Help Rich Info Get Richer: Enriching a semi-structured dataset using a Semantic Web approach

YUJIE LIU, University of Twente, The Netherlands

This article presents a data integration approach to enrich a semi-structured dataset called ProVerB. ProVerB is a project to explain and classify program verification tools. It offers some URLs linking to external supporting resources for each tool. However, these URLs contain little information, and a user needs to explore them manually to find what they need. Therefore, a data integration project is conducted to supply these tools with additional information by fetching original data from these URLs. The solution is implemented using a Semantic Web approach to preserve the structure of ProVerB and explore possible relationships among incoming data. The new data and their potential relationships are inserted back into ProVerB to help this dataset become richer.

Additional Key Words and Phrases: Semantic Web, Data integration, Program verification, Ontology, Open API

1 INTRODUCTION

The ProVerB [12, 13] project aims to explain and classify program verification tools and helping software developers find the tools on their demand. It is a part of the SLEBoK project [6] in a specific domain. ProVerB is constructed with semi-structured data. For each tool, a brief introduction and sample input/output are given, along with internals linking to other tools in the project and URLs linking to external supporting resources, like project repositories and related papers. All these tools are classified into six hierarchy PV levels based on their application scenarios, namely PV1 - PV6.

1.1 Research goals and requirements

Since ProVerB only contains basic information, the users need to turn to the supporting resources to get details about the tools they are interested in. However, these URLs contain little information, and the users need to click on them and find what they need in the new tabs manually. This process is cumbersome when quite a few tools need to be reviewed. For example, suppose a user wants to find the tool that is under the most active development from several ones. In that case, the user needs to open several tabs for repositories to see the latest commit of each tool.

Moreover, a cross-reference for the data with the same semantic meaning but from different sources can be helpful for the user to explore the details of the tools. For example, exploring the authors of the tool's related papers and the code contributors of the tool to see who may be the most suitable one to contact for the latest research of this tool.

Therefore, the goals of this research are:

- **G1**: Find an efficient way to reduce the cost of the repeated click & browse process for the ProVerB users.
- **G2**: Find a way to explore the possible relationships for the incoming data from different supporting resources.

We also notice that although the ProVerB dataset is not huge, manual data modifications should also be avoided in achieving our goals. What's more, the dataset is visualized on a website hosted by GitHub IO. A python script is used to auto-generate the web pages following the structure of the dataset. This structure should also be intact after modifications to not destroy the website generation.

Based on these facts, two requirements that this research should follow are proposed below:

- **R1**: The structure of the ProVerB dataset should be preserved.
- **R2**: The process of achieving the goals should be automatic to minimize manual operations.

1.2 Technology

The main technologies in this research are explained below, along with a brief introduction about why and where these technologies were used in our approach. Some references are also provided for the reader to get more if necessary. Similar works using some of these technologies are discussed in 2. The implementation of these technologies in our approach are detailed explained in section 3.

A data integration project can be introduced to solve the **G1** problem by using mashup [10] methodology to build a Semantic Web application [3], compositing information from different sources into one dataset. A mashup is a web-based application that is created by combining and processing content from more than one online third-party resource that contributes to data and/or presentation [11]. In this research, the mashup approach is used to combine the semi-structured data from ProVerB and data from third-party open APIs like GitHub and Springer.

The Semantic Web approach is chosen because we need to consider the semi-structural nature of the ProVerB dataset (**R1**). To preserve the structure during the integration with additional information, the project is implemented with a Semantic Web application based on a customized ProVerB ontology.

An ontology is an engineering artifact widely used in the Semantic Web approach, which model (some aspect of) the world. It introduces vocabulary describing various aspects of the domain being modeled, and provides an explicit specification of the intended meaning of that vocabulary [8]. For example, Figure 1 shows a basic model of ontology. The Author and Book are two classes in this ontology, each class has a data property called 'name'. And the 'writer' is an object property indicating the relationship between these two classes.

A particular file type called Resource Description Framework (RDF) [7] is used to persist the ontology with classes and their properties. Figure 2 shows an example of how data is stored in an RDF file in a triple format, i.e. <Domain, Relationship, Range>. The

TScIT 37, July 8, 2022, Enschede, The Netherlands
© 2022 Association for Computing Machinery.
This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record was published in , https://doi.org/10.1145/nnnnnnn.nnnnnnn.

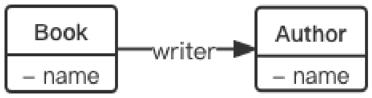


Fig. 1. Example of ontology

edges represent the named relationship between the two instances of the Domain and Range class, represented by the graph nodes.

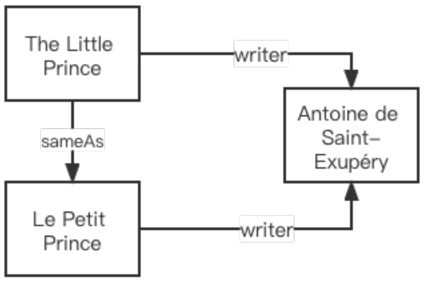


Fig. 2. Example of RDF and Linked Data

Similar approaches are widely used to build a formal dataset conforming to OWL ontology in Semantic Web application development [8, 10, 23], which are discussed in section 2.

To fulfill the goal **G2** based on the data integration project of **G1**, the application also performs a cross-evaluation of the data to distinguish the possible relationships, following the predefined patterns written by the ProVerB maintainers. In this research, we mainly focus on adding 'same as' relationship. which is one of the methods to construct linked relationship in Linked Data [4, 20, 22]. The Linked Data is a kind of structured data that contains interlinks with other data, which makes it more useful through semantic queries.

As you can see in Figure 2, it looks like there are two books with different titles, but in fact, it is one book with the same story translated into different languages. In this way, no matter the language we use to query, the system will know we are looking for the same concept.

A specific language called SPARQL [2] is also introduced to query the linked data. The structure of SPARQL is similar to SQL, but uses specifications in a triple format that follows the RDF specification. An example of SPARQL query can be found in Figure 6 with explanation in section 3.4.

Moreover, to fulfill the requirement **R2**, we also considered how to automate the enrichment process. GitHub workflow is chosen to support the entire process. Details about the workflow are introduced in section 3.

1.3 Research questions and contributions

The goals and the technologies used to reach the goals lead to the following two main research questions.

- **RQ1**: How to implement the data integration project of goal **G1**? This can be divided into three ordered sub-questions:
  - **RQ1-1**: How to design the ProVerB ontology?
  - **RQ1-2**: How to implement the integration application?
  - **RQ1-3**: How to save the composited data into one dataset?
- **RQ2**: How to cross-evaluate the data to fulfill the goal **G2**? This can be divided into two ordered sub-questions:
  - **RQ2-1**: How to explore the possible relationships?
  - **RQ2-2**: How to make sure that the explored possible relationships are valid?

The contributions of this research are as follows:

(1) **C1**: An application for data enrichment and relationship exploration for the ProVerB dataset. This is the deliverable of answering the **RQ1** and **RQ2**, which fulfills the goals of **G1** and **G2**.
(2) **C2**: Execute the enrichment process to get and transform the composited dataset back into the ProVerB's original dataset. This can ensure that the application in **C1** can be deployed, and this data integration process can be executed regularly on a schedule using GitHub workflow.

The remainder of this paper is organized as follows. Section 2 introduces the related work. In section 3, the implementation of our approach is discussed in detail. In section 4, we analyze the results and artifacts of the research. Section 5 discusses the limitations and future work of this research, and section 6 concludes.

2 RELATED WORK

In Semantic Web development, the ontology approach has become more popular for integrating heterogeneous data from different sources. These related research projects can be divided into two main categories based on research direction: how an ontology can be used in Semantic Web applications[21], and how an ontology is designed and implemented in a specific topic [8, 10, 23].

Souza et al. [21] use Web Ontology Language (OWL) [1] to design an ontology for web data integration in a specific domain and come up with five scenarios where semantic information can be used for integration. Compared to our work, they mainly focus on the domain of Information Quality (IQ) to explore the combination of semantic information and IQ.

Horrocks [8] gives a detailed introduction and a demo to show how an ontology can be used to provide the semantics. This article is an introduction about Semantic Web and ontology design, offering some high-level guidance for us to develop our own ProVerB ontology.

Kalou et al. [10], and Vivar et al. [23] come up with corresponding topic-specific ontologies to handle their problems in web book mashup application and academic data management. Compared to our research, we share a similar approach, i.e., building an ontology as a guidance for domain specific application, but we have different goals for respective research problems.

3 APPROACH

To answer the proposed research questions, we take the following six steps shown in Figure 3. After steps 1-5, the application for contribution **C1** should be delivered. Step 6 is the data transformation

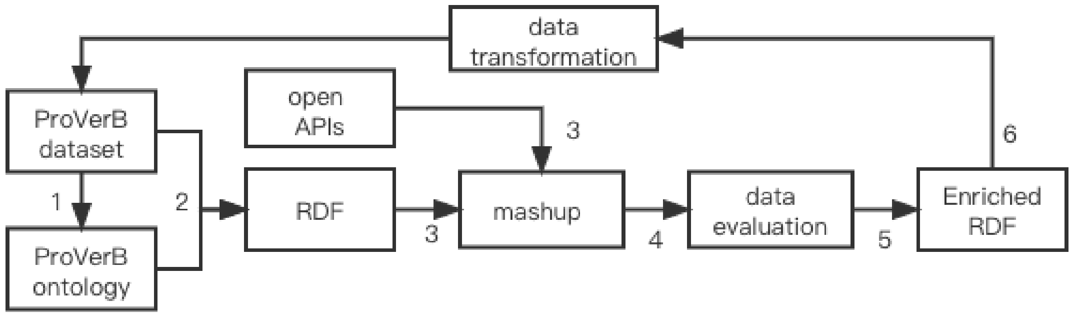


Fig. 3. Steps towards the enrichment solution

for contribution **C2**. The details of each step are introduced in the upcoming subsections.

The development of this six-steps process is performed in an iterative way. Each iteration will add more concepts into the ontology, get more data from different sources, extract possible relationships, and transform more data back to the original dataset. A GitHub workflow is deployed to support this process. This is because the ProVerB project is also hosted by GitHub, and the data enrichment process should also be platform compatible to maximize the process automation.

3.1 Step 1: Design the ontology

In the first step, an ontology is designed using OWL and Protege, which conforms to the structure of the ProVerB dataset and can be extended to fit the data structure of APIs. Protege [19] is a suite of tools to construct domain models with ontology. The ProVerB ontology is written in OWL and saved as triples in an RDF file [9]. In this step, the **RQ1-1** is answered.

The current version of the ProVerB ontology [17] includes 11 classes shown in Figure 4, with respective functions and properties listing in table 1 and table 2. The `Repository` and `Article` classes are used to instantiate the repository links and DOI links extracted from the original dataset and open APIs, which will be introduced in section 3.2 and 3.3. The `CodeContributor` and `Writer` classes are used to instantiate the contributors of the code and the authors of the paper, and explore the possible overlap between these two roles in section 3.4.

We also need to clarify that since the ProVerB ontology is only an intermediate product for the enrichment process and not a crucial part of this research, the design, development, and validation of this ontology are simplified to meet the only requirement, i.e., "If it works, it works." But we do leave some room for future study based on this ontology. For example, the `Concept` class is left for domain experts in the program verification area to classify the tools on their application domain. And the `Format` class is left for future research to explore the possible input-output relationship between tools. For example, if the output of one tool can be used as the input of another tool. These two directions introduced above are beyond the scope of this research.

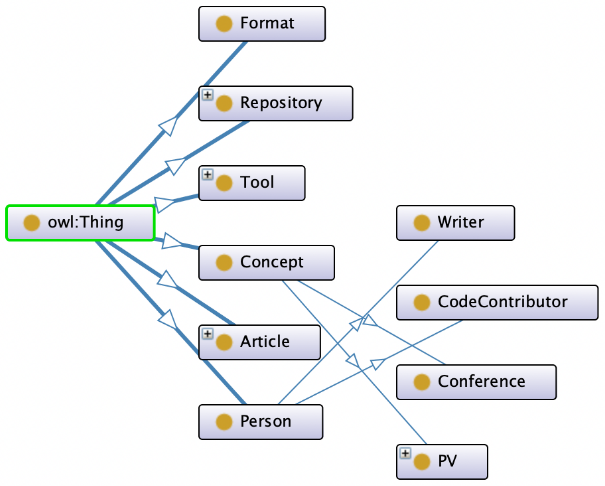


Fig. 4. ProVerB Ontology

3.2 Step 2: From ProVerB to RDF

In the second step, the data from the ProVerB dataset is imported into an RDF file containing the ProVerB ontology, which is an artifact released in the previous step. The extracted data is stored in triples format conforming to the ProVerB ontology. A helper tool [15] is designed and implemented to achieve this conversion. This helper tool is a part of the mashup application to answer **RQ1-2**, but in a different GitHub repository to decouple the application artifacts and simplify the GitHub workflow.

The implementation of this converter is based on pattern matching through manually adding patterns into the converter application. The converter will follow these patterns to extract the data from the original ProVerB dataset. The data includes the tool's name, URLs for the GitHub repository, and DOI links to the related paper. After extraction, the data will be cleaned based on the predefined patterns and escape some characters that the charset of RDF standard doesn't support.

Class	Functions	Example
Format	Input/output format of the tools and other specification formats	`.aadl` file
Repository	A GitHub or GitLab URL indicating the repository of the tools	https://github.com/loonwerks/AGREE
Tool	A tool in ProVerB dataset	AGREE
Concept	All tags, application domains, etc., can be the concepts. Some specific concepts in the ProVerB domain will be created as sub-class. This allows the domain experts to modify the ontology for the tool classification	Model checking
Conference	Sub class of Concept. The conferences that the proceedings come from	(Reliability Engineering & System Safety '21)
PV	Sub class of Concept. Six hierarchy levels that classify tools	PV 1 - PV 6
Article	An DOI link pointing to the publication of the tool	https://doi.org/10.1016/j.ress.2021.107649
Person	Persons related to the tools, the proceedings and any other concepts	
Writer	Sub class of Person. Specific to the author of Article	name of the writer
CodeContributor	Sub class of Person. Specific to the contributor of Repository	name of the contributor

Table 1. Classes with respective functions

Property	Domain	Range
author	Article	Person
category	Tool	Concept
contributor	Repository	Person
relatedpaper	Tool	Article
repository	Tool	Repository
name	Thing	
abstract	Thing	

Table 2. Ontology Properties

The GitHub Actions will support this conversion by fetching the newest ProVerB ontology and the newest ProVerB dataset as the inputs of the workflow. After that, the converter will be executed and finally publish the conversion artifact to a daily release. This artifact includes the ProVerB ontology and the data from the ProVerB dataset, which will be used as an input for the next step.

3.3 Step 3: Data enrichment

In step 3, the mashup application is implemented under the Java Spring framework, along with a popular framework called Jena. Jena is an open-source Java framework Apache project for building Semantic Web and Linked Data applications [5]. There are three main services in this mashup application listing below. After this step, **RQ1-2** is answered.

Ontology Service Ontology service is used to execute queries to get the supporting URLs from the RDF file we created in the previous step. This service is also responsible for converting the enriched data from different sources back to the RDF file, conforming to the ontology.

Repository Service Repository Service is for fetching repository and contributor-related data from the third-party code hosting platforms. Currently, we only support GitHub since most of the code bases that appeared in the ProVerB dataset are hosted by GitHub. GitHub offers a set of open REST APIs for developers under fair rate limitations. The APIs for the repository, the contributors of the specific repository, and the user detail are used in this service.

Article Service Article Service is responsible for getting the abstract, authors, and metadata of the article through a DOI link. In the ProVerB dataset, most of the related papers are published by Springer, but there are a number of papers hosted by other publishers. To deal with this situation, the API provided by CrossRef is used as a backup of Springer.

The GitHub Actions will support this enrichment step. The time schedule of this step is set after the previous conversion in section 3.2, so that the newest extracted RDF file can be used for enrichment. Some tricks are used to let the GitHub Actions automatically close this Spring-boot application after enrichment. After execution, the enriched RDF file will be released under the daily tag in GitHub Release, which will be used in RDF to ProVerB dataset conversion in section 3.6.

3.4 Step 4: Data evaluation

The data evaluation is developed as a built-in module of the integration application in the previous step. This is an engineering trade-off to reduce the workload of the development.

In this section, we will discuss how the data can be cross-evaluated in order to explore relationships between data from different sources (**RQ2**). Predefined patterns are used to identify possible linked relationships. These patterns are stored as GitHub issues with the label 'automate', conforming to an issue template we designed. This allows the ProVerB maintainers to easily direct the exploration of new relationships. The evaluation module reads these patterns and use them to generate corresponding SPARQL queries to explore the relationships. Once a relationship has been found, it will be saved into the dataset.

```
{
  "name": "Writer and CodeContributor",
  "description": "The Writer and CodeContributor...",
  "version": 1.0,
  "domain": "Writer",
  "range": "CodeContributor",
  "relationship": "sameAs",
  "patterns": [{
    "domain_property": "name",
    "range_property": "name",
    "methods": ["equal"]
  }
  ]
}
```

Fig. 5. Pattern for 'sameAs' relationship between Contributor and Writer

Since we already know the data structure of ProVerB and APIs, we can easily infer the fields which may have relationships. In this research, only the code contributors and the authors of the papers will be evaluated to inspect the possible overlaps. A pattern for this exploration can be found in Figure 5 or in this issue [1]. It means there are possible `sameAs` relationships between the **domain** class `Writer` and the **range** class `CodeContributor`, and the pattern to identify this relationship is to find the instances from these two classes whose **property** `name` is the same as the other. To maintain the quality of these relationships, only the perfect match of the names will be considered. That's why only the `equal` method is used. In this research, the template supports one **relationship** (`sameAs`) and two **methods** (`equal` and `approximate equal`).

After fetching the patterns from GitHub, the evaluation module will transform these patterns into corresponding SPARQL queries. The transformation result of the above pattern is shown in Figure 6. It's straightforward to understand this SPARQL query. It will select the instances of **domain** and **range** classes who have identical property names. After query execution, a `sameAs` relationship is added to the model for each **domain-range** pair of the query result.

After exploration, the application saves the enriched model with additional relationships into an RDF file. This RDF file is published as an artifact in the GitHub Release. These GitHub issues are left open for evaluation in the next enrichment iteration. The ProVerB maintainers can prevent the application from executing the relationship exploration patterns by closing the corresponding issues.

3.5 Step 5: Enriched RDF file

After the step of data evaluation, the additional data will be inserted into the RDF file in OWL format with optionally linked relationships. The implementation of this step is a part of the **Ontology Service** in section 3.3. This step is the answer of **RQ1-3**.

3.6 Step 6: From RDF to ProVerB

After constructing the enriched RDF file in section 3.3, a helper tool [16] will be executed to transform the triples in the RDF file back to the ProVerB dataset. This step is also under the GitHub workflow with time schedule after the enrichment and evaluation.

[1]https://github.com/LiuLiujie/ProVerBMate/issues/3

```
PREFIX  rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX  pvb:  <http://slebok.github.io/proverb/ontology#>

SELECT  ?domain ?range
WHERE
  { ?domain  rdf:type  pvb:Writer .
    ?range   rdf:type  pvb:CodeContributor .
    ?domain  pvb:name  ?dom_pro0 .
    ?range   pvb:name  ?ran_pro0
    FILTER ( ?dom_pro0 = ?ran_pro0 )
  }
```

Fig. 6. Generated SPARQL query for relationship exploration

The helper tool is also implemented using the Jena framework to query the data from the enriched RDF file, and save the data into the Markdown file of the corresponding tool in the ProVerB dataset. After conversion, a GitHub Release will be published containing the enriched ProVerB dataset.

4 RESULT

In this section, the result of executing the enrichment process will be discussed. The goal of this section is to explain the results we achieved from each step in the enrichment process and try to make a conclusion from those results for the research questions. The released artifacts and metrics of the following four steps will be discussed. Noted that the following result is based on the logs from GitHub Action on 29-06-2022 daily workflow, and the artifacts released on 29-06-2022 under tag `daily`.

An running example will also be given by tracing the enrichment steps of a verification tool called AGREE. Noted that all the figures about this example are shown in Appendix A.

4.1 ProVerB to RDF

The result of this step is related to the answer of the research question **RQ1-1**.

The daily release of the ProVerB to RDF (P2R) converter can be found here[2]. There are 425 tools in total in the ProVerB dataset, and 400 tools are extracted properly. What's more, since not all tools include supporting URLs, only 269 repository links and 525 article links are extracted from the original dataset. These links are not clean enough since pattern matching can only eliminate the data which does not follow the predefined patterns. And there are still some links that are invalid but still conform to the patterns, for example, the link pointing to GitHub organization instead of concrete repositories of the organization, or the link directing to specific branch of the repository.

As for the example tool AGREE, Figure 8 in Appendix A.1 shows the result of extraction. We can see that one repository and two related papers are extracted, along with metadata like name, basic introduction (abstract), and corresponding PV level.

4.2 Data Enrichment

The result of this step is related to the answer of research question **RQ1-2** and **RQ1-3**.

[2]https://github.com/LiuLiujie/ProVerBMate-P2RConverter/releases/tag/daily

The daily release of the enrichment can be found here[3]. During the enrichment, 256 repositories and 518 articles are enriched successfully, 1419 code contributors and 1188 paper authors are found. Of these code contributors, 1086 people have provided their names, which can be proved through the SPARQL query shown in Figure 7.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX pvb: <http://slebok.github.io/proverb/ontology#>

SELECT (COUNT(?n))
WHERE
{
   ?a rdf:type pvb:CodeContributor .
   ?a pvb:name ?n
}
```

Fig. 7. SPARQL query for the number of contributor with name

For the running example, Figure 9 and Figure 10 in Appendix A.2 show the enriched repository of AGREE and one of the contributors. The last commit date of AGREE repository is 20 May, 2022, which means this tool is still under active development. The tool AGREE has two related papers, and the enrichment results of these articles are also shown in Figure 11 and Figure 12 in Appendix A.2.

4.3 Data Evaluation

The result of data evaluation be responsible for answering the research question **RQ2**.

During the evaluation step, 273 `sameAs` relationships have been found following our predefined patterns. This number can be validated through the SPARQL query shown in Figure 13 in Appendix B.

This information can be used to determine which author may be the best one to contact for further information of the tools, since they participate in both code development and paper writing. Through the SPARQL query in Figure 14 in Appendix B, we show that 169 tools contain at least one expert who knows both the papers and the code.

4.4 RDF to ProVerB

The RDF to ProVerB transformation is the last step of this integration approach. The result of this step is not directly related to any research questions, but an evidence that the contribution **C1** and **C2** are fulfilled.

The daily release of the enrichment can be found here[4]. All the data items about repositories and related papers in the RDF file are successfully converted back to the corresponding Markdown file of the tool.

5 DISCUSSION

There are still some optimizations we can do to improve this research.

Firstly, the workflow of the enrichment step can still be optimized. In the current design, all the extracted repositories and papers will be enriched again in every iteration. However, this is unnecessary since some of the additional data items may not be changed or seldom changed once published, for example, the authors and the abstract of a paper. For these data items, enriching once is enough.

Also, more open API providers can be included in the mashup application. Currently, only GitHub API for repositories, and Springer and CrossRef API for articles are used in the application. However, we do notice that some repositories of the tools are hosted by GitLab, which provides another set of APIs for fetching information related to repositories and code contributors. And integrating their APIs may increase the success rate of repository enrichment.

Some future research projects can also be conducted based on this data integration project.

Firstly, several properties can still be enriched through this process. For example, the introduction of the application domain/field can be enriched by using third-party APIs like Wikipedia, or linking the instances in our dataset to Linked Open Dataset [18] like DBpedia [14]. Another example is adding some code snippets to the expected input and output format using GitHub search APIs, which is helpful for the developers to get started with the tools.

Moreover, as we state in section 3.1, the ontology of ProVerB is not fully utilized. A lot of further research can be conducted based on this ontology to inspect the possible matches among the tools. For example, catalog the tools based on their application domains, or explore the input-output relationships of the tools.

6 CONCLUSION

This article presented a data integration project to enrich a semi-structured dataset with additional linked data using a Semantic Web approach without breaking the original structure.

We show that 400 of 425 tools are included in this enrichment. 256 of 269 repositories and 518 of 525 articles related to these tools are enriched with their name and abstract. 1419 new code contributors of the tools and 1188 authors of the related papers are added to our dataset. By evaluating these new data, we show that at least 273 authors participate in the tool's code development and paper writing. And at least 169 tools have at least one such expert who may be the best contact person for further information. Based on these numbers, we can conclude that our research questions **RQ1** and **RQ2** are answered, and the application that should be delivered in contribution **C1** is finished.

We also show that a GitHub workflow is well configured to automatically iterate the enrichment process regularly on a schedule, since all the numbers above are fetched and analyzed from the daily release on 29-06-2022, which is published by the workflow automatically. An enriched ProVerB dataset in the original structure will also be published daily. We can conclude that the contribution **C2** is finished.

Finally, we can make a conclusion that our goals **G1** and **G2** are fulfilled by answering the two research questions **RQ1** and **RQ2** with contributions **C1** and **C2**.

[3]https://github.com/LiuLiujie/ProVerBMate/releases/tag/daily
[4]https://github.com/LiuLiujie/ProVerBMate-R2PConverter/releases/tag/daily

REFERENCES

[1] OWL Working Group 2012. *OWL - Semantic Web Standards*. OWL Working Group. Retrieved May 2, 2022 from https://www.w3.org/OWL/

[2] SPARQL Working Group 2013. *SPARQL - Semantic Web Standards*. SPARQL Working Group. Retrieved May 2, 2022 from https://www.w3.org/2001/sw/wiki/SPARQL

[3] Tim Berners-Lee, James Hendler, and Ora Lassila. 2001. The Semantic Web: A New Form of Web Content That is Meaningful to Computers Will Unleash a Revolution of New Possibilities. *ScientificAmerican.com* (05 2001).

[4] Christian Bizer, Tom Heath, and Tim Berners-Lee. 2009. Linked Data - The Story So Far. *Int. J. Semantic Web Inf. Syst.* 5, 3 (2009), 1–22. https://doi.org/10.4018/jswis.2009081901

[5] Jeremy J. Carroll, Ian Dickinson, Chris Dollin, Dave Reynolds, Andy Seaborne, and Kevin Wilkinson. 2004. Jena: Implementing the Semantic Web Recommendations. Association for Computing Machinery. https://doi.org/10.1145/1013367.1013381

[6] Jean-Marie Favre, Ralf Lämmel, Anya Helene Bagge, Tijs van der Storm, Mats Stijlaart, and Vadim Zaytsev. 2017. *SLEBoK: Software Language Engineering Body of Knowledge*. Retrieved May 2, 2022 from https://slebok.github.io

[7] Jeremy J. Carroll Graham Klyne. 2004. *Resource Description Framework (RDF): Concepts and Abstract Syntax*. W3C Recommendation. Retrieved May 2, 2022 from https://www.w3.org/TR/rdf-concepts/

[8] Ian Horrocks. 2008. Ontologies and the Semantic Web. *Commun. ACM* 51, 12 (dec 2008), 58–67. https://doi.org/10.1145/1409360.1409377

[9] Ian Horrocks, Peter F. Patel-Schneider, and Frank van Harmelen. 2003. From SHIQ and RDF to OWL: the making of a Web Ontology Language. *Journal of Web Semantics* 1, 1 (2003), 7–26. https://doi.org/10.1016/j.websem.2003.07.001

[10] A.K. Kalou, Dimitrios Koutsomitropoulos, and Georgia Solomou. 2016. Combining the Best of Both Worlds: A Semantic Web Book Mashup as a Linked Data Service Over CMS Infrastructure. *Journal of Library Metadata* 16 (10 2016), 0. https://doi.org/10.1080/19386389.2016.1258897

[11] Agnes Koschmider, Victoria Torres, and Vicente Pelechano. 2009. Elucidating the mashup hype: Definition, challenges, methodical guide and tools for mashups. (01 2009).

[12] Sophie Lathouwers and Vadim Zaytsev. 2021. *ProVerB: Program Verification Book*. Retrieved May 2, 2022 from https://slebok.github.io/proverb/index.html

[13] Sophie Lathouwers and Vadim Zaytsev. 2022. Modelling Program Verification Tools for Software Engineers. In *Proceedings of the ACM/IEEE 25th International Conference on Model Driven Engineering Languages and Systems (MoDELS)*, Eugene Syriani, Houari Sahraoui, and Nelly Bencomo (Eds.). IEEE. In print.

[14] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. 2015. DBpedia - A large-scale, multilingual knowledge base extracted from Wikipedia. *Semantic Web* 6, 2 (2015), 167–195. https://doi.org/10.3233/SW-140134

[15] Yujie Liu. 2022. *LiuLiujie/ProVerBMate-P2RConverter*. Retrieved Jun 19, 2022 from https://github.com/LiuLiujie/ProVerBMate-P2RConverter

[16] Yujie Liu. 2022. *LiuLiujie/ProVerBMate-R2PConverter*. Retrieved Jun 20, 2022 from https://github.com/LiuLiujie/ProVerBMate-R2PConverter

[17] Yujie Liu. 2022. *ProVerB-Ontology/ProVerB_1.4.0.owl*. Retrieved Jun 19, 2022 from https://github.com/LiuLiujie/ProVerB-Ontology/blob/main/ProVerB_1.4.0.owl

[18] John P. McCrae. 2022. *The Linked Open Data Cloud*. Retrieved Jun 21, 2022 from https://lod-cloud.net

[19] Mark A. Musen. 2015. The protégé project: a look back and a look forward. *AI Matters* 1, 4 (2015), 4–12. https://doi.org/10.1145/2757001.2757003

[20] Lixian Ni, Zhuoming Xu, Ting Wu, and Wenjie He. 2013. Visualizing Linked Data with JavaScript. In *2013 10th Web Information System and Application Conference*. 211–216. https://doi.org/10.1109/WISA.2013.48

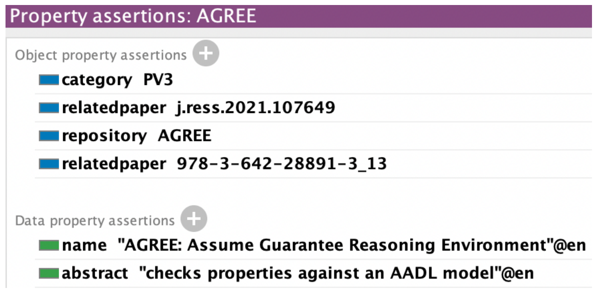
[21] Damires Souza, Bernadette Farias Lóscio, and Ana Carolina Salgado. 2012. Combining semantic information and information quality on the enrichment of Web Data Integration Systems. *WEBIST 2012 - Proceedings of the 8th International Conference on Web Information Systems and Technologies* (2012), 219 – 224. https://doi.org/10.5220/0003961602190224

[22] Joachim Van Herwegen, Pieter Heyvaert, Ruben Taelman, Ben De Meester, and Anastasia Dimou. 2018. Knowledge Representation as Linked Data: Tutorial. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management* (Torino, Italy) *(CIKM '18)*. Association for Computing Machinery, New York, NY, USA, 2299–2300. https://doi.org/10.1145/3269206.3274275

[23] José Ortiz Vivar, José Segarra, Boris Villazón-Terrazas, and Víctor Saquicela. 2022. REDI: Towards knowledge graph-powered scholarly information management and research networking. *Journal of Information Science* 48, 2 (2022), 167–181. https://doi.org/10.1177/0165551520944351

A ENRICHMENT EXAMPLE AGREE

A.1 Data Extraction

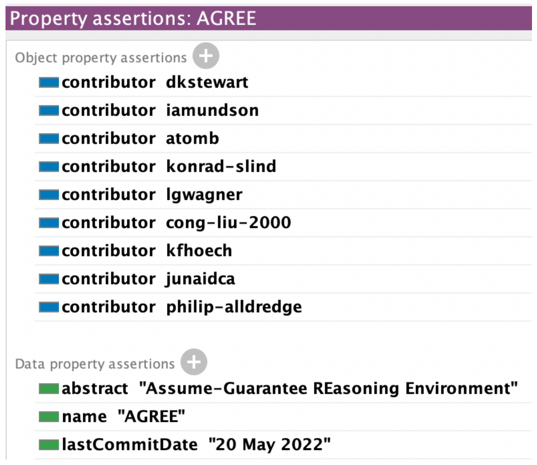


category PV3
relatedpaper j.ress.2021.107649
repository AGREE
relatedpaper 978-3-642-28891-3_13

name "AGREE: Assume Guarantee Reasoning Environment"@en
abstract "checks properties against an AADL model"@en

Fig. 8. Data extraction

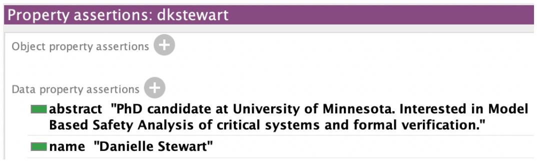
A.2 Data Enrichment



contributor dkstewart
contributor iamundson
contributor atomb
contributor konrad-slind
contributor lgwagner
contributor cong-liu-2000
contributor kfhoech
contributor junaidca
contributor philip-alldredge

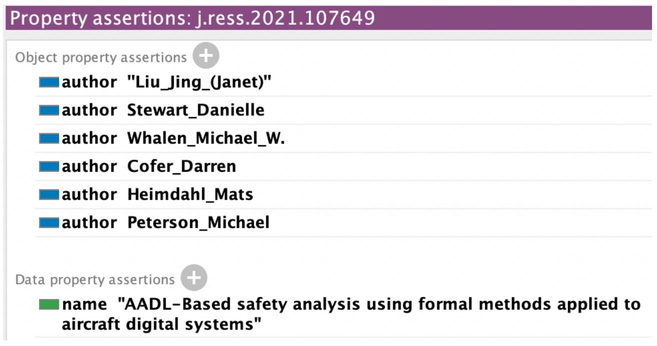
abstract "Assume-Guarantee REasoning Environment"
name "AGREE"
lastCommitDate "20 May 2022"

Fig. 9. Enriched AGREE repository



abstract "PhD candidate at University of Minnesota. Interested in Model Based Safety Analysis of critical systems and formal verification."
name "Danielle Stewart"

Fig. 10. Enriched one of the contributors of AGREE



Property assertions: j.ress.2021.107649

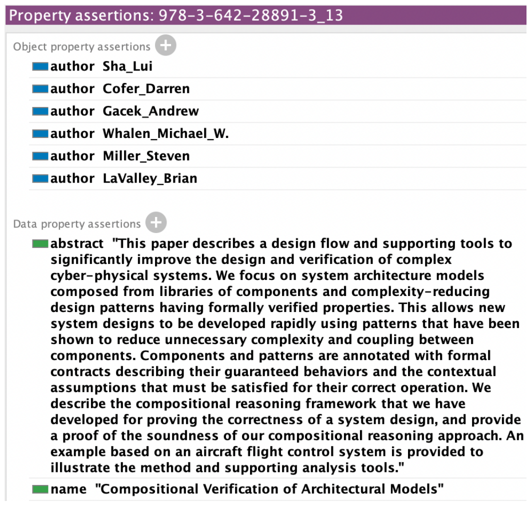
Object property assertions

- author "Liu_Jing_(Janet)"
- author Stewart_Danielle
- author Whalen_Michael_W.
- author Cofer_Darren
- author Heimdahl_Mats
- author Peterson_Michael

Data property assertions

- name "AADL-Based safety analysis using formal methods applied to aircraft digital systems"

Fig. 11. Enriched article 1



Property assertions: 978-3-642-28891-3_13

Object property assertions

- author Sha_Lui
- author Cofer_Darren
- author Gacek_Andrew
- author Whalen_Michael_W.
- author Miller_Steven
- author LaValley_Brian

Data property assertions

- abstract "This paper describes a design flow and supporting tools to significantly improve the design and verification of complex cyber-physical systems. We focus on system architecture models composed from libraries of components and complexity-reducing design patterns having formally verified properties. This allows new system designs to be developed rapidly using patterns that have been shown to reduce unnecessary complexity and coupling between components. Components and patterns are annotated with formal contracts describing their guaranteed behaviors and the contextual assumptions that must be satisfied for their correct operation. We describe the compositional reasoning framework that we have developed for proving the correctness of a system design, and provide a proof of the soundness of our compositional reasoning approach. An example based on an aircraft flight control system is provided to illustrate the method and supporting analysis tools."
- name "Compositional Verification of Architectural Models"

Fig. 12. Enriched article 2

B SUPPORTING SPARQL QUERIES

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX pvb: <http://slebok.github.io/proverb/ontology#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>

SELECT COUNT(?contributor)
WHERE
{
   ?contributor rdf:type pvb:CodeContributor .
   ?writer rdf:type pvb:Writer .
   ?writer owl:sameAs ?contributor
}
```

Fig. 13. SPARQL query for the number of sameAs relationships

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX pvb: <http://slebok.github.io/proverb/ontology#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>

SELECT COUNT(DISTINCT ?tool)
WHERE
{
   ?tool rdf:type pvb:Tool .
   ?contributor rdf:type pvb:CodeContributor .
   ?writer rdf:type pvb:Writer .
   ?writer owl:sameAs ?contributor .
   ?tool pvb:repository ?repo .
   ?repo pvb:contributor ?contributor .
   ?tool pvb:relatedpaper ?paper .
   ?paper pvb:author ?writer
}
```

Fig. 14. SPARQL query for the number of tools with at least one expert