Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

A multi-facets ontology matching approach for generating PLC domain knowledge graphs

Yameng An * **Feiwei Qin** ** **Danfeng Sun** *** **Huifeng Wu** ****

* *Hangzhou Dianzi University, Hangzhou, China (e-mail: anyamg@pm.me).*
** *Hangzhou Dianzi University, Hangzhou, China (e-mail: qinfeiwei@hdu.edu.cn)*
*** *Institut f. Automation und Kommunikation e.V., Magdeburg, Germany, (e-mail: danfeng.sun@ifak.eu)*
**** *Hangzhou Dianzi University, Hangzhou, China (e-mail: whf@hdu.edu.cn)*

Abstract: Programmable Logic Controller (PLC) has been playing an important role in industrial automation. Users want to improve programming efficiency by implementing code reuse and more intelligent code retrieval. Due to the heterogeneity of different PLC development environments, it is then necessary to design a computable knowledge model to semantically represent, organize, and utilize these diversified resources. Using the ontology technique is a common way to achieve the interoperability of heterogeneous systems. Aiming at this, we propose an ontology matching approach in this paper. Knowledge extraction and alignment are difficult for most of the knowledge graphs construction tasks, however, we are able to build the PLC domain knowledge graph with high accuracy and completeness by considering PLC domain characteristics, designing layered ontology, and implementing the matching process primarily on schema level instead of instance level.

Keywords: Software reuse, Semantic representation, Ontology matching, Knowledge graph

1. INTRODUCTION

PLC has been widely used in industrial automation to improve system reliability, stability, and performance. It is used for logic control in all kinds of industrial, such as transportation and manufacturing, which requires complicated automatic processes. In addition, according to Wu et al. (2018), PLC has motion control abilities which could be used to simplify the control system. Currently, there are several PLC development platforms independent from hardware and manufacture widely used, such as CODESYS, Beremiz, and MULTIPROG. Since the release of IEC 61131-3 Programming languages standard, users want to be able to exchange their PLC programs and libraries between heterogeneous environments. Despite all software vendors claim that their products conform to the IEC 61131-3 standard, they still have their own knowledge structure and are not compatible with each other. It is usually required to rewrite the PLC projects when transferring from one software platform to another. Not to mention share and exchange information on the semantic layer. According to TC 65/SC 65B (2019), now the IEC committee publishes the IEC 61131-10 PLC open XML exchange format based on the work of PLCopen TC6 - XML. If an IEC 61131-3 project is saved in this XML format, it could be reused by any other development environment who supports this standard XML format. But this standard only provides a common XML format for representing the IEC 61131-3 software models and languages, without providing a mechanism to support semantic information between different PLC programming environments. The help for code reuse is limited since all software platforms are heterogeneous and have their own implementation of PLC controllers, along with different terminologies. Not to mention vendor-specific data-types and controller libraries.

To support PLC code reuse and retrieval on the semantic level, a unified semantic description of heterogeneous PLC projects is needed. Ji et al. (2015) show that with the help of knowledge representation technology, particularly the knowledge graph, it is then possible to achieve code retrieval on semantic and further knowledge-driven software design. As is introduced in Ehrlinger and Wöß (2016), a knowledge graph acquires and integrates information into an ontology and applies a reasoner to derive new knowledge. Knowledge graph has been widely used in many areas. For instance, Alibaba knowledge graph in E-commerce area mentioned in Wang et al. (2018), Links Life Data in the medical domain discussed in Momtchev et al. (2009), and Kensho in financial scope by Kensho Technologies LLC (2013). And this technology is now increasingly used in the automation domain. Terkaj et al. (2017) provide a modular ontology of the building control and automation domain for reusing building data (e.g., design data, product data, and sensor data) across various formats. Ryabinin et al. (2019) put forward an ontology-driven approach to automate the firmware and middleware

generation for IoT devices. Yang et al. (2016) propose a method for the automatic generation of smart-grid automation system software based on ontology transformation. Dai et al. (2013) propose a system design approach using ontology as knowledge base. This approach is based on IEC 61499 function blocks. However, the succeeded application of knowledge graph in industry control, especially in the PLC domain, is still limited.

Though using knowledge graph raises heterogeneity problems to a higher level, conflicting ontologies in the same domain usually cannot interoperate with each other. Shvaiko and Euzenat (2008) propose that ontology matching is a solution to the semantic heterogeneity problem. Ontology matching detects correspondences between semantically related entities of ontologies, which further can be used for ontology merging, query answering, data translation, etc. Euzenat et al. (2007) show that matching ontologies enables the knowledge and data expressed in the matched ontologies interoperate with each other. In this paper, we provide an ontology matching based domain knowledge graph automatic construction approach, aiming to transfer and uniformly represent the PLC projects without much additional effort. The specific contributions of this paper are as follows.

- We build a PLC domain knowledge graph to describe semantic information. Based on knowledge representation and ontology techniques, we could use the reasoning mechanism and inference tool to exchange data from heterogeneous information silos.
- The construction of the knowledge graph uses a layered approach. The core schema represents the most common concepts in the PLC domain while the outer layers' schemas integrate semantics in manufacture area, application area, etc.
- A multi-facets ontology matching approach is applied to align heterogeneous terms. This approach implements on the schema level based on the common core ontology.
- Use of semantic knowledge to enable code reuse between heterogeneous PLC platforms.

2. PLC DOMAIN KNOWLEDGE GRAPH CONSTRUCTION

The construction of the PLC domain knowledge graphs mainly involves knowledge modeling, knowledge extraction, ontology matching, and other technologies, as is shown in Fig. 1. A usual way of building domain knowledge graph is to build a unified schema. But this leads to the difficulty of entity recognition and linking. Also, the accuracy and preciseness of natural language do not meet the requirements of the industry. Thus, we propose a layered method, which builds a core ontology first, then create a separate ontology for each software platform. This has two main advantages. First, the difficulty of entity recognition is reduced since the semantic structure and composition of PLC projects are relatively consistent with its development platform. The second is that such construction has better versatility and extensibility. We construct a Common Core Ontology (CCO) to represent common concepts in PLC projects. And for heterogeneous PLC development environment, code comments, test logs, and bug reports,

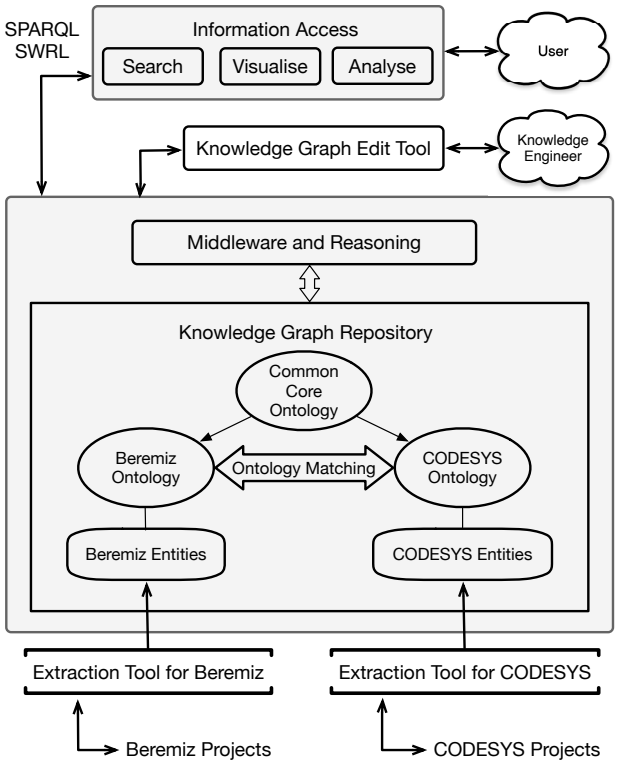

Fig. 1. PLC domain knowledge graph architecture. The knowledge graph repository is in a layer hierarchy, which consists a Common Core Ontology (CCO) layer, an Application Specific Ontology (ASO) layer and a instance layer.

we use Application Specific Ontology (ASO) to express layered concepts and data. The definitions of CCO and ASO are introduced as follows.

Definition 1. (Common Core Ontology). A CCO is a tuple $< C, R, D, A >$, where

- C is a set of concepts which represent types of object, specifically the definitions of PLC program elements concepts in the XML format and program hierarchy between these concepts.
- R is a set of semantic relations between concepts or instances (e.g., subClassOf, partOf).
- D is a set of datatypes.
- A is a set of axioms that constrain the possible interpretation for the defined terms.

Definition 2. (Application Specific Ontology). An ASO is a tuple $< C, R, D, A, I >$, where

- I is a set of instances of corresponding concepts which indicate specific instance in the real world.

C, R, D and A are the same as those in CCO.

Leveraging the constructed PLC domain knowledge graph, we propose a method in the following section for code reuse and retrieval by ontology matching. The matching operation determines the alignment A of the source ontology O and the target ontology O'. Alignment expresses the correspondences between concepts of different ontologies. Correspondence is the relation hold between two concepts of different ontologies. The definitions of correspondence and alignment are presented as follows.

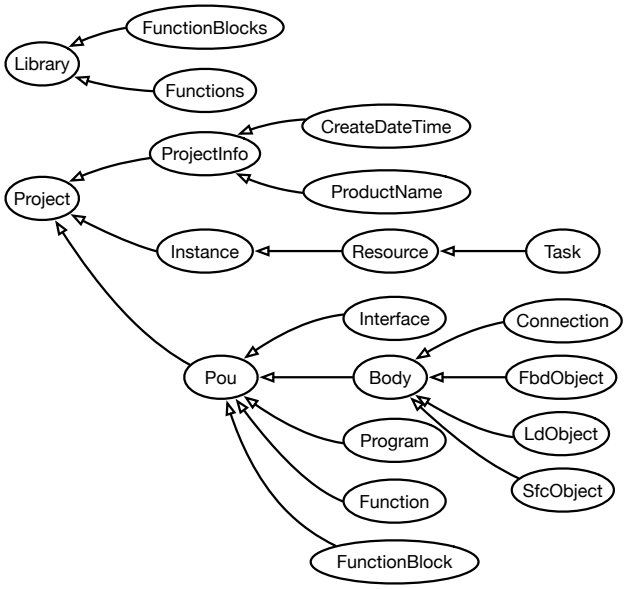

Fig. 2. Common Core Ontology (CCO) of the PLC domain knowledge graphs (partly).

Definition 3. (Correspondence). Given two ontologies O and O', a correspondence between two ontologies O and O' is a tuple $< id, C, C', M, R >$, where

- id is a unique identifier of the correspondence.
- C and C' are concepts of the first and the second ontology respectively.
- M is a confidence measure in range $[0, 1]$ holding for the correspondence between the concepts C and C'. The higher the confidence, the higher the likelihood of a relationship.
- R is a relation holding between the concepts C and C' (e.g., equivalence($=$), subset of($\subseteq$), superset of($\supseteq$)).

Definition 4. (Alignment). Given two ontologies O and O', an alignment between two ontologies O and O' is *Align*, where

- *Align* is a set of correspondences between concepts belongs to O and O'.
- The correspondence might be one to one, one to more, or more to one.

We construct and apply the PLC domain knowledge graphs according to the following steps. According to our observation, PLC programs usually have a certain pattern, which makes entity extraction simpler than unstructured plain-text.

(1) Build CCO for the PLC domain.
(2) Build ASO for each development platform respectively.
(3) Knowledge extraction of PLC programs from multiple heterogeneous software platforms.
(4) Knowledge alignment based on ontology matching.

As a result, a snippet of CCO is shown in Fig. 2. Here we use Beremiz as an example, and present part of the ASO of Beremiz in Fig. 3. Beremiz is an open-source IEC 61131-3 integrated development environment developed by Tisserant et al. (2007). Compared to other development environments, Beremiz has several features of its own. First, Beremiz supports high-level programming languages C and Python as POU (Program Organisation Unit) types. Second, Beremiz has its own controllers library with a variety of platform-specific *Functions* and *FunctionBlocks*. These *Functions* and *FunctionBlocks* cannot be export and import to other development platforms. One reason is that other platforms may not have the same *Function* or *FunctionBlock*. The other reason is terminological heterogeneity, which means other platforms may use other names for the same *Function* or *FunctionBlock*.

3. ONTOLOGY MATCHING IMPLEMENTATION

A computable knowledge model is constructed leveraging ontology matching which can be used to reduce semantic heterogeneity. In PLC domain, semantic heterogeneity mainly refers to terminological heterogeneity in controllers libraries in different PLC development environments. Terminological heterogeneity occurs due to variations in names when indicating the same concepts in different development environments. This is usually caused by the use of abbreviation, capitalization, and affix. To remedy this kind of heterogeneity, name-based ontology alignment methods proposed in Cohen et al. (2003) and Euzenat et al. (2004) could be considered. However, the alignment result by using name-based techniques alone is not good enough. If the confidence measure is set high to ensure higher precision, the recall will be significantly low. This is because many concept names in PLC domain are abbreviated, and concepts with different affixes in source ontology usually should be mapped to the same concept in target ontology. For example, Beremiz has *JumpStep* and *Step*, which equals to *Jump* and *Step* in CODESYS respectively. Name-based techniques are not able to match *JumpStep* to *Jump* effectively since the confidence measure between *JumpStep* and *Jump* is roughly equal to the confidence measure between *JumpStep* and *Step*. Moreover, simply adding the relation factor into consideration is not helpful enough. In the given example, *JumpStep* in Beremiz ontology is the subclass of SFC, while *Jump* in CODESYS ontology is the subclass of concept *General*.

To solve this, we propose an ontology matching approach which builds correspondence on schema level instead of instance level, according to the characteristics of PLC domain. The CCO acts as an intermediate ontology, which could be used as a reference to determine the relevance of the concepts in source and target ontologies. Through multi-facets ontology matching which considers names, data properties, object properties, paths, etc., the correspondence between different PLC development platforms (e.g. Beremiz, CODESYS) and projects are established. After that, the instances under corresponding concepts are recognized and aligned automatically.

3.1 Similarity Measure

The similarity measure is used to calculate the similarity between the concepts of different ontologies. The difficulty of similarity measure is that different PLC vendors may not only use different terms to represent PLC controllers but also use different structures to organize them. Therefore, it is necessary to compute the similarity from multiple perspectives to implement accurate measurement.

Fig. 3. Application Specific Ontology (ASO) of Beremiz (partly). Beremiz supports high-level programming language C and Python as Pou types, and has its own controllers library with some platform-specific *Functions* and *FunctionBlocks*.

Concept names are compared to compute string similarity. Internal structure, which is the definition of concepts without references to other concepts (Euzenat et al. (2007)), specifically the properties and cardinality of the concepts are used to calculate the internal structure similarity. The relational structure is ignored since the position and the relationship of a concept in the knowledge architecture do not affect its role. Similarity should be normalized if it ranges over the unit interval of real numbers $[0, 1]$. A normalized similarity σ is a function from a pair of concepts to a real number ranging in $[0, 1]$ expressing the similarity between two concepts, as in:

$$\begin{aligned}\sigma(c_i, c_j) = & \, sim\_string(c_i, c_j) \\ & \sqcap sim\_internal\_structure(c_i, c_j) \\ & \sqcap sim\_datatype(c_i, c_j)\end{aligned} \tag{1}$$

According to Guerrini et al. (2007), having two ontologies C and C' containing k_1 and k_2 concepts respectively, the similarity between them can be expressed as the average similarity of their elements. It is represented in (2), where $\sigma(c_i, c_j)$ is the similarity between concepts $c_i \in C$ and $c_j \in C'$ computed by (1).

$$Sim(C, C') = \frac{\Sigma_{i=1}^{k_1} \Sigma_{j=1}^{k_2} \sigma(c_i, c_j)}{k_1 + k_2} \tag{2}$$

3.2 Multi-facets Ontology Matching Method

The ontology matching algorithm between CCO and ASO utilizes both name-based techniques and structure-based techniques. Considering that similar names are used to represent the same semantic concept, name comparison is firstly applied to determine the similarity of two concepts. Besides, the internal structure of concepts is compared to ensure the correctness of similarity measure and to find more correspondence. The flowchart of ontology matching is shown in Fig. 4, and the specific steps are as follows.

Preprocessing Remove vendor prefixes, blank characters, and connecting characters like "_". Then convert each alphabetic character in the concept names into their lower case counterpart.

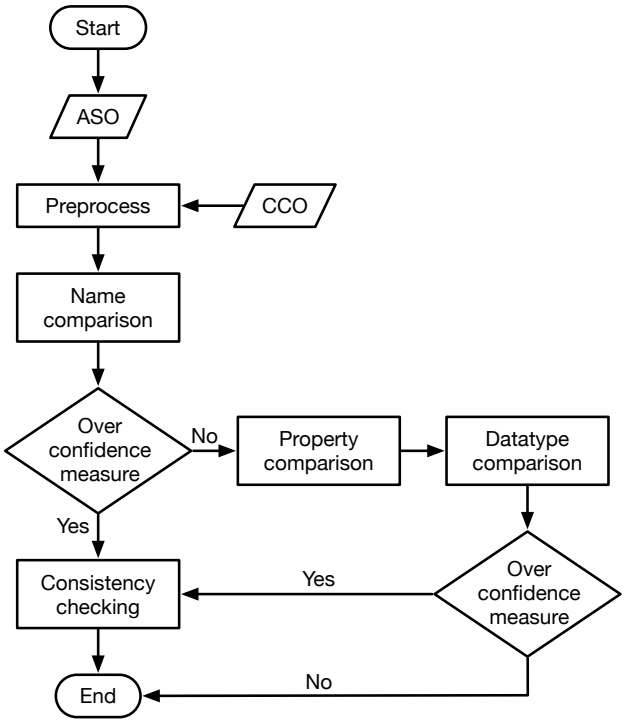

Fig. 4. Flowchart of ontology matching algorithm.

Name comparison Edit distance representing the minimal cost of transfer a string to the other one is designed to measure the similarity between two strings. The Levenshtein distance, an edit distance algorithm provided in Levenshtein (1966), which calculates the minimum number of insertions, deletion, and substitution of characters required to transfer one string to another, is applied for name comparison. The string similarity is calculated as in 3, where $lev(c_i, c_j)$ represents the Levenshtein distance between the names of two concepts c_i and c_j respectively.

$$sim\_string(c_i, c_j) = 1 - \frac{lev(c_i, c_j)}{|c_i|+|c_j|} \tag{3}$$

Property comparison Properties represent relationship. For a *FunctionBlock*, it would typically be the *instance name*, for a *Function* it can be its *input variable*and *output variable*. Such information could be used to identify a concept. If different concepts provide highly similar properties, it is plausible to determine that they are equivalent. The quantification of property similarity is in 4, c_{i_p} and c_{j_p} represents the properties of c_i and c_j respectively.

$$sim\_internal\_structure(c_i, c_j) = \frac{|c_{i_p} \vee c_{j_p}| - |c_{i_p} \wedge c_{j_p}|}{|c_{i_p} \vee c_{j_p}|} \tag{4}$$

Datatype comparison Datatype can be part of the relation applied to the property of the concept in onology, which is corresponding to PLC's data types. Data types are not disjoint in PLC domain. Intuitively, the similarity between datatypes should be highest when they are of the same type, relatively higher similarity when they are compatible, relatively lower when they are not compatible. Part of the data type categories compatibility table in PLC domain is listed in table 1. Similarity between specific data types within its category, for example, *SINT*, *DINT*, and *ULINT* in *INTEGER*, is determined as 0.9.

Table 1. PLC data type compatibility (partly)

	Bit Strings	INTEGER	REAL	Character
Bit Strings	1.0	0.3	0.4	0.8
INTEGER	0.3	1.0	0.6	0.3
REAL	0.4	0.6	1.0	0.3
Duration	0.1	0.2	0.2	0.1
Date	0.1	0.2	0.2	0.1
Time of day	0.1	0.2	0.2	0.1
Character	0.8	0.3	0.3	1.0

3.3 Reasoning

After ontology matching, it is possible to process ontology merging based on the matching results by reasoning. Reasoning involves applying the alignment as reasoning rules of the two ontologies O and O', which is represented as follows.

$$Reason(O, O', TransformToRules(Align)) = O'' \tag{5}$$

The transformed rules can be written in ontology language rules such as OWL (Web Ontology Language) or SWRL (Semantic Web Rule Language). A reasoner Pellet is used to perform consistency checking on the merged ontology in this paper.

4. CASE STUDY ON MAPPING BEREMIZ AND CODESYS

In this section, we implement the proposed multi-facets ontology matching approach integrated with COMA++, which is developed by Aumueller et al. (2005). COMA++ provides an extensible library of matching algorithms that could be used for composite matching strategies. To demonstrate our approach, ontology matching between Beremiz and CODESYS is provided as an example. Protégé, an ontology editor developed by Musen (2015), is used to create Beremiz ontology, CODESYS ontology, and the core ontology.

Part of the mapping results of Beremiz ontology and CODESYS ontology is given based on the strategy illustrated in the previous section in Fig. 5.

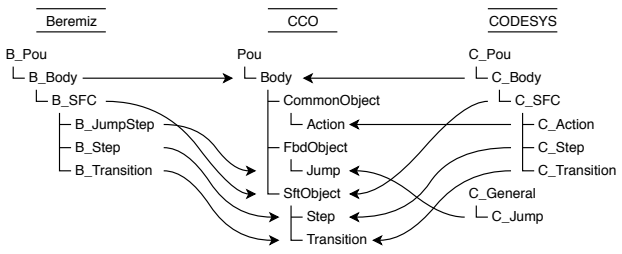

Fig. 5. Ontology matching results on Beremiz ontology and CODESYS ontology (partly).

In this case, Beremiz ontology contains 276 items and CODESYS ontology contains 196 items. As a result, 67 out

of 69 equivalent items are mapped. In practice, the ASOs containing PLC data in different levels and scenarios are distinct. Two most common ASO types are PLC software ontology and bug report ontology. The matching system adapts well to all these situations and it produces merged results with high accuracy in about a second.

5. CONCLUSION

This paper proposes a novel multi-facets ontology matching approach for generating PLC domain knowledge graphs. Specifically, the layered knowledge graphs are constructed with CCO and ASO to represent heterogeneous PLC development environments. Semantic information in PLC programs is extracted as entities of corresponding development platforms. We rely on schema-level input information for performing ontology matching. The similarity between concepts of different ontologies is quantified according to their names, relations, and datatypes etc. Ontology merging is processed based on the matching results. We demonstrate our approach by applying it to mapping Beremiz ontology and CODESYS ontology.

In the future, we will continue with semantic reasoning, retrieval, and analysis by using techniques and tools such as SWRL and SPARQL. Furthermore, we will implement semantic retrieval and recommendation of PLC code and function blocks and analyze the code structure to evaluate code quality.

ACKNOWLEDGEMENT

This research work was supported in part by National Natural Science Foundation of China (No. 61972121).

REFERENCES

Aumueller, D., Do, H.H., Massmann, S., and Rahm, E. (2005). Schema and ontology matching with COMA++. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, 906–908. ACM.

Cohen, W., Ravikumar, P., and Fienberg, S. (2003). A comparison of string metrics for matching names and records. In *Kdd workshop on data cleaning and object consolidation*, volume 3, 73–78.

Dai, W., Dubinin, V.N., and Vyatkin, V. (2013). Migration from PLC to IEC 61499 using semantic web technologies. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 44(3), 277–291.

Ehrlinger, L. and Wöß, W. (2016). Towards a definition of knowledge graphs. *SEMANTiCS (Posters, Demos, SuCCESS)*, 48.

Euzenat, J., Shvaiko, P., et al. (2007). *Ontology matching*, volume 18. Springer.

Euzenat, J., Valtchev, P., et al. (2004). Similarity-based ontology alignment in OWL-lite. In *ECAI*, volume 16, 333.

Guerrini, G., Mesiti, M., and Sanz, I. (2007). An overviewof similarity measures for clustering XML documents. In *Web data management practices: emerging techniques and technologies*, 56–78. IGI Global.

Ji, G., He, S., Xu, L., Liu, K., and Zhao, J. (2015). Knowledge graph embedding via dynamic mapping matrix. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, 687–696.

Kensho Technologies LLC (2013). Kensho. `https://www.kensho.com`. Accessed: 2020-03-01.

Levenshtein, V.I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, 707–710.

Momtchev, V., Peychev, D., Primov, T., and Georgiev, G. (2009). Expanding the pathway and interaction knowledge in linked life data. *Proc. of International Semantic Web Challenge*.

Musen, M.A. (2015). The protégé project: a look back and a look forward. *AI Matters*, 1(4), 4–12.

Ryabinin, K., Chuprina, S., and Belousov, K. (2019). Ontology-driven automation of IoT-based human-machine interfaces development. In *International Conference on Computational Science*, 110–124. Springer.

Shvaiko, P. and Euzenat, J. (2008). Ten challenges for ontology matching. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, 1164–1182. Springer.

TC 65/SC 65B (2019). Programmable controllers - Part 10: PLC open XML exchange format. Standard, International Electrotechnical Commission.

Terkaj, W., Schneider, G.F., and Pauwels, P. (2017). Reusing domain ontologies in linked building data: the case of building automation and control. In *8th International Workshop on Formal Ontologies meet Industry*, volume 2050.

Tisserant, E., Bessard, L., and de Sousa, M. (2007). An open source IEC 61131-3 integrated development environment. In *2007 5th IEEE International Conference on Industrial Informatics*, volume 1, 183–187. IEEE.

Wang, J., Huang, P., Zhao, H., Zhang, Z., Zhao, B., and Lee, D.L. (2018). Billion-scale commodity embedding for e-commerce recommendation in alibaba. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 839–848.

Wu, H., Yan, Y., Sun, D., and Simon, R. (2018). A customized real-time compilation for motion control in embedded PLCs. *IEEE Transactions on Industrial Informatics*, 15(2), 812–821.

Yang, C.W., Dubinin, V., and Vyatkin, V. (2016). Ontology driven approach to generate distributed automation control from substation automation design. *IEEE Transactions on Industrial Informatics*, 13(2), 668–679.