Representing Technical Standards as Knowledge Graph to Guide the Design of Industrial Systems

Jose Illescas, Georg Buchgeher(✉), Lisa Ehrlinger, David Gabauer, and Jorge Martinez-Gil

Software Competence Center Hagenberg GmbH (SCCH), Hagenberg, Austria
{Jose.Illescas,Georg.Buchgeher,Lisa.Ehrlinger, David.Gabauer,Jorge.Martinez-Gil}@scch.at
http://www.scch.at

Abstract. Technical standards help software architects to identify relevant requirements and to facilitate system certification, i.e., to systematically assess whether a system meets critical requirements in fields like security, safety, or interoperability. Despite their usefulness, standards typically remain vague on how requirements should be addressed via solutions like patterns or reference architectures. Thus, software architecture design remains a time-consuming human-centered process.

In this work, we propose an approach on how to use knowledge graphs for supporting software architects in the design of complex industrial systems. We discuss how project-generic knowledge (e.g., technical standards) and project-specific knowledge like the description of a concrete system can be modeled as knowledge graph. Making the architectural knowledge, which is currently present in technical standards and other resources, machine-readable, enables the support of the software architect through expert systems and therefore, improve the quality of the overall system design. However, since architectural knowledge is currently presented in many different formats, the transformation to a uniform, machine-readable form is required. We demonstrate the applicability of our approach with a representative example of an industrial client-server architecture and outline research challenges for future work.

Keywords: Knowledge graph · Ontology · Technical standard · System architecture · Architecture design · Architecture evaluation

1 Introduction

The design of industrial hardware and software systems involves expert knowledge of a broad spectrum of areas and fields, considering application domain knowledge (over general software engineering activities), system kind-specific application, and technology-specific expertise, among others. As systems evolve, the importance of meeting their quality (and functional) requirements, such as safety, security, integrity, maintainability, etc., increases because they determine the foundation of the system [5]. Nevertheless, in practice, many of the fundamental quality requirements are not addressed due to the lack of knowledge, time, or expertise [1].

Thus, software architects are confronted with more demanding activities during the architecture design process as presented in [11] and have to work with many kinds of knowledge. Architectural knowledge generalizes explicit and implicit reusable knowledge, which can vary in form, type, degree of formality, etc., such as reference architectures, architectural styles, patterns, design patterns, technical standards, or guidelines.

Technical standards, developed and maintained by networks of international institutions (such as ISO, IEC, and ETSI) together with national organizations (such as NIST, DIN, and ANSI), contain relevant knowledge to be considered during the software architecture design as they reflect a global consensus of people with expertise in their subject of matter and who know the needs of the organizations they represent [2,13,14]. A technical standard contains a set of requirements and recommendations that are relevant for specific kinds of systems in a domain or for certain processes. To claim conformance to a standard all requirements defined by the standard must be fulfilled. Technical standards support software architects to identify relevant requirements and facilitate system certification. However, the adherence to such standards can be challenging, e.g., due to the vagueness of how to address provisions, or the expected familiarity with the standard that the architect requires to use it.

Recent advancements in artificial intelligence (AI) allow the development of novel kinds of systems and the automation of knowledge-intensive activities that previously had to be carried out manually by humans. Thus, the software engineering community researches how software engineering can be supported with AI-based technologies. Knowledge graphs (KGs) are an emerging technology for the development of explainable AI applications. KGs are used for semantically modelling a complex domain [9,19], and use reasoning- and AI-based methods for the development of knowledge-based systems like question-answering (QA) [15], decision support systems [18] and recommendation systems [7].

This work explores how KGs can be used to support software architects in designing complex industrial systems. The remainder of the paper is distributed as follows: we present our approach to build a KG for supporting software architects considering project-specific and generic knowledge in Sect. 2. Section 3 presents a representative example during the architecture design process supported by the KG and automated reasoning. Section 4 presents a short overview of the related work. Finally, we present research challenges and conclusions.

2 Approach

Architecture design is an incremental and iterative process, which makes it an exhaustive human-centered, resource-, time- and knowledge-intensive consuming activity [8]. Since software architects have a great impact on the quality of the system [4], it is important to support them with relevant knowledge that needs to be considered during architecture design and evaluation. Reusable architecture knowledge is therefore used to validate if the requirements have been met.

Such automated support would improve not only the quality, but the access of reusable knowledge gained by experience of the architect as well, two key factors of the continuous software evolution and development phases.

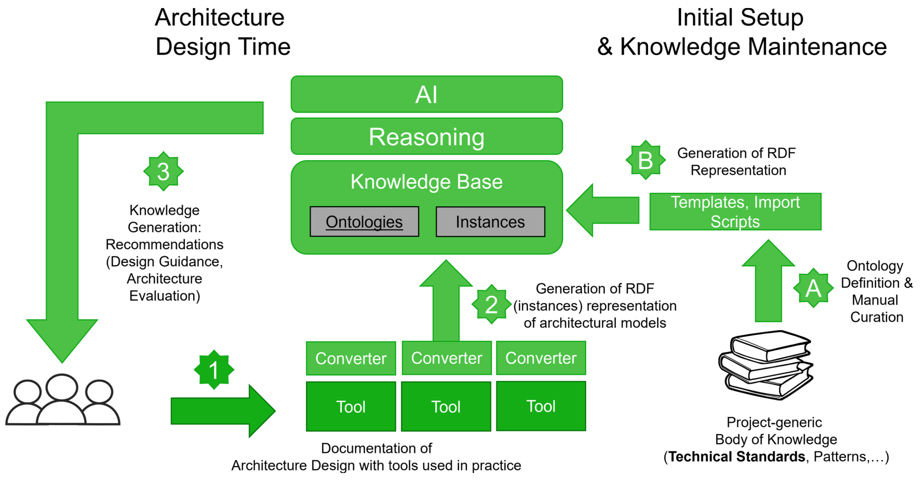
Architecture Design Time
Initial Setup & Knowledge Maintenance
AI
Reasoning
Knowledge Base
Ontologies
Instances
3
Knowledge Generation: Recommendations (Design Guidance, Architecture Evaluation)
B
Generation of RDF Representation
Templates, Import Scripts
A
Ontology Definition & Manual Curation
2
Generation of RDF (instances) representation of architectural models
Converter
Converter
Converter
Tool
Tool
Tool
1
Documentation of Architecture Design with tools used in practice
Project-generic Body of Knowledge (**Technical Standards**, Patterns,...)

Fig. 1. Approach of the system for assisting and validating design decisions.

Figure 1 illustrates our approach of building KGs based on existing project-generic knowledge (such as technical standards) to support the software architecture design phase. As mentioned in [6], a KG is built by two main components, a knowledge base consisting of ontologies, describing the domain, and a reasoning component, which is used to perform inference and exploit non-obvious relations and derive new knowledge. Thus, several base ontologies are defined in a first step, e.g., a generic schema for technical standards which covers their main structure, relationships and main characteristics.

Software architects work with many kinds of architectural knowledge and tools which have different degrees of abstraction. The knowledge can be either project-generic or specific, tacit or explicit, and so forth. Certain kind of knowledge is also prone to the phenomena denoted vaporization of knowledge. Nevertheless, the implementation of such a KG, would support the main concept of knowledge management by capturing, sharing, using and reusing it, which also conforms greatly to the definition of an ontology.

On top of the KG, we intend to exploit reasoning and AI-based methods to automate the design process to find recommendations, alternatives, disregarded provisions that might be of great importance and be able to find and react to conflicting approaches, etc. Thus, enabling the KG to support the software architect during the architecture design process by providing design guidance and automated architecture evaluation. As part of design guidance, context-dependent relevant architecture knowledge is derived and proposed. During automated architecture evaluation a candidate architecture solution structure is analyzed, e.g., to identify not or incorrectly addressed requirements and to suggest potential improvements for an existing architecture design.

3 Exemplary Application to Industrial Control System

This chapter shows how to apply our approach from Sect. 2 to a client-server communication in an industrial setting, i.e., an industrial control system (ICS) that provides a remote maintenance API. The example considers only the security context of the system's communication, which can be evaluated to determine how secure it is. A technical standard whose scope considers cyber-security requirements, could provide provisions that can be used to recommend and achieve a more secure communication, thus improving the security of the system.

On the one hand, we are provided with a system component structure, on the other hand we use a technical standard addressing cyber-security for components of industrial automation and control systems IACS. The system component structure, shown in Fig. 2, is comprised of two components, namely, the ICS component acting as the server and the remote maintenance component, which is a web application acting as the client. The ICS component provides a service port for accessing maintenance data. The remote maintenance component has a reference port for communicating with the port provided by the ICS system component. Both ports use a dedicated API, i.e., the *Remote Maintenance API* and support the communication protocols HTTP and HTTPS. The two components communicate over HTTP.

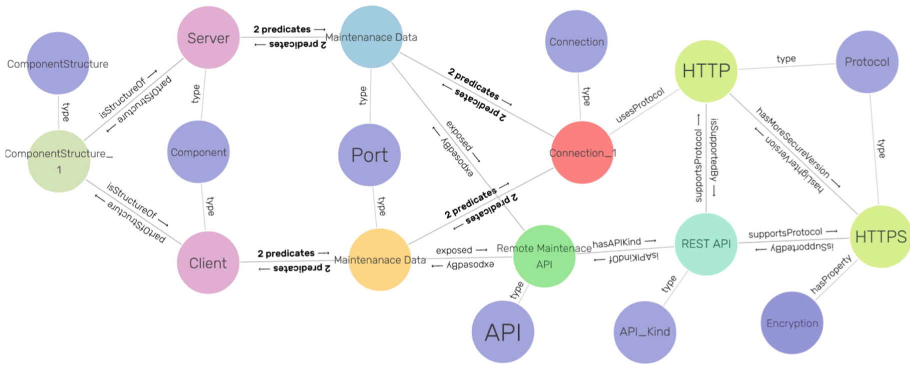

Fig. 2. Representation of the initial setup of the application (GraphDB visualization).

The IEC 62443, a standard series that address cyber-security for operational technology in automation and control systems, is considered. More specifically, in this example we refer to the IEC 62443-4-2 standard, titled Security for industrial automation and control systems - Part 4–2: Technical security requirements for IACS components, which will be further addressed as the base standard. The base standard provides a set of cyber security requirements for the components that compose an IACS. The document specifies security capabilities of those components with regards to the ability of mitigating security threats without the assistance of compensating countermeasures [12]. Among the relevant requirements we identified the following:

CR 3.1 Communication integrity. Capability to protect transmitted information, as the data being transmitted is a common target of manipulation.
CR 3.1 (Enhancement) Communication authentication. Components should be capable of verifying the authenticity of the received information.
CR 4.1 Information confidentiality. Components should be able to guarantee protection of confidentiality of data at rest and/or in transit.

Despite the simplicity of the example, more specific information is needed to answer questions like *Is the system solution tamper-proof?*, with respect to the data being transmitted. This question will be referred to as the example question. Notoriously, concepts such as *Components*, *Ports*, *API*, and their relationships, among others are needed to define system solutions, technical standards and their content. Those concepts and relationships are distributed in multiple ontologies that cover project-generic and project-specific information. The ontologies are classified as reusable solutions, system base, type specific instance ontologies, etc. For this example, the following base ontologies of our KG are used and further exploited by a reasoning engine that uses the OWL2 QL rule set:

Component Structures: Contains concepts relevant for component solution structures such as components, referenced and provided ports, exposed API and their respective attributes.
Standards Schema: A generic construct to define standards, provisions, their categories, enhancements and other kinds of supplemental information.
IEC 62443 Instances: Concrete instantiation of the base standard.
Protocol Instances: Instances of protocols with complementary properties and information.

We define an ontology based on the example description, mapping the description entities to the KG concepts. Then the system can check whether the proposed component solution has a partial conformance to the base standard. With partial conformance, we limit the scope to the three requirements mentioned above with respect to the data in transit.

To demonstrate the approach, we consider two scenarios. First, HTTP is used as communication protocol, which is not considered safe and does not conform to the base standard. In the second scenario, HTTPS is used. HTTPS is transmitted over TLS, which makes it a more secure, encrypted and robust variant of HTTP. Hence, a better recommendation can be proposed. The connection, as stated above, supports both protocols. This is a simple but relevant evaluation and assessment achieved with the help of the KG. After considering the suggestion of using HTTPS, the answer of the same example question shows a positive result, see Table 1.

Table 1. Result of the automated security evaluation for the two scenarios

ReferencePort	API	Protocol	isSafe
Maintenance data	Remote maintenance API	HTTP	FALSE
Maintenance data	Remote maintenance API	HTTPS	TRUE

The exemplary application in this section shows how a design decision can be supported by requirements obtained from a technical standard and how it can be complemented with more data.

4 Background and Related Work

There have been studies which expose the challenges that architects face due to the heterogeneity of the knowledge, as shown in [16], which analyzes how efficiently search engines can find AK to solve specific architectural tasks and stress how challenging the process of finding the required AK is.

Moreover, due to the growing amount of technical standards as an effect of Industry 4.0 (I4.0), the importance of technical KGs, based on technical content, to support engineering design activities is critical [10]. In [2] a KG, referred to as I40KG, for I4.0 related standards, norms, and reference frameworks is proposed. I40KG is intended to support newcomers, experts, and other stakeholders in understanding how to implement I4.0 systems by providing a Linked Data-conform collection of annotated, classified reference guidelines. The need to make such relevant knowledge accessible to system stakeholders and the semantic definition of standards and their relationships is considered. Nevertheless, our approach goes a step beyond. Besides classifying them and making their relationships clear, we want to use their content, especially the provisions and complementary information, to provide guidance and assistance during architectural evaluation, where a given system solution can be further analyzed and backed up by standards.

Furthermore, the application of AI methods in software engineering has been an exciting topic, as shown in [3] where a systematic review is presented. The different kinds and methods of AI applied at different stages of software engineering are reviewed with the conclusion that AI has successfully optimized many tasks related to the development cycle of software engineering. In [2], AI was used to create embeddings and find similarities between standards. However, we want to analyze to which extent we can use AI methods to explore and make more accessible the architectural knowledge silos within the KG.

Finally, [17] proposed a KG for bug resolution, showing some interesting results because the bug KG can provide more accurate and comprehensive information related to a bug issue. Due to the highly explainable content, we consider a KG for assisting software architects in a more complex process such as the design of industrial systems.

5 Conclusion

In this paper, we explained the idea of modeling a KG that covers project-generic knowledge, e.g., technical standards and project-specific knowledge for supporting software architects. Making architectural knowledge present in technical standards and other sources accessible in a more automated fashion can improve the quality of the industrial control systems as the architectural design process would be less demanding for such critical stakeholders.

Most of the available architectural knowledge is presented in natural language, thus, Natural Language Processing methods are required to automatically transform this data to a machine-readable form (e.g., a KG). The benefit of using KGs for representing architectural knowledge is the high expressiveness in contrast to classic data models (e.g., relational data model) [9]. Therefore, the highly explainable content of the KGs can support software architects during human-centric tasks of the architectural design process. The use of explainable AI like KGs is able to clarify the recommendations and analyses rather than relying on purely abstract black-box models.

Tools that interact with KGs in a more natural way are needed. Then arbitrary system context-relevant questions can be placed without the need of a query language like SPARQL. Such a demanding human-centered activity should support a flexible query composition mechanism to make the retrieval of required knowledge less demanding and more accessible.

Finally, defining a more robust and complete architectural knowledge domain to improve systems quality is essential. However, full automation might not be feasible. Manual curation, pre-, and post-preparation of the models and data are still required to improve the results.

Acknowledgements. This work was supported in part by the Interreg Österreich-Bayern 2014–2020 Programme funded under Grant (AB292) and in part by the FFG BRIDGE project AK-Graph (grant no. 883718).

References

1. Assal, H., Chiasson, S.: Security in the software development lifecycle. In: Proceedings of the Fourteenth USENIX Conference on Usable Privacy and Security, SOUPS 2018, pp. 281–296. USENIX Association, USA (2018)
2. Bader, S.R., Grangel-Gonzalez, I., Nanjappa, P., Vidal, M.-E., Maleshkova, M.: A knowledge graph for industry 4.0. In: Harth, A., et al. (eds.) ESWC 2020. LNCS, vol. 12123, pp. 465–480. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-49461-2_27

3. Barenkamp, M., Rebstadt, J., Thomas, O.: Applications of AI in classical software engineering. AI Perspect. **2**(1), 1–15 (2020). https://doi.org/10.1186/s42467-020-00005-4
4. Capilla, R., Jansen, A., Tang, A., Avgeriou, P., Babar, M.A.: 10 years of software architecture knowledge management: practice and future. J. Syst. Softw. **116**, 191–205 (2016). https://doi.org/10.1016/j.jss.2015.08.054
5. Doukidis, G., Spinellis, D., Ebert, C.: Digital transformation - a primer for practitioners. IEEE Softw. **37**(05), 13–21 (2020). https://doi.org/10.1109/MS.2020.2999969
6. Ehrlinger, L., Wöß, W.: Towards a definition of knowledge graphs. In: SEMANTiCS (Posters, Demos, SuCCESS) (2016)
7. Engleitner, N., Kreiner, W., Schwarz, N., Kopetzky, T., Ehrlinger, L.: Knowledge graph embeddings for news article tag recommendation (2021). https://doi.org/10.13140/RG.2.2.12602.52161
8. Farshidi, S., Jansen, S., van der Werf, J.M.: Capturing software architecture knowledge for pattern-driven design. J. Syst. Softw. **169**, 110714 (2020). https://doi.org/10.1016/j.jss.2020.110714
9. Feilmayr, C., Wöß, W.: An analysis of ontologies and their success factors for application to business. Data Knowl. Eng. **101**, 1–23 (2016). https://doi.org/10.1016/j.datak.2015.11.003
10. Han, J., Sarica, S., Shi, F., Luo, J.: Semantic networks for engineering design: state of the art and future directions. J. Mech. Des. **144**(2) (2021). https://doi.org/10.1115/1.4052148
11. Hofmeister, C., Kruchten, P., Nord, R.L., Obbink, H., Ran, A., America, P.: A general model of software architecture design derived from five industrial approaches. J. Syst. Softw. **80**(1), 106–126 (2007). https://doi.org/10.1016/j.jss.2006.05.024
12. Security for industrial automation and control systems - part 4–2: Technical security requirements for iacs components. Standard, International Electrotechnical Commission (2019)
13. International Electrotechnical Commission: Understanding standards. https://iec.ch/understanding-standards
14. International Organization for Standarization: Standards. https://www.iso.org/standards.html
15. Lukovnikov, D., Fischer, A., Lehmann, J., Auer, S.: Neural network-based question answering over knowledge graphs on word and character level. In: Proceedings of the 26th International Conference on World Wide Web, pp. 1211–1220. International World Wide Web Conferences Steering Committee (2017). https://doi.org/10.1145/3038912.3052675
16. Soliman, M., Wiese, M., Li, Y., Riebisch, M., Avgeriou, P.: Exploring web search engines to find architectural knowledge (2021)
17. Wang, L., Sun, X., Wang, J., Duan, Y., Li, B.: Construct bug knowledge graph for bug resolution. In: 2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C), pp. 189–191 (2017). https://doi.org/10.1109/ICSE-C.2017.102
18. Wang, X., He, X., Cao, Y., Liu, M., Chua, T.S.: Kgat: knowledge graph attention network for recommendation. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, pp. 950–958. Association for Computing Machinery, New York (2019). https://doi.org/10.1145/3292500.3330989
19. Yahya, M., Breslin, J.G., Ali, M.I.: Semantic web and knowledge graphs for industry 4.0. Appl. Sci. **11**(11) (2021). https://doi.org/10.3390/app11115110