Web Semantics: Science, Services and Agents on the World Wide Web 57 (2019) 100472

Contents lists available at ScienceDirect

Web Semantics: Science, Services and Agents on the World Wide Web

journal homepage: www.elsevier.com/locate/websem

Automating ontology engineering support activities with OnToology

Ahmad Alobaid [a], Daniel Garijo [b], María Poveda-Villalón [a], Idafen Santana-Perez [a,*], Alba Fernández-Izquierdo [a], Oscar Corcho [a]

[a] Ontology Engineering Group, Universidad Politécnica de Madrid, Campus de Montengancedo s/n, 28660 Boadilla del Monte, Spain
[b] Information Sciences Institute, University of Southern California, Marina del Rey, Los Angeles, 90292 California, USA

ARTICLE INFO

Article history:
Received 18 December 2017
Received in revised form 10 August 2018
Accepted 21 September 2018
Available online 9 October 2018

Keywords:
Ontology engineering
Ontology evaluation
Ontology documentation
Ontology publication

ABSTRACT

Due to the increasing uptake of semantic technologies, ontologies are now part of a good number of information systems. As a result, software development teams that have to combine ontology engineering activities with software development practices are facing several challenges, since these two areas have evolved, in general, separately. In this paper we present OnToology, an approach to manage ontology engineering support activities (i.e., documentation, evaluation, releasing and versioning). OnToology is a web-based application that builds on top of Git-based environments and integrates existing semantic web technologies. We have validated OnToology against a set of representative requirements for ontology development support activities in distributed environments, and report on a survey of the system to assess its usefulness and usability.

1. Introduction

Since the late 1990s, several ontology engineering methodologies have been proposed to transform the art of developing ontologies into an engineering activity. Methodologies such as METHONTOLOGY [1], On-To-Knowledge [2] and the NeOn Methodology [3] define clear guidelines, processes, activities and life cycles to guide ontology development.

Now that ontologies are being increasingly adopted in information systems, it is clear that ontology development tasks may also benefit from the application of common software engineering practices. Most of the ontology development support activities, such as documentation, visualization and evaluation, are usually performed individually, executing heterogeneous tools that make these activities cumbersome and time consuming. In addition, maintaining and keeping track of the generated resources for each version of an ontology has become a challenge for ontology developers.

The ontology engineering community has already shown progress towards adapting ontology development to agile software development methodologies [4–6]; as well as supporting collaborative ontology development throughout the use of common-practice software engineering tools [7,8]. In fact, it is now common among ontology developers to use Git-based environments [9] such as GitHub[1] (usual in software development) for keeping track of ontology revisions. However, existing approaches present either partial solutions ; require specialized skills that complicate their adoption (e.g., complex installation setup); or produce their outcome using idiosyncratic formats that are difficult to integrate into existing ontology development workflows.

In this paper we present OnToology, our approach towards addressing these issues. The main contributions of OnToology are: (1) automation of coarse-grained support activities involved in ontology development, including documentation, versioning, evaluation and publication of ontologies that are maintained and versioned in a Git-based environment; (2) a workflow for continuous integration of support activities when new changes in the ontology are registered ; and (3) integration of the workflow with a collaborative environment. OnToology can support individuals and teams of developers working on an ontology. While distributed collaboration to change an ontology is naturally supported by Git, the required support activities have to be integrated into Git's lifecycle.

OnToology is a web-based system that integrates a set of existing tools for documentation, evaluation and publishing. By the time of writing this paper, OnToology has been used to produce the documentation, visualizations and evaluation of more than 100 projects and for the online publication of over 50 ontologies.

The paper is structured as follows. In Section 2, we describe a typical scenario we encounter during ontology development and the collection of requirements induced from it. Section 3 describes

* Corresponding author.
E-mail addresses: aalobaid@fi.upm.es (A. Alobaid), dgarijo@isi.edu (D. Garijo), mpoveda@fi.upm.es (M. Poveda-Villalón), isantana@fi.upm.es (I. Santana-Perez), albafernandez@fi.upm.es (A. Fernández-Izquierdo), ocorcho@fi.upm.es (O. Corcho).

[1] http://github.com/.

https://doi.org/10.1016/j.websem.2018.09.003
1570-8268/

the structure of OnToology, user interaction, and the details of the generated resources. Section 4 presents the evaluation of our approach. A review of the related work is provided in Section 5. Section 6 concludes and presents future lines of work.

2. Motivation scenario

In this section we illustrate a typical scenario that motivated the development of OnToology, based on the authors' years of experience developing ontologies and common use cases reported in state of the art ontology development methodologies [3].

Consider a team of ontology developers that are generating an ontology network in a particular domain. In order to build each ontology, the developers need to interact with domain experts that are not necessarily familiar with semantic technologies or ontologies. Therefore, such domain experts need *visualizations* and a *human readable documentation* to understand the taxonomy and relations included in a vocabulary, and assess whether it addresses their requirements. Once they conclude their assessment, they provide feedback to the ontology development team, which incorporates it in a *new version/release of the ontology* and starts a *new documentation process*.

Ontology developers also need to *evaluate the quality of the generated ontologies* during their development process, so as to identify potential errors before making them public as a new release. Due to the parallel development of the ontology network by different ontology developers, the team needs to make any quality report public to all its members, so they are aware of any potential issue with any published ontology.

When a version of the ontology network is ready to be released, the ontology development team needs to *publish the ontologies* online following best practices (i.e., content negotiation of the ontology in different standard formats and permanent URIs [10]). After the ontology network is published, other researchers may extend any of its ontologies. Thus, new releases of ontologies in the network would need to explain new changes in their documentation so any third party can assess their impact.

Once the target ontologies are built and published, data producers may ellaborate a JSON-LD context[2] in order to publish data following their specification. These contexts need to be updated on every iteration [11] to properly reflect the latest changes of the ontology network.

2.1. System requirements

Based on the above scenario, we have collected the following set of general requirements to address it, which are later tackled by the features provided by OnToology in Section 4:

R1. Ontology Documentation: Ontology adopters often require human-readable documentation of developed ontologies with a definition of their classes, properties, and individuals; including figures illustrating its main structure.

R2. Automatic change-based resource generation: Automatically produced files (e.g., HTML documentation, figures of ontology visualizations, quality reports) need to be updated every time a new change in the ontology is registered.

R3. Coexistence with non-ontological resources: Automatically generated resources may reside within a given code repository without affecting the rest of the files in the project (e.g. source code).

R4. Compatibility with a collaborative workflow: Users often work with each other to develop ontologies, discussing issues and recording the reached consensus. Any tool or instrument designed to help them should be integrated within their workflow in a non-intrusive manner. For an adequate collaboration between multiple users, it is also required to control who is able to view and/or contribute to the project.

R5. Versioning: It is crucial to record changes between ontology versions and track the evolution of the ontology with the option to revert to an older version is required.

R6. Publication: When released, an ontology version must follow best practices for ontology publication[3] with appropriate content negotiation (i.e., HTML for humans, RDF for software agents) and updating its respective JSON-LD context if required. Publication of the ontology under a permanent URI is also required in some cases to ensure its long-term accessibility.

R7. Evaluation: Before publishing an ontology, it is crucial to detect potential modeling errors [3] and be able to share them with the rest of the team.

3. System description

In order to address the set of requirements introduced in Section 2, we have designed OnToology,[4] an open source web application that monitors changes in ontologies stored in a Git repository and triggers a series of actions for supporting the ontology development process. These actions include: (1) generation of customized documentation, diagrams and evaluations for each of the ontologies contained in a repository, according to a configuration set for each ontology; (2) generation of a landing page containing basic descriptions of the ontologies included in the repository as well as a dedicated web page per ontology with extended descriptions and metadata; and (3) publication of the ontology under a w3id permanent URI.[5]

OnToology is based on existing technologies and best practices from both the software and ontological communities. We have integrated it with GitHub, one of the most popular hosting platforms for version control and collaborative development. GitHub serves as the main online infrastructure to maintain ontology versions as they are being developed and publish their associated resources. In addition, GitHub provides the ability to open issues and maintain discussions over a certain modeling decision, while giving the usual support to the rest of artifacts, processes, discussions, etc.

In this section we introduce the main features of OnToology, the user interaction with the system, and the main results that are produced.

3.1. System architecture

As depicted in Fig. 1, OnToology is composed of three main layers: the *presentation layer*, the *logic layer*, and the *persistence layer*.

The Presentation Layer exposes a web-based GUI, which allows for registering GitHub repositories. Given a repository, each ontology may be individually configured (enabling/disabling the automatic generation of resources).

The Logic Layer serves as the glue for integrating the different tools and services that compose OnToology. We can distinguish three main modules in this layer: the *Change Monitor*, *Orchestration service*, and the *Integrator*.

The *Change monitor* acts as a monitoring service for keeping track of changes in a GitHub repository. The *Orchestration service* controls the main actions of the system, and decides when to invoke the different tools integrated in the OnToology suite. Finally, the *Integrator* module is responsible for the execution of processes,

[2] https://www.w3.org/TR/json-ld/.

[3] https://www.w3.org/TR/swbp-vocab-pub/.

[4] http://ontoology.linkeddata.es/.

[5] http://w3id.org/.

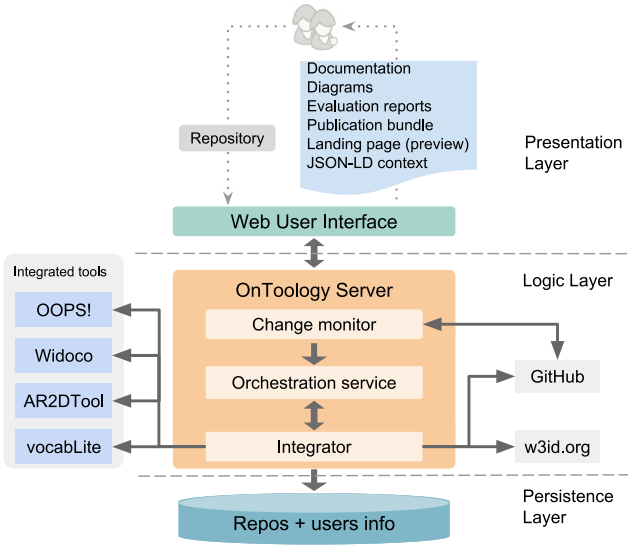


Fig. 1. OnToology Architecture.

implemented within OnToology or third party applications. In this sense, the *Integrator* module handles the following functionalities:

- Documentation, including several resources generated by different tools. First, it generates an ontology HTML documentation, supported by the integration of WIDOCO [12], a standalone application for generating HTML documentation for an individual ontology. WIDOCO extracts metadata properties from ontologies, generates several serializations for content negotiation and creates a provenance summary that records the changes from earlier versions. Second, ontology diagrams are created by AR2DTool [13], a library for generating diagrams based on RDF-encoded information. AR2DTool is meant for drawing diagrams of ontologies, generating not only compiled figures (i.e. PNG files) but also their source code (i.e. GraphML[6] files) for users to further edit them. Third, OnToology generates a repository pre-visualization page, which is supported by the integration of VocabLite [14], an application designed to create of overview pages for a set of ontologies (and their metadata) in a repository.
- Evaluation, by means of OOPS! [15],[7] a web application for ontology diagnosis that automatically detects 33 types of pitfalls in OWL ontologies, including semantic and structural checks as well as best practices verification. For each identified pitfall in the ontology, OOPS! provides its title, description, list of ontology elements affected and its importance level (critical, important or minor). It is worth noting that OOPS! does not perform reasoning on the ontology, as it is a time-consuming task and costly in terms of resources. However, OOPS! detects pitfalls that may lead to reasoning issues. In case of need, users may execute reasoners over their ontologies within their ontology edition environment such as Protégé before registering them in OnToology.
- Publication, providing different features following ontology publication best practices. OnToology allows users to reserve a name for their ontology with a permanent URI following the scheme https://w3id.org/def/<user-chosen-name>. OnToology uses the w3id services to point to the location of the published ontology and its resources with content negotiation. Alternatively, OnToology allows downloading a publication-ready bundle containing all the resources needed to publish an ontology on a custom server.
- JSON-LD context, generated using owl2jsonld [16], a tool that maps the classes and relation from the ontology to JSON keys.

The Persistence Layer maintains an internal database for storing information about users and repositories, as well as the current status of the repositories (i.e., the progress while processing ontologies in a given repository). This allows users to be aware of the pending actions to be performed by OnToology (e.g., generating the documentation or diagrams, validating the ontology, etc.).

3.2. User interaction with ontology

OnToology is accessible through a web interface, which requires no installation and allows configuring which actions are triggered for each ontology.

Fig. 2 depicts an overview of the user interaction workflow with OnToology. The process starts after a user has created a GitHub repository, in which at least one ontology file is stored. Once the user adds the repository to OnToology, the system will ask for permission to add the "OnToologyUser" as a collaborator to the repository, and link a *webhook* for enabling notifications to OnToology when any changes are made to the repository. This set up phase is represented by the second step in the workflow in Fig. 2.

After this, users can edit their ontology and once the changes are pushed to the GitHub repository, OnToology is notified through the corresponding webhook. A fork of the tracked repository will be created by OnToology and used as the working repository. The system then generates the appropriate files for all monitored ontologies, and pushes the changes to the forked repository. Finally, OnToology creates a pull request from the forked repository to the user's one in GitHub, including the new version of all the generated files. The user can then review the pull request and decide whether to accept it and merge it to the GitHub repository.

In most cases users need to customize some parts of the HTML documentation generated by OnToology, such as the ontology release date or additional information about the changes made. Once the resources have been produced and customized, users may want to publish the ontology online. Two different scenarios may occur:

- It is the first time the ontology is being published. The user can choose between: (i) publishing the ontology under a permanent URL using OnToology to handle the publication, or (ii) downloading a bundle containing the HTML documentation; ontology serializations; and an *.htaccess* file to handle the content negotiation in a server controlled by the user.
- If the ontology is already published, the user can choose between: (i) re-publishing the ontology under the same permanent id, or (ii) downloading a new bundle from OnToology web user interface for publication.

3.3. Generated resources

All resources generated by OnToology will be located in the tracked GitHub repository under a new folder called "OnToology", which includes a folder per ontology in the repository. Each ontology folder contains: (1) a *documentation* folder with the HTML documents divided by sections to facilitate their editing, (2) an *evaluation* folder with HTML-based report describing different pitfalls found in the ontology, (3) a *diagrams* folder containing depictions of the ontology with their source files for further customization. A detailed example of the resources generated by OnToology for a given vocabulary can be found in Appendix.

6 http://graphml.graphdrawing.org/.
7 http://oops.linkeddata.es/.

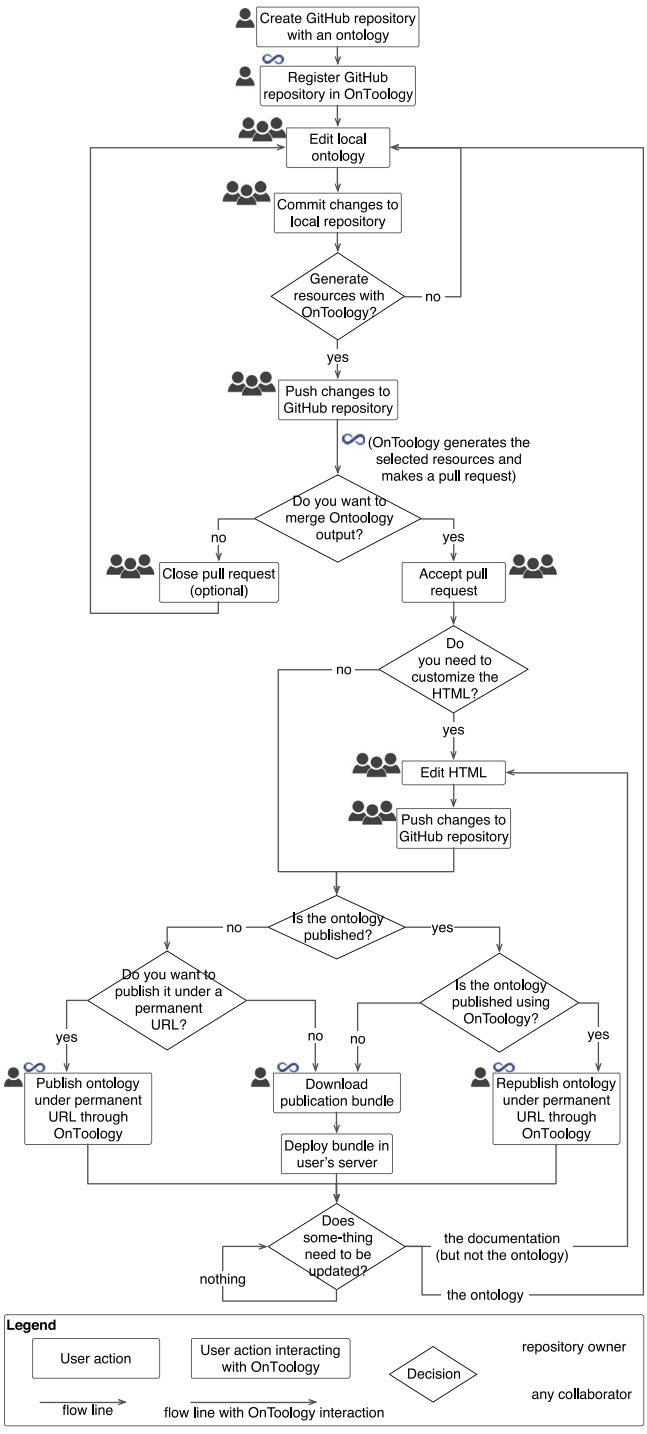


Fig. 2. User workflow in OnToology.

OnToology is available as a web service, and its code has been released online [17].

4. Evaluation

In this section we describe the validation of the system based on the coverage of the requirements presented in Section 2. Next, we introduce and discuss the results of a user survey we conducted, studying the usefulness, ease of use, ease of learning and satisfaction with the system. Finally, we present the community uptake and the adoption of OnToology in various ontology development projects.

Table 1
User survey results for usability satisfaction.

	Usefulness	Ease of use	Ease of learning	Satisfaction
Median	4	3.5	4	4
Mean	4.2	3.5	3.7	3.9

4.1. Requirement validation

The requirements for supporting ontology evaluation are covered by the integration of OOPS!, which produces a report including basic structural, semantic and best practices checks for an ontology (R7). The documentation of the ontology is produced by the combination of WIDOCO, VocabLite and AR2DTool (R1). The generation of these resources (e.g. documentation and evaluation) is triggered automatically after each ontology change (R2). With OnToology, users can contribute collaboratively to a registered repository (if they have the appropriate permission) and resources will be generated automatically as users push new changes to the repository (R4). The resources generated by OnToology will not interfere with other existing resources (e.g. source codes, readme files, etc.) as they are stored in a separate folder called "OnToology" in the top level of the repository (R3). OnToology supports the publication process by exporting a single bundle that a user can retrieve as a snapshot for publishing the ontologies, and if desired, OnToology enables the content negotiation of an ontology release, and publishes the files online with persistent URLs (R6).

OnToology benefits from a Git-based existing infrastructure that is widely used among the software development community (GitHub). This infrastructure supports several of the requirements presented in Section 2, such as having distinct roles for collaboration, and providing access to discussion channels by using discussion boards where users can reply and share their thoughts about a given design decision (R4). GitHub also allows for tracking the releases of an ontology and records how users contribute to a project with release tags (R5).

4.2. User-based evaluation

We have conducted an evaluation process, collecting user feedback by means of an open call to the community of OnToology users. It should be mentioned that none of the participants received training or guidance on how to use OnToology beforehand in order to avoid bias towards the tool.

Users were asked to evaluate their interaction with OnToology, in terms of usefulness of the tool, its usability and overall satisfaction. In order to collect their feedback we have developed an online questionnaire, adapted from the USE Questionnaire (usefulness, satisfaction, and ease of use) [18], in which we have reduced the number of questions in order to make it more friendly for users. The feedback form consists of 16 quantitative questions divided into the aforementioned categories (usefulness, satisfaction and usability). These questions are rated in a Likert scale from 1 (strongly disagree) to 5 (strongly agree). Besides the quantitative questions, we included questions for profiling user expertise as well as for collecting comments on the tool.

The questionnaire and all the collected results, both raw and aggregated values, are available online.[8] We received 15 responses to the questionnaire, 66% from experts in ontology engineering, 26% from users who were familiar but not experts and 6% from

[8] http://ontoology.github.io/OnToology/journalofwebsemantics/#questionnaire.

users who were not familiar at all with ontology engineering. The quantitative results obtained through the user based evaluation are depicted in Table 1, showing the median and mean scores aggregated for the above-mentioned categories. For all the analyzed categories, the scores for the median and mean are above or equal to 3 (out of 5), being 2.5 the threshold for considering it a positive rating. In particular, those categories more relevant to OnToology features (i.e. usefulness and satisfaction), obtained high scores, whereas those related to the ease of use of the system (i.e. usability) are around 3. This fact shows that users actually perceived an improvement in their productivity during the ontology development process thanks to OnToology's features. However, it also indicates that there is still room for improvement regarding the user experience, as the ease of use and learning is not that well rated.

As part of the evaluation form we also included three open questions, asking the users about whether they have used other similar tools before and about the most negative and positive aspects of OnToology. Most of the users reported that they had not used any tool in this context. Regarding the negative aspects, most respondents stress the need for better documentation and a more informative interface (e.g., The "publish ontology" feature is quite hidden), which aligns with the ease of learning and usability scores obtained in our survey. Among the most positive aspects, users generally state that OnToology implements interesting features, saving them time when dealing with supporting activities, being fast and "automatizing boring tasks", without having to install or set up any environment.

Overall, we can consider that, based on the obtained results, OnToology addresses successfully user needs, listed in Section 2, being useful when carrying out the support activities related with ontology development.

4.3. Community uptake

Since OnToology was announced[9] and at the time of writing this paper, a total of 113 repositories containing more than 530 ontologies have been added to OnToology,[10] including projects from different knowledge domains such as energy, e-Science, smart cities, etc. Examples of these projects are OpenCityData,[11] BOT[12] or Eiffel.[13]

Statistics of user activity in OnToology[14] indicate that the bundle download option has been used 29 times, while the ontology publication feature has been activated in 57 occasions. Thanks to this feature, the HTML documentation of ontologies hosted by OnToology has been accessed 843 times, while the different serializations of their RDF code have been requested 382 times.

As these statistics show, OnToology has been and still is used and helpful for users on the tasks related to publication of their ontologies, as most of them tend to publish their ontology using OnToology rather than downloading the corresponding bundle. More examples on the adoption of OnToology in different projects can be found in OnToology website.[15]

5. Related work

In recent years, different systems have been developed to support collaborative ontology development. One of the most well-known tools is WebProtégé [19]. Besides ontology editing functionalities, WebProtégé provides a discussion board and functionality for annotating ontology terms. Once an ontology is generated, developers may resort to their local installation of Protégé to produce documentation and diagrams using plug-ins. WebProtégé does not integrate features for the online publication of the ontology. Another collaborative approach is Moki [8], a tool for modeling ontologies based on MediaWiki. Moki provides either a light-weight view or a full source-code view of the ontology. It also integrates evaluation functionalities like a model checklist, quality indicators and ontology publication, but it does not provide any documentation functionalities.

Neologism [7] was also designed to provide support for the online development process of ontologies, as a vocabulary editor and publishing system. Neologism provided an offline file-based model and an automatic diagram creation. However, Neologism did not address the evaluation and versioning of its ontologies. It should also be mentioned that this system is no longer maintained. A more recent effort is VoCol [9], designed as a tool to help collaborative vocabulary development and use Git repositories to maintain vocabulary-related files. VoCol provides support for project management, quality assurance, documentation and visualization components. Both Neologism and VoCol provide support for publishing ontologies and their documentation.

Fig. 3. Register a repository.

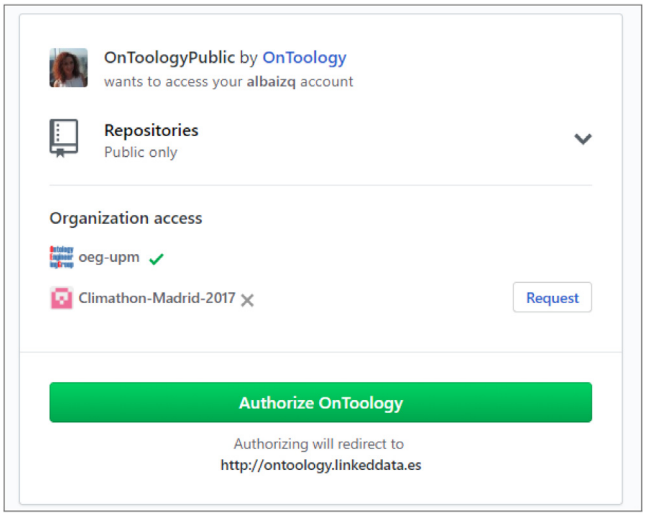


Fig. 4. OnToology authorization.

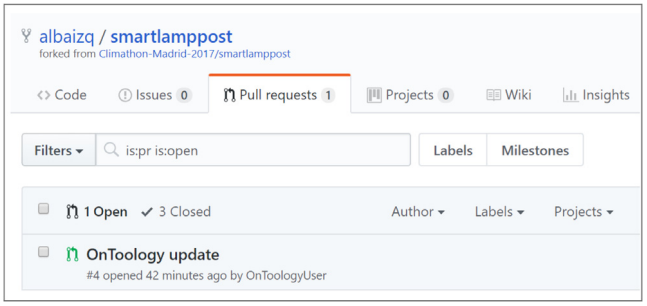


Fig. 5. Pull request.

[9] https://lists.w3.org/Archives/Public/public-vocabs/2015Jul/0003.html.
[10] http://ontoology.github.io/OnToology/journalofwebsemantics/#ontologies.
[11] https://github.com/opencitydata.
[12] https://github.com/GeorgFerdinandSchneider/bot.
[13] https://github.com/hartig/eiffel-rdf-vocabularies.
[14] http://ontoology.github.io/OnToology/journalofwebsemantics/#stats.
[15] http://ontoology.linkeddata.es/faqs.

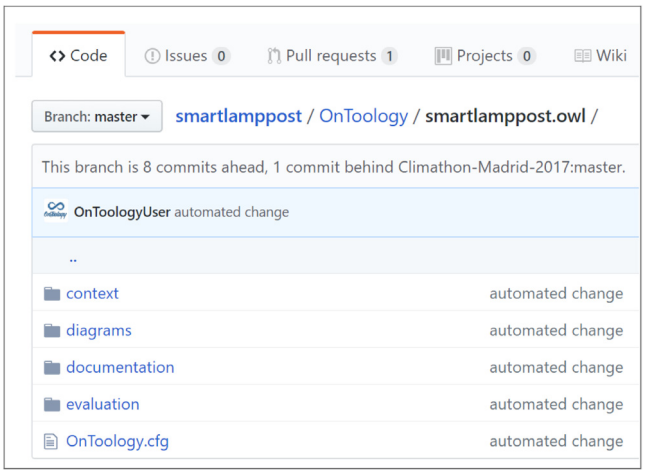


Fig. 6. Merged resources.

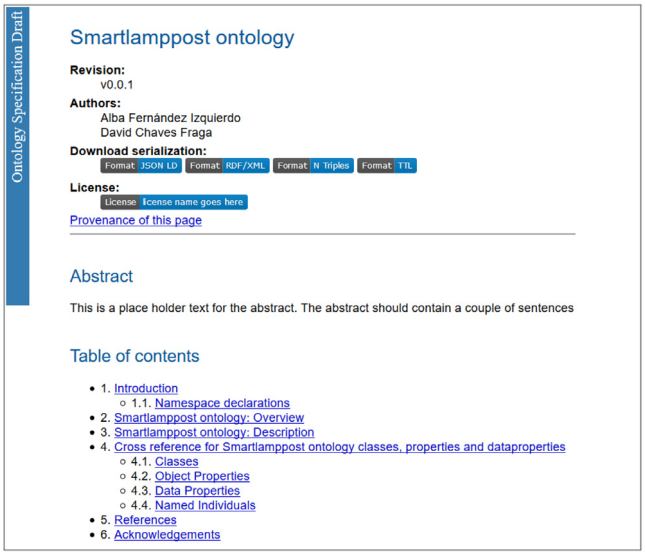


Fig. 7. Documentation resource.

VocBench [20] is an open source web application for editing thesauri. VocBench allows for collaborative management of the overall editorial workflow, by introducing different roles with specific competencies, and provides features for content validation and publication of vocabularies. Furthermore, it provides changes history and a SPARQL query service. However, VocBench does not provide any documentation or evaluation functionalities, and focuses only on SKOS models.

Table 2 illustrates how these tools can be positioned according to the requirements defined in Section 2. For each requirement it is indicated whether a tool supports it by means of the symbol "✓" or whether is partially covered "≈". As shown in the table, none of these tools (except for OnToology) address all the identified support activities for ontology development. The most complete one in this regard is VoCol, as it allows easy documentation and publication of ontologies in an integrated solution. However, VoCol does not allow users to decide on a naming scheme when publishing their ontologies, it cannot assign permanent identifiers to an ontology and it is cumbersome to edit each of its produced outcomes (e.g., adding a new figure or paragraph in the documentation).

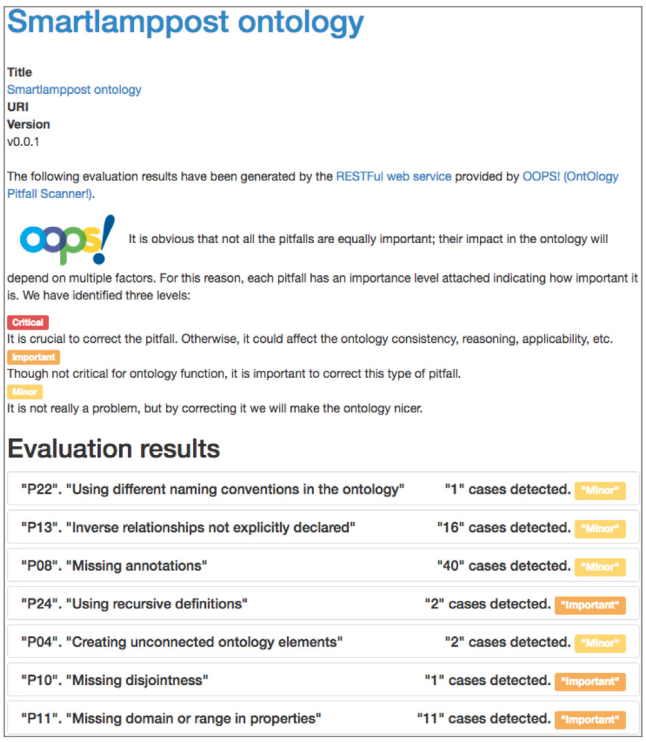


Fig. 8. Evaluation resource.

```
{
  "@context" : {
    "IPv6" : {
      "@id" : "http://www.semanticweb.org/def/smartLamppost#IPv6",
      "@type" : "http://www.w3.org/1999/02/22-rdf-syntax-ns#PlainLiteral"
    },
    "LightBulb" : {
      "@id" : "http://www.semanticweb.org/def/smartLamppost#LightBulb"
    },
    "UnitOfMeasure" : {
      "@id" : "https://w3id.org/saref#UnitOfMeasure"
    },
    "hasPrice" : {
      "@id" : "http://www.semanticweb.org/def/smartLamppost#hasPrice",
      "@type" : "@id"
    },
    "versionInfo" : {
      "@id" : "http://www.w3.org/2002/07/owl#versionInfo"
    },
    "Filter" : {
      "@id" : "http://www.semanticweb.org/def/smartLamppost#Filter"
    },
    "Photometer" : {
      "@id" : "https://w3id.org/def/saref4envi#Photometer"
    },
    "CompassDirection" : {
      "@id" : "https://w3id.org/def/saref4envi#CompassDirection"
    },
    "SmartLamppost" : {
      "@id" : "http://www.semanticweb.org/def/smartLamppost#SmartLamppost"
    },
```

Fig. 9. Context resource.

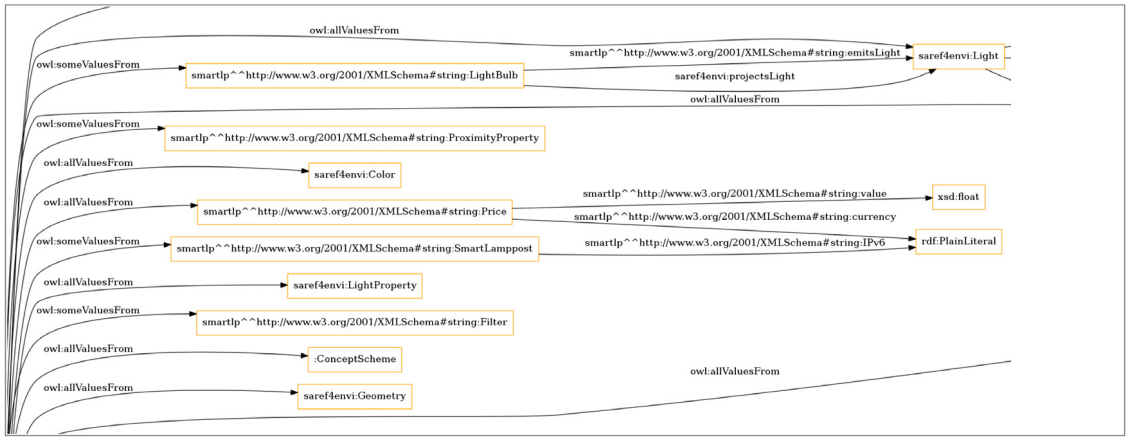
6. Conclusions and future work

In this paper we have described OnToology, an approach for automating ontology engineering support activities including documentation, evaluation and publication. OnToology considers ontologies as a first class citizen in software development projects and provides means for their continuous integration. OnToology is built on top of GitHub, integrating a suite of existing tools and services to support different ontology development scenarios.

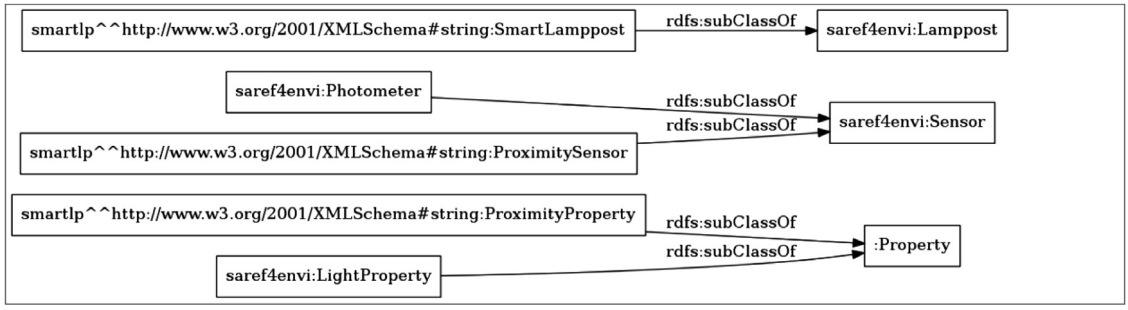
Table 2
Comparison of tools for supporting collaborative ontology development.

Support category	Moki	Neologism	VocBench	VoCol	Web Protégé	OnToology
R1. Documentation		✓		✓		✓[avw]
R2. Complementary resources	≈	≈		≈		✓[ajow]
R3. Coexistence with non-ontological resources						✓
R4. Compatible with a collaborative workflow	✓	✓	≈	✓[g]	✓	✓[gh]
R5. Versioning	✓		≈	✓[g]	≈	✓[gh]
R6. Publication and content negotiation	✓	✓		≈	✓	✓[w]
R7. Evaluation	✓			✓		✓[o]

✓ supported
≈ partially supported.
[a]Supported by AR2DTool.
[g]Supported by Git.
[h]Supported by GitHub.
[j]Supported by OWL2JSONLD.
[o]Supported by OOPS!
[v]Supported by VocabLite.
[w]Supported by Widoco.



(a) Class diagram



(b) Taxonomy diagram

Fig. 10. Ontology visualization.

We have evaluated OnToology in terms of usefulness, usability and ease of use with a user survey, which has provided positive feedback regarding usefulness of and satisfaction with the tool. Whereas the study suggests that the interface could be improved in terms of usability, results show that the system is useful and that its features are relevant in the context of ontology development. Users value especially the features related to documentation and publication, as well as the ability to release their ontologies using OnToology features, without having to set up their own infrastructure.

Regarding future work, we are now integrating OnToology with other Git-based platforms more commonly used for private projects, such as Bitbucket[16] or GitLab.[17] We plan on adding reasoning capabilities to notify the user on potential inconsistencies in a configurable way. We also plan on integrating OnToology to publish ontologies with Linked Open Vocabularies [21].

Finally, in order to improve the usability and learning curve of OnToology, we are currently improving tutorials and the GUI of the system. For this task, we pay special attention to user suggestions collected in our GitHub project.[18] Following this idea, and while

16 https://bitbucket.org/.
17 https://about.gitlab.com/.
18 https://github.com/OnToology/OnToology/issues.

Fig. 11. Reserve a name.

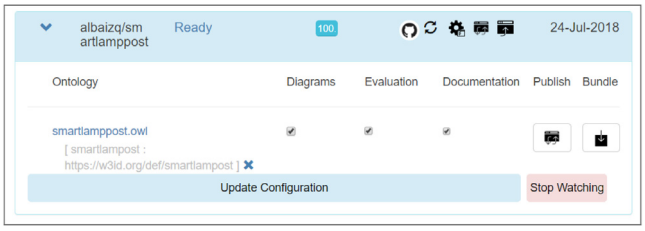


Fig. 12. Publication of the ontology.

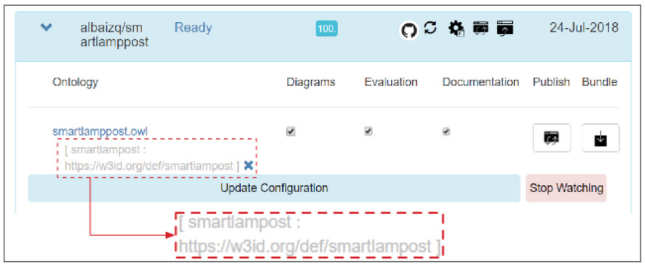


Fig. 13. Re-publication of the ontology.

keeping OnToology editor-agnostic, we plan to integrate the system with popular editors such as Protégé, allowing users to manage the version-release cycle directly while editing an ontology.

Acknowledgments

This work has been partially supported by the project Datos 4.0: retos y soluciones (TIN2016-78011-C4-4-R), funded by the Spanish Ministerio de Economía, Industria y Competitividad, and a predoctoral grant from the I+D+i program of the Universidad Politécnica de Madrid. The authors would like to thank the community of users who have contributed, requested features or provided feedback on OnToology and Yolanda Gil for her comments and feedback.

Appendix. OnToology: a running example

In this section we present a running example of an actual ontology developed by a subset of the authors of this paper. We show the steps that we followed using OnToology, accompanied with snapshots capturing the state of the tool.

1. **Register the repository:** We add the repository *albaizq/smartlamppost* to OnToology via the web interface (Fig. 3). The repository already includes the ontology file *smartlamppost.owl*.
2. **Authorize OnToology to access the repository:** Due to the fact that this is our first repository using OnToology, GitHub asks us to authorize OnToology to access the repository. By clicking on "Authorize Application", as shown in Fig. 4, we authorize OnToology to track our repository. As a result, "OnToologyUser" will be added as a collaborator and a webhook will be created to track future changes on the repository. OnToology forks the repository of the user to be the working repository. Next, OnToology will generate all the resources automatically (i.e., HTML documentation, diagrams, quality report, and JSON-LD Context) in the working repository *OnToologyUser/smartlamppost*.
3. **Generation of the pull request:** OnToology creates a pull request with all the generated resources from the working repository (*OnToologyUser/smartlamppost*) to the original one (*albaizq/smartlamppost*). We can see in GitHub the generated resources under the folder "OnToology" in the top level of the repository. Fig. 5 shows the pull request generated by OnToology.
4. **Merge resources:** After accepting this pull request, all the resources are merged into the repository. The structure of the generated resources is shown in Fig. 6. Screenshots of the generated resources are show in Figs. 7, 8, 9, 10a and 10b.
5. **Fix pitfalls:** We address the issues raised by OOPS! and push the changes to the ontology in the target repository. OnToology will automatically generate the resources again with the updated ontology. The documentation has been updated to match the new changes, the class and taxonomy diagrams have been updated accordingly, and the new quality report shows the pitfalls of the updated ontology. OnToology will create a new pull request with the updated resources. We merge the new resources by accepting the pull request.
6. **Publish the ontology:** At this point, the new generated resources according to the updated ontology are in the repository. We finally have reached a stable release, and we would like to publish it. The OnToology web interface provides a button to publish the ontology with w3id service. We choose the name *smartlammpost* and, therefore, our ontology will be published in https://w3id.org/def/smartlampost. Fig. 11 shows the interface to reserve a name for the ontology, while Fig. 12 shows the OnToology interface with the URI where the ontology is published.
7. **Discuss the new requirements:** After some time, a new requirement appears and the ontology needs to be extended. But one of the developers has a doubt about a specific issue and asks the question in the discussion board. The team leader answers it and they start a discussion with other developers and agree about how to address the new requirement.
8. **Update the ontology:** Developers start updating the ontology to meet the new requirements and push the changes using Git. The resources will be regenerated for the new ontology and a pull request will be created. Changes are checked and the generated resources match the expectation.
9. **Re-publish the ontology:** The developers finish the implementation of the new requirements to the ontology and the team leader republishes it using the republish button in the web interface of OnToology. Fig. 13 shows the interface for re-publishing the ontology. The republish button will only be visible if the ontology has been published before, otherwise, it will not be shown.

References

[1] M. Fernández-López, A. Gómez-Pérez, N. Juristo, METHONTOLOGY: from ontological art towards ontological engineering, in: Proceedings of the Ontological Engineering AAAI97 Spring Symposium Series, American Association for Artificial Intelligence, 1997.

[2] S. Staab, R. Studer, H.-P. Schnurr, Y. Sure, Knowledge processes and ontologies, IEEE Intell. Syst. 16 (1) (2001) 26–34.

[3] M.C. Suárez-Figueroa, A. Gómez-Pérez, M. Fernández-López, The NeOn Methodology framework: A scenario-based methodology for ontology development, Appl. Ontol. 10 (2) (2015) 107–145.

[4] S. Auer, The RapidOWL methodology–towards agile knowledge engineering, in: 15th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE 2006), IEEE Computer Society, 2006, pp. 352–357.
[5] V. Presutti, E. Blomqvist, E. Daga, A. Gangemi, Pattern-based ontology design, in: M.d.C. Suárez-Figueroa, A. Gómez-Pérez, E. Motta, A. Gangemi (Eds.), Ontology Engineering in a Networked World, Springer, 2012, pp. 35–64.
[6] S. Peroni, A simplified agile methodology for ontology development, in: Proceedings of the 13th OWL: Experiences and Directions Workshop and 5th OWL reasoner evaluation workshop (OWLED-ORE 2016), Springer, 2016, pp. 55–69.
[7] R. Cyganiak, C. Basca, S. Corlosquet, T. Schandl, S. Fernández, Neologism: Easy Vocabulary Publishing, 2008.
[8] A. Bosca, M. Casu, M. Dragoni, A. Rexha, Modeling, managing, exposing, and linking ontologies with a wiki-based tool, in: Proceedings of LREC, 2014, p. 1668.
[9] L. Halilaj, N. Petersen, I. Grangel-González, C. Lange, S. Auer, G. Coskun, S. Lohmann, VoCol: an integrated environment to support version-controlled vocabulary development, in: Proceedings of the 20th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2016), Springer, 2016, pp. 303–319.
[10] D. Berrueta, J. Phipps, Best Practice Recipes for Publishing RDF Vocabularies, W3C, https://www.w3.org/TR/swbp-vocab-pub/, 2008.
[11] O. Corcho, I. Santana-Pérez, H. Laguente, D. Portolés, C. Cano, A. Peris, J.M. Subero, Publishing linked statistical data: Aragón, a case study, in: Joint Proceedings of the International Workshops on Hybrid Statistical Semantic Understanding and Emerging Semantics, and Semantic Statistics (HybridSemStats), 2017.
[12] D. Garijo, WIDOCO: A wizard for documenting ontologies, in: International Semantic Web Conference, Springer, 2017, pp. 94–102.
[13] I. Santana-Pérez, D. Garijo, F. Siles, AR2DTool, 2018. http://dx.doi.org/10.5281/zenodo.1317796.
[14] D. Garijo, M. Poveda-Villalón, dgarijo/vocabLite: Vocablite 1.0.2, 2018. http://dx.doi.org/10.5281/zenodo.1318782.
[15] M. Poveda-Villalón, A. Gómez-Pérez, M.C. Suárez-Figueroa, OOPS! (OntOlogy Pitfall Scanner!): An on-line tool for ontology evaluation, Int. J. Semant. Web Inf. Syst. (IJSWIS) 10 (2) (2014) 7–34.
[16] S. Soiland-Reyes, owl2jsonld 0.2.1, Convert OWL ontology to JSON-LD context, 2014. http://dx.doi.org/10.5281/zenodo.10565.
[17] A. Alobaid, D. Garijo, M. Poveda-Villalón, I. Santana-Perez, A. Fernández-Izquierdo, OnToology, 2018. http://dx.doi.org/10.5281/zenodo.1317786.
[18] A.M. Lund, Measuring usability with the USE questionnaire, Usability Interface 8 (2) (2001) 3–6.
[19] T. Tudorache, J. Vendetti, N.F. Noy, Web-Protégé: A lightweight owl ontology editor for the web, in: OWLED, Vol. 432, 2008.
[20] A. Stellato, S. Rajbhandari, A. Turbati, M. Fiorelli, C. Caracciolo, T. Lorenzetti, J. Keizer, M.T. Pazienza, VocBench: A web application for collaborative development of multilingual thesauri, in: European Semantic Web Conference, Springer, 2015, pp. 38–53.
[21] P.-Y. Vandenbussche, G.A. Atemezing, M. Poveda-Villalón, B. Vatant, Linked Open Vocabularies (LOV): a gateway to reusable semantic vocabularies on the Web, Semant. Web 8 (3) (2017) 437–452.