontology classes, Journal of Web Semantics 57 (2019) 100469.
[19] Z. Khan, C. M. Keet, ONSET: Automated foundational ontology selection and explanation, in: Proc. of EKAW'12, volume 7603 of *LNAI*, Springer, 2012, pp. 237–251.
[20] A. Morales-González, F. C. Fernández-Reyes, C. M. Keet, Ontoparts: A tool to select part-whole relations in owl ontologies, in: The Semantic Web: ESWC 2012 Satellite Events, Springer, 2015, pp. 452–457.
[21] K. Davies, C. M. Keet, A. Lawrynowicz, TDDonto2: A test-driven development plugin for arbitrary TBox and ABox axioms, in: The Semantic Web: ESWC 2017 Satellite Events, volume 10577 of *LNCS*, Springer, 2017, pp. 120–125.
[22] Z. C. Khan, C. M. Keet, ROMULUS: a Repository of Ontologies for MULtiple USes populated with foundational ontologies, Journal on Data Semantics 5 (2016) 19–36.
[23] Z. C. Khan, C. M. Keet, Dependencies between modularity metrics towards improved modules, in: Proc. of EKAW'16, volume 10024 of *LNAI*, Springer, 2016, pp. 400–415.
[24] Z. C. Khan, C. M. Keet, NOMSA: Automated modularisation for abstraction modules, in: Proc. of EKAW'18 Posters & Demonstrations, volume 2262 of *CEUR-WS*, 2018, pp. 13–16.
[25] L. Sanby, I. Todd, C. M. Keet, Comparing the template-based approach to GF: the case of Afrikaans, in: Proc. of WebNLG'16, ACL, 2016, pp. 50–53.
[26] M. Horridge, S. Bechhofer, The OWL API: A java API for OWL ontologies, Semantic Web 2 (2011) 11–21.
[27] J.-B. Lamy, Owlready: Ontology-oriented programming in python with automatic classification and high level constructs for biomedical ontologies, Artificial Intelligence in Medicine 80 (2017) 11–28.
[28] C. M. Keet, M. Xakaza, L. Khumalo, Verbalising OWL ontologies in isiZulu with Python, in: ESWC 2017 Satellite Events, volume 10577 of *LNCS*, Springer, 2017, pp. 59–64.