Aligning openCDE APIs with Linked Building Data through Constrained Containers in Common Data Environments

Oliver Schulz[1,*], Jakob Beetz[1]

[1]Design Computation, Department of Architecture, RWTH Aachen University, Aachen, Germany

Abstract

Common Data Environments (CDEs) serve as container-based web platforms, facilitating shared data storage for all project stakeholders and simplifying communication between them. Building on the DIN SPEC 91391 and ISO 19650 standards, the openCDE APIs are developed by buildingSMART to provide a unified interface across the various providers of CDEs. This research aims to investigate integrating the container-based CDE approach and the openCDE APIs with the concepts of Linked Building Data to establish deeper connections between the buildings' datasets. In particular, this paper examines how data can be stored in a Linked Data-based container environment to remain interoperable with existing schemas and APIs. The Linked Data Platform (LDP) specification, in conjunction with the Shapes Constraint Language (SHACL), is examined to tailor containers for specific use cases and schemas in the CDE ecosystem. This work should demonstrate how constraints can enforce specialised containers on the LDP. The process is illustrated using a subset of openCDE APIs - the BIM Collaboration Format (BCF) API - as an example. Our findings highlight that the LDP specification offers the necessary functionalities to specialise containers to the respective needs, but the specification does not explain how this mechanism shall be enforced. Consequently, this work is a step toward aligning the industry's approaches with the openCDE APIs and the current scientific concepts of Linked Building Data.

Keywords

SHACL, Common Data Environments, Linked Data Platform, Linked Building Data, openCDE APIs

1. Introduction

Common Data Environments (CDEs) are used in the industry and are investigated in research as platforms that provide project stakeholders with a single source of information [1]. They bring structure to construction projects and make communicating between stakeholders easier [2]. Considerable effort is made to define and regulate these concepts of CDEs in the standards of ISO 19650 [3] and the DIN SPEC 91391 [1]. To avoid isolated solutions, these standards aim to make these platforms accessible to other applications and developers so that data gathered throughout the life cycle of the buildings can be leveraged.

From a technical point of view, buildingSMART International is offering a collection of application programming interfaces (APIs) - called openCDE APIs [4] - that cover some of the functionalities of these CDEs and make them accessible. The currently available APIs enjoy

LDAC 2024: 12th Linked Data in Architecture and Construction Workshop, June 13–14, 2024, Bochum, Germany

*Corresponding author.

schulz@dc.rwth-aachen.de (O. Schulz); beetz@dc.rwth-aachen.de (J. Beetz)

0000-0002-4722-4621 (O. Schulz); 0000-0002-9975-9206 (J. Beetz)

great popularity and are widely used throughout the industry. From a research perspective, we can also see a rising increase in interconnecting data and breaking the data silos of single applications and providers. This applies to the realm of CDEs and, generally, the linking of building-related data. For example, the Linked Building Data (LBD) community has set itself the task of investigating and defining Linked Data appliances for building life cycles[1].

While Linked Data and its underlying technology are promising approaches to solving interconnectivity issues, front-end developers are often unfamiliar with them [5], making it difficult for them to integrate them into a production environment. More so, domain experts in the built environment and building owners who are the end users of CDEs cannot be expected to master the complexity behind Linked Data and utilise it appropriately.

While other research focuses on the manifestation of the concepts of CDEs (ISO 19650 and DIN SPEC 91391) and their containerisation, this paper investigates how to connect the widely adopted technology of buildingSMART's openCDE APIs with the concepts of Linked Data and its underlying container concepts. As covering all CDE functionalities in this paper is out of scope, we will illustrate our approach using the Issue Management process as a primary use case. Issue Management is commonly used in the construction industry to identify and communicate *Issues* - such as clashes of or missing properties in building elements - during, but not exclusively, the design phase of a building. While there are several vendor-specific options for conducting Issue Management, the BIM Collaboration Format (BCF) API is one of the most developed and prominent processes covered by the openCDE APIs.

This research builds on the author's previous work, which dealt specifically with the topics of BCF and CDEs and their application with Linked Data [6, 7]. This work is limited to verifying incoming data for storage in a web container for compatibility with buildingSMARTs specifications. However, it does not address how the client communicates the data to the final destination in a container. Therefore, this should lead the way to connecting the advances from both industry and research without disrupting achievements in one or the other.

The paper is structured as follows. The subsequent section (Section 2) provides background information on related work and research in the Architecture Engineering and Construction (AEC) industry. Section 3 then examines how constraints can be imposed on containers, applies this approach to the general framework of CDEs, and further specifies this approach using the CDE subset Issue Management. The following section (Section 4) discusses how this approach can be technically implemented in the communication between the server and the client. The work is concluded in Section 5, and future work is outlined.

```
@PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@PREFIX ldp: <http://www.w3.org/ns/ldp#> .
@PREFIX bcfOWL: <http://lbd.arch.rwth-aachen.de/bcfOWL#> .
@PREFIX sh: <http://www.w3.org/ns/shacl#> .
```

Listing 1: Prefixes used throughout this paper.

[1]Linked Building Data Community Group, W3C: https://www.w3.org/community/lbd/ accessed 17.04.2024

2. Related Work

2.1. Information Container

Information container is a term in the AEC industry that describes how files reside in a web environment or a file system. Although several definitions of this term exist, they generally all describe the same concept. DIN SPEC 91301 describes them as "[...] the smallest unit for a file or a model and logical construct for file or model management within a CDE" [1]. According to the standard, these can be nested and also contain multi-models. ISO 19650 describes them as "[...] named persistent set of information retrievable from within a file, system or application storage hierarchy" [3]. The following two subsections present two technical implementations that can be considered information containers.

2.1.1. Information Container for Linked Document Delivery

The Information Container for Linked Document Delivery (ICDD) is an ISO-standardised (ZIP) Container for exchanging heterogeneous files between different stakeholders while allowing for establishing and communicating links between the documents or elements inside the documents [8, 9]. It was originally designed with the use cases of information requests and deliveries in the AEC industry in mind. ICDD containers are files themselves and are exchanged between the different project stakeholders. ICDD is based on the Resource Description Framework (RDF) and has two base ontologies, one for container[2] and one for linksets[3]. It requires the following structure:

- Index Document: Contains general information about the container and its contents.
- Ontology Container: This contains all ontologies necessary to interpret the RDF data in the ICDD. Dereferenceable ontologies from the web do not necessarily need to be stored in the folder.
- Payload Container: Contains all documents in a folder structure.
- Payload Triples Container: Contains the links for the documents to internal and external resources

In AEC, researchers investigate, for example, how to link RDF with non-RDF data for document delivery [10] or how to publish ICDD container on CDEs based on the DIN SPECs 91391 openCDE API [11, 12].

2.1.2. Linked Data Platform

The Linked Data Platform (LDP) is a World Wide Web Consortium (W3C) recommendation that provides a structure to serve (Linked Data) resources via the Hypertext Transfer Protocol (HTTP) on the web [13]. The platform specification is published in various documents, including examples, best practices and recommendations [14, 15, 13]. An ontology for the specification

[2]Container Ontology: https://standards.iso.org/iso/21597/-1/ed-1/en/Container.rdf accessed 07.02.2024
[3]Linkset Ontology: https://standards.iso.org/iso/21597/-1/ed-1/en/Linkset.rdf accessed 07.02.2024

is published on the web[4]. LDPs are based on URLs, meaning that a dereferenceable URI shall name every entity, and entities can point to other dereferenceable URIs. Therefore, following links to traverse through the platforms using LDP is possible. The core structure of LDP comes down to two main concepts:

- Resources (*ldp:Resource*) represent entities on the web
- Containers are specialised resources that shall contain other resources.

On the LDP, RDF is used to describe the metadata of the Containers and Resources, even though the specific contents of a resource may be in RDF (*ldp:RDFsource*) or a native format (*ldp:NonRDFsource*). Besides this, LDP is not enforcing any rigid structure on where to store the data. Oversimplified, it is comparable to a file explorer, whereas the containers are similar to folders and the resources to files. Several implementations of LDP exist, while one of the most current ones - Solid [16] - is loosely based on it.

LDP is used in AEC-related research, for example, to federate CDEs [17, 18, 6], by using the LDP implementation of Solid, or by combining the DIN SPECs 91391 openCDE API and ICDD with the LDP in [11].

2.2. Linked Data Validation

Utilising Linked Data, it is important to have a mechanism that can validate incoming data that shall be added to a graph. A validation process can ensure that the data conforms to specific standards and project requirements. While it is theoretically possible to incorporate restrictions into the ontologies with the Web Ontology Language (OWL), these are not necessarily usable in validation because of the open-world assumption[5]. Although SPARQL could validate the data, this leads to complex and impractical queries[19]. Therefore, the Shapes and Constraint Language (SHACL) [20] - a W3C recommendation - was developed to validate RDF. The restrictions are defined in a concept called Shapes, which themselves are also serialised in RDF and - like ontologies - can be published on the web. For a more in-depth discussion of validation, the reader is referred to [21]. SHACL is frequently used in the AEC industry and academia. For example, SHACL is used in [22] to check building legislation and in [23] to define scheduling constraints. Furthermore, Hagedorn et al. discuss how to validate containers in ICDD with SHACL [24]. Finally, [25] suggests a way to define SHACL shapes via Visual Programming, which should improve usability and enable the approaches to non-linked data experts.

2.3. Open CDE APIs

The openCDE APIs are an initiative of buildingSMART to provide standardised APIs to communicate with CDE providers. They are not to be confused with the openCDE API described in the DIN SPEC 91391 [26]. The current collection of APIs contains:

[4]The W3C Linked Data Platform (LDP) Vocabulary, W3C: https://www.w3.org/ns/ldp accessed 12.02.2024
[5]Web Ontology Language (OWL) Guide, W3C: https://www.w3.org/TR/2004/REC-owl-guide-20040210/ accessed 17.04.2024

- The Foundation API as a common ground for the different APIs to cover authentication and users[6].
- The BCF API for communicating issues occurring in the (digital) building between different stakeholders and applications[7].
- The Documents API for uploading and downloading files that reside on a CDE[8].

Furthermore, buildingSMART plans to develop additional APIs for CDEs, like an interface for "[...] data-driven object exchange [...]" [4]. The topic of the BCF API already has connections with the area of Linked Data. In [27] and [7], an ontology is proposed that maps the schema of BCF to Linked Data, and in [18] and [6] a framework for federated CDEs is proposed on the example of BCF.

3. Constraining the Container

For this paper, the Linked Data Platform was chosen to store and constrain the data. The platform's architecture is already tailored towards the web and is, therefore, a suitable candidate for Common Data Environments. In a basic setup of a container in LDP (*ldp:Container*), it simply lists all its containing resources (Listing 2),

```
<.../ldp/Container_1> a ldp:Container;
    ldp:contains <Resource1>, <Resource2>, <Resource3> .
<Resource1> a ldp:Resource .
```

Listing 2: A default Container listing its Resources.

but LDP also comes with the property of *ldp:hasMemberRelation*, which can be used to point out the class of the resources a container contains (Listing 3) when using the concept of a specialised ldp:DirectContainer. This, e.g., is a filter mechanism when discovering data on the platform. If containers are equipped with this property, it can be determined immediately when discovering the platform whether a container contains the requested data or not. Even though this is already narrowing down what should be expected from the resources when looking at the container, it is not enforcing that the resources stick to a specific schema or structure.

```
<.../ldp/Container_1> a ldp:DirectContainer;
    ldp:hasMemberRelation ex:hasExampleClass ;
    ldp:contains <Resource1>, <Resource2>, <Resource3> .
<.../ldp/container_1/#it> a rdfs:Resource;
    ex:hasExampleClass <Resource1> .
<Resource1> a ldp:Resource, ex:ExampleClass .
```

Listing 3: A container that specified what kind of Resources reside in it.

[6]buildingSMART Foundation API: https://github.com/BuildingSMART/foundation-API/tree/v1.0 accessed: 13.02.2024
[7]buildingSMART BCF API: https://github.com/buildingSMART/BCF-API accessed: 13.02.2024
[8]buildingSMART Documents API: https://github.com/buildingSMART/documents-API accessed 13.02.2024

This indicates that the property is mainly used for exploring data on a platform rather than creating new data that must adhere to the specialisation of the corresponding *ldp:Container*. To enforce more specific schemas for *ldp:Container*, *ldp:constrainedBy* can be employed, as shown in (Listing 4). In LDP, it is defined that this property can point to any *rdfs:Resource*. It is, therefore, not limited to what a constraint mechanism can be used for it. For this paper, it was decided to use SHACL since this is - as LDP - a W3C recommendation.

```
<.../ldp/Container_1> a ldp:Container;
    ldp:constrainedBy ex:ExampleClassShape ;
    ldp:contains <Resource1>, <Resource2>, <Resource3> .
<Resource1> a ldp:Resource, ex:ExampleClass .
<.../ldp/Constraints/SimpleConstraint> a sh:NodeShape ;
    sh:targetClass ex:ExampleClass ;
    sh:property [
    ...
    ] .
```

Listing 4: An extension of the container with specific constraints and a basic SHACL shape.

Nevertheless, since LDP just describes an architecture for a platform based on Linked Data on the web [13], it does not describe how this constraint shall be enforced by the platform. The functionality has to be either integrated into an application that uses the LDP architecture or could be used by a middleware that checks for these constraints before it posts to an *ldp:Container*.

3.1. Application with CDEs

In the previous section, we covered a basic architecture on applying SHACL shapes on the LDP to constrain what *ldp:Resource* can be added to an *ldp:Container*. In this section, the approach is taken one step further and applied to the concept of a CDE to check data that is uploaded to the CDE for conformity to either project internal or external constraints.

An internal constraint can be, for example, an agreement between the different project stakeholders on specific naming conventions or labels that shall be used when creating new issues in the process of clash detection. These individual requirements and conventions in AEC projects are often defined in the Exchange Information Requirements (EIR) and the BIM Execution Plan (BEP) and vary from project to project. Although it is beyond the scope of this work, a machine-readable and directly translatable conversion of these documents into constraints - e.g. by functionalities as suggested in [25] - seems desirable.

External constraints, on the other hand, such as industry-wide standards like BCF, can define specific formatting that the data must adhere to in order to be considered compliant or compatible with them. Contrary to the internal constraints, these are consistent throughout the projects. New versions of the standards are exceptions, however, and should be made available under a new URL.

Therefore, the constraints of a CDE based on LDP should be established in two ways. On the one hand, the stakeholders define their own project's internal constraints. On the other

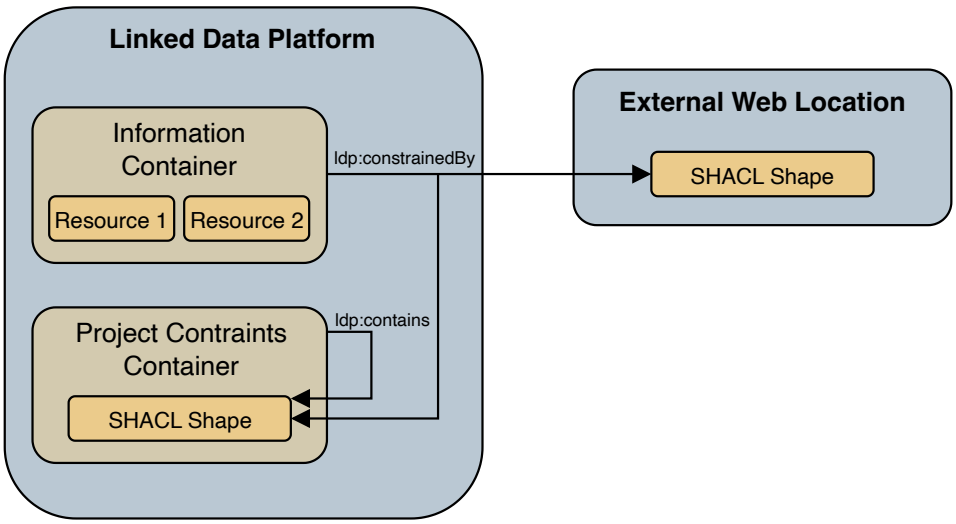

Figure 1: Conceptual representation of where the various constraints are located.

hand, there must be publicly available external constraints - preferably compatible with the FAIR principles [28] - that guarantee uniformity across projects and applications (Figure 1). An ldp:Container in the CDE context can, therefore, be used with both internal and external constraints at the same time, resulting in containers with multiple constraints (Listing 5).

3.2. Applied to Issue Management

In this section, we apply the approach to the subset of CDEs with the field of Issue Management in the form of buildingSMART's BCF.

```
<.../cde/project_1/BCF/TopicsContainer/> a ldp:Container;
    ldp:constrainedBy externalShapes:TopicShape projectShapes:TopicShape;
    ldp:contains <Topic1> .
<Topic1> a bcfOWL:Topic ;
    bcfOWL:hasTopicStatus "Active" ;
    bcfOWL:hasTitle "A generic topic title" ;
    bcfOWL:hasLabel project:Architecture .
```

Listing 5: Example BCF Topic Container with shortened content.

The *Topic* container itself is constrained by an external and an internal shape (Listing 5). The external shape (Listing 6) is a universally applicable BCF shape that ensures compatibility with buildingSMART's specifications. It is not responsible for checking the specific values of the *bcfOWL:Topic* but ensures that the structure is correct. For a BCF *Topic*, the minimum requirement is that it contains at least a title. Other properties such as *bcfOWL:hasTopicStatus* can either not be present in a *Topic* or only occur once.

```
<...external-url/TopicShape>  a sh:NodeShape;
    sh:targetClass bcfOWL:Topic ;
    sh:property [
        sh:path bcfOWL:hasTitle ;
        sh:minCount 1;
        sh:maxCount 1;
    ];
    sh:property [
        sh:path bcfOWL:hasTopicStatus;
        sh:maxCount 1;
    ] .
```

Listing 6: External shape requiring a title for each Topic and allowing only one Status.

The internal shapes are based on the project agreements. There, it can, for example, specify what terms should be used for defining if a *Topic's* status is still open or what values are acceptable for *bcfOWL:hasLabel*. This can be checked with shapes as seen in Listing 7. While APIs often work with datatypes like strings, dates and numbers, we can leverage more advanced properties in Linked Data and allow meta descriptions of objects. Therefore, our example shows the potential uses on the one hand with a string for the *Status* and on the other hand with an object consisting of a URI for the *Labels*. How this is implemented in the end depends on the project, but it should generally be ensured that the values specified as URIs are dereferenceable.

```
<.../cde/project_1/constraints/TopicShape> a sh:NodeShape ;
    sh:targetClass bcfOWL:Topic ;
    sh:property [
        sh:path bcfOWL:hasTopicStatus ;
        sh:in ("Resolved" "Active" "Closed") ;
     ] ;
    sh:property [
        sh:path bcfOWL:hasLabel ;
        sh:in (project:Architecure project:MEP project:Documentation) ;
    ] .
```

Listing 7: Internal SHACL shape used to constrain the Status and the Labels.

Another example is the *Comment*, which is not constrained by any project-specific values in BCF. But it is constrained because it needs to have a link to a specific *Topic* and either a link to a *bcfOWL:Viewpoint*, a textual comment, or both. This can be expressed in SHACL, as seen in Listing 8 and would be defined in an external universal shape.

```
<...external-url/CommentShape> a sh:NodeShape ;
sh:targetClass bcfOWL:Comment ;
  sh:or (
    [
       sh:path bcfOWL:hasCommentText ;
       sh:minCount 1 ;
    ]
    [
       sh:path bcfOWL:hasViewpoint ;
       sh:minCount 1 ;
    ]
  ) .
```

Listing 8: Example for a shape for BCF Comments.

4. Conceptual Implementation

This section describes the theoretical scenario in which a client wants to create a new *Topic* with the status "Active". The data should be validated against the shapes from the previous section. Therefore, we envision an infrastructure consisting of the clients sending the *Topic* (POST), the LDP for storing the Issue Management data, and a middleware that acts as an intermediary for negotiating the communication between these agents. The middleware is responsible for checking the *Topic* requests against the constraints specified in the LDP container. Suppose the client wants to POST *Topic* data to the LDP. In that case, the middleware checks the container at the location of the route for the predicate *ldp:constrainedBy*, and if such a constraint exists, the middleware fetches the constraints - in our case, the SHACL shapes from Listing 6 and 7 - and checks the data against them. The constraints may reside either on the LDP itself and are based on project agreements or on more general standards that, e.g. check for conformity with a specific format like BCF.

In our scenario, the constraints are violated because the string "Active" is not allowed as a status in this project and "Open" shall be used. Therefore, the middleware rejects the POST with an HTTP 400 and an error message. The client should now exchange "Active" with "Open" and send the request again. The server and the middleware should now respond with an HTTP 201 and the content of the newly created *Topic*. The process is depicted in Figure 2.

5. Conclusion

This paper explores how information containers can host CDE-specific data following buildingSMART's openCDE APIs. First, different container approaches were investigated, and the W3C recommendation LDP was decided on since it reflects the structure of the web and describes resources as dereferenceable, which allows data accessibility. Subsequently, it was demonstrated through the example of Issue Management in the form of BCF. As CDEs are meant

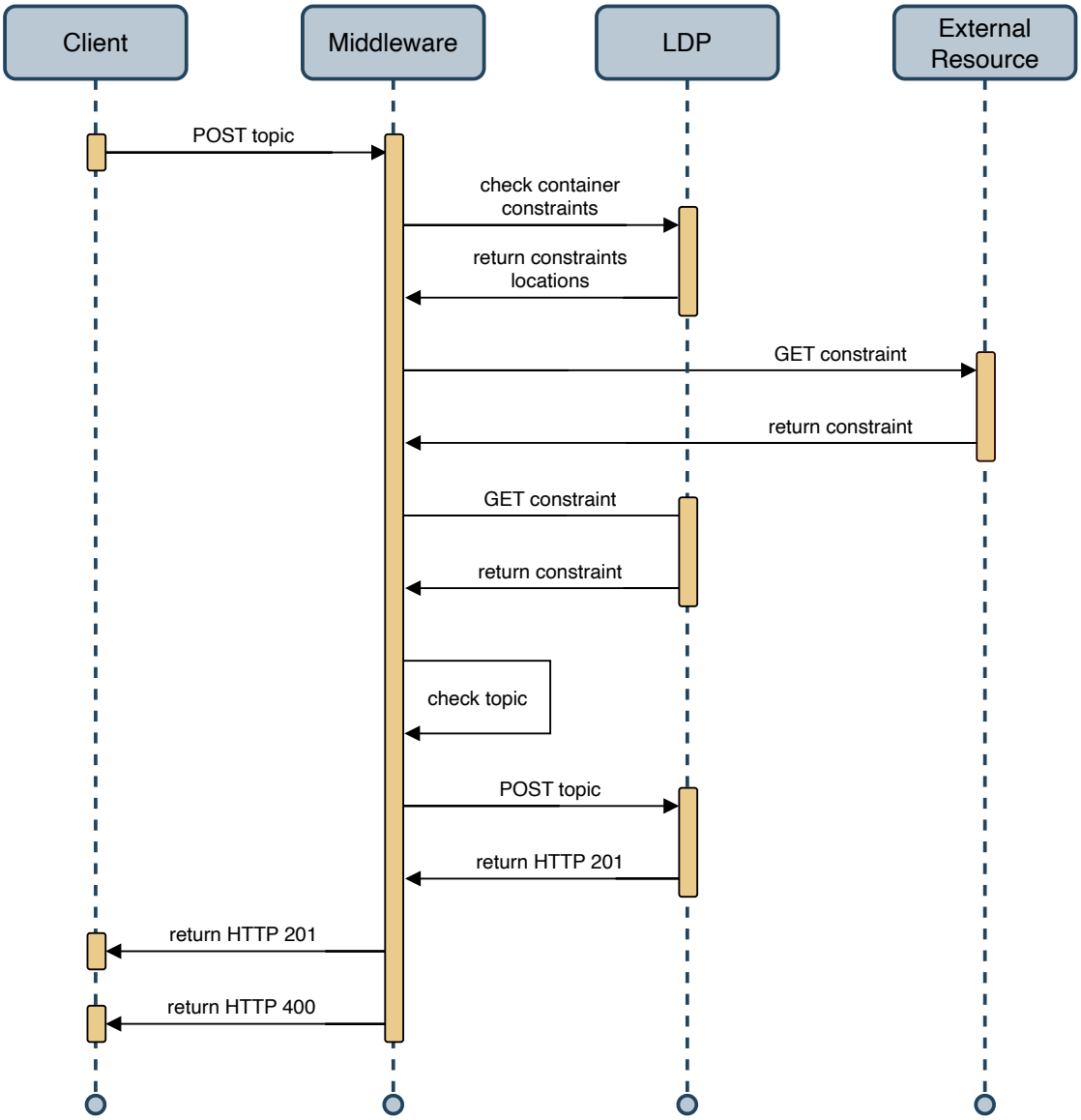

Figure 2: Sequence Diagram, describing the communication process of sending a *Topic* from the client to the LDP.

for web-based collaboration, LDP also supports that behaviour. Using the provided predicates, *ldp:hasMemberRelation* and *ldp:constrainedBy*, the specification can constrain the incoming data by using SHACL shapes as constraints without any extension of the standard. Due to the decentralised structure of Linked Data and LDP, the constraints do not necessarily reside in the exact location of the resources. Still, they can be split into project-specific and general applicable requirements, reflecting the nature of the AEC industry. In addition, it also allows the reuse of these requirements, preventing the need to set them up repeatedly. While the LDP can define that a container is constrained, it is simply a specification and not a framework

that can be used to host containers and resources. An official approach to implementing and enforcing these constraints still needs to be agreed on. Therefore, adding these constraints to an *ldp:Container* will only achieve the desired behaviour with an additional setup. Potential solutions for this problem are implementing a checking mechanism directly into an LDP or using a middleware that checks for these constraints. The latter approach was described in this paper by using a middleware that checks incoming data and decides to store or reject the data, depending on its conformance to the constraints. In subsequent steps, the conceptual implementation presented here will undergo further evaluation and will be tested in a prototype. In doing so, not only will the validation of constraints be examined, but also, how to create these requirements user-friendly without overwhelming users with Linked Data.

While this paper has explained how incoming data can be examined for conformity, future work will investigate how the data can be sent to the LDP. Serialising the data into RDF seems disadvantageous and would pose a significant disruption to current workflows. Therefore, the aim is to explore how the concepts of openCDE API communication can be combined with Linked Data approaches. Furthermore, research will be conducted on how approaches beyond specific APIs in CDEs can be used in conjunction with Linked Data. The AEC industry is based on many universal and project-specific requirements that are documented, for example, in the EIR and BEP as text. Combining these in a machine-readable format with the principle of LDP is a promising next step.

Acknowledgements

This research is funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project number 501812634

References

[1] DIN, DIN SPEC 91391-1:2019-04 Common Data Environments (CDE) for BIM projects - Function sets and open data exchange between platforms of different vendors - Part 1: Components and function sets of a CDE; with digital attachment, https://dx.doi.org/10.31030/3044838, 2019.

[2] C. Preidel, A. Borrmann, H. Mattern, M. König, S.-E. Schapke, Common Data Environment, in: A. Borrmann, M. König, C. Koch, J. Beetz (Eds.), Building Information Modeling: Technology Foundations and Industry Practice, Springer International Publishing, Cham, 2018, pp. 279–291. doi:10.1007/978-3-319-92862-3_15, https://doi.org/10.1007/978-3-319-92862-3_15.

[3] ISO, ISO 19650-1:2018 Organization and digitization of information about buildings and civil engineering works, including building information modelling (BIM) - Information management using building information modelling - Part 1: Concepts and principles, https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/06/80/68078.html, 2018.

[4] Y. Kulbak, K. Linhard, G. Dangl, P. Pasi, Activity Proposal: Open, Standardized Application Programming Interfaces (APIs) for Common Data Environments (CDEs) to Lower Threshold of Data Exchange and Sharing on BIM-based Projects AKA “Open CDE

APIs", https://github.com/buildingSMART/OpenCDE-API/blob/master/Documentation/AP-OpenCDEAPIs-20200428-v1_0.pdf, 2020.
[5] R. Verborgh, R. Taelman, LDflex: A Read/Write Linked Data Abstraction for Front-End Web Developers, in: The Semantic Web – ISWC 2020, volume 12507, Springer International Publishing, Cham, 2020, pp. 193–211. doi:10.1007/978-3-030-62466-8_13, https://link.springer.com/10.1007/978-3-030-62466-8_13.
[6] J. Werbrouck, O. Schulz, J. Oraskari, E. Mannens, P. Pauwels, J. Beetz, A generic framework for federated CDEs applied to Issue Management, Advanced Engineering Informatics 58 (2023) 102136. doi:10.1016/j.aei.2023.102136, https://www.sciencedirect.com/science/article/pii/S1474034623002641.
[7] O. Schulz, J. Oraskari, J. Beetz, Lessons Learned from Designing and Using bcfOWL, in: Proceedings of the 11th Linked Data in Architecture and Construction Workshop, Matera, Italy, 2023. https://ceur-ws.org/Vol-3633/paper2.pdf.
[8] DIN Standards Committee Building and Civil Engineering, Information container for linked document delivery Exchange specificaton Part 1: Container (ISO 21597-1:2020); German version EN ISO 21597-1:2020, https://www.beuth.de/de/-/-/318930068, 2021. doi:10.31030/3137795.
[9] DIN Standards Committee Building and Civil Engineering, Information container for linked document delivery Exchange specification Part 2: Link types (ISO 21597-2:2020); German version EN ISO 21597-2:2020, https://www.beuth.de/de/-/-/328268142, 2021. doi:10.31030/3192763.
[10] P. Hagedorn, M. Senthilvel, H. Schevers, L. B. Verhelst, Towards usable ICDD containers for ontology-driven data linking and link validation, in: Proceedings of the 11th Linked Data in Architecture and Construction Workshop, Matera, Italy, 2023. https://ceur-ws.org/Vol-3633/paper3.pdf.
[11] M. Senthilvel, J. Beetz, Conceptualizing Decentralized Information Containers for Common Data Environments using Linked Data, in: Proceedings of the Conference CIB W78 2021, Luxembourg, 2021. https://itc.scix.net/paper/w78-2021-paper-059.
[12] M. Senthilvel, J. Oraskari, J. Beetz, Common Data Environments for the Information Container for linked Document Delivery, in: Proceedings of the 8th Linked Data in Architecture and Construction Workshop, Dublin, Ireland (virtually hosted), 2020. https://ceur-ws.org/Vol-2636/10paper.pdf.
[13] S. Speicher, J. Arwe, A. Malhotra, Linked Data Platform 1.0, https://www.w3.org/TR/ldp/, 2015.
[14] N. Mihindukulasooriya, R. Menday, Linked Data Platform 1.0 Primer, https://www.w3.org/TR/ldp-primer/, 2015.
[15] C. Burleson, M. Esteban Gutiérrez, N. Mihindukulasooriya, Linked Data Platform Best Practices and Guidelines, https://www.w3.org/TR/ldp-bp/, 2014.
[16] A. V. Sambra, E. Mansour, S. Hawke, M. Zereba, N. Greco, A. Ghanem, D. Zagidulin, A. Aboulnaga, T. Berners-Lee, Solid: A Platform for Decentralized Social Applications Based on Linked Data (2016) 16. http://emansour.com/research/lusail/solid_protocols.pdf.
[17] J. Werbrouck, P. Pauwels, J. Beetz, R. Verborgh, E. Mannens, ConSolid: A federated ecosystem for heterogeneous multi-stakeholder projects, Semantic Web (2023) 1–32. doi:10.3233/SW-233396, https://www.semantic-web-journal.net/content/

consolid-federated-ecosystem-heterogeneous-multi-stakeholder-projects-0.
[18] J. Oraskari, O. Schulz, J. Werbrouck, J. Beetz, Enabling Federated Interoperable Issue Management in a Building and Construction Sector, in: Proceedings of the 29th EG-ICE International Workshop on Intelligent Computing in Engineering, EG-ICE, 2022, pp. 92–101. doi:10.7146/aul.455.c200, https://ebooks.au.dk/aul/catalog/view/455/312/1848-2.
[19] J. Labra Gayo, E. Prud'hommeaux, S. Staworko, H. Solbrig, Towards an RDF validation language based on regular expression derivatives, in: Proceedings of the Workshops of the EDBT/ICDT 2015 Joint Conference (EDBT/ICDT), volume 1330, 2015, pp. 197–204. https://ceur-ws.org/Vol-1330/paper-32.pdf.
[20] H. Knublauch, D. Kontokostas, Shapes Constraint Language (SHACL), https://www.w3.org/TR/shacl/, 2017.
[21] J. E. L. Gayo, E. Prud'hommeaux, I. Boneva, D. Kontokostas, Validating RDF Data, Synthesis Lectures on Data, Semantics, and Knowledge, Springer International Publishing, Cham, 2018. doi:10.1007/978-3-031-79478-0.
[22] E. Nuyts, J. Werbrouck, R. Verstraeten, L. Deprez, Validation of building models against legislation using SHACL, in: Proceedings of the 11th Linked Data in Architecture and Construction Workshop, Matera, Italy, 2023. https://ceur-ws.org/Vol-3633/paper13.pdf.
[23] R. K. Soman, M. Molina-Solana, J. K. Whyte, Linked-Data based Constraint-Checking (LDCC) to support look-ahead planning in construction, Automation in Construction 120 (2020) 103369. doi:10.1016/j.autcon.2020.103369.
[24] P. Hagedorn, P. Pauwels, M. König, Semantic rule checking of cross-domain building data in information containers for linked document delivery using the shapes constraint language, Automation in Construction 156 (2023) 105106. doi:10.1016/j.autcon.2023.105106, https://linkinghub.elsevier.com/retrieve/pii/S0926580523003667.
[25] M. Senthilvel, J. Beetz, A Visual Programming Approach for Validating Linked Building Data, in: EG-ICE 2020 Workshop on Intelligent Computing in Engineering, 2020, p. 9. https://depositonce.tu-berlin.de/items/a3f8b447-3925-40c9-ba9c-bfd1b9a9834e.
[26] DIN, DIN SPEC 91391-2: Common Data Environments (CDE) for BIM projects Function sets and open data exchange between platforms of different vendors Part 2: Open data exchange with Common Data Environments, https://www.beuth.de/de/technische-regel/din-spec-91391-2/302483177, 2019.
[27] O. Schulz, J. Oraskari, J. Beetz, bcfOWL: A BIM collaboration ontology, in: Proceedings of the 9th Linked Data in Architecture and Construction Workshop, Luxembourg, 2021, pp. 1–12. https://linkedbuildingdata.net/ldac2021/files/papers/CIB_W78_2021_paper_122.pdf.
[28] M. D. Wilkinson, M. Dumontier, I. J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J.-W. Boiten, L. B. da Silva Santos, P. E. Bourne, J. Bouwman, A. J. Brookes, T. Clark, M. Crosas, I. Dillo, O. Dumon, S. Edmunds, C. T. Evelo, R. Finkers, A. Gonzalez-Beltran, A. J. G. Gray, P. Groth, C. Goble, J. S. Grethe, J. Heringa, P. A. C. 't Hoen, R. Hooft, T. Kuhn, R. Kok, J. Kok, S. J. Lusher, M. E. Martone, A. Mons, A. L. Packer, B. Persson, P. Rocca-Serra, M. Roos, R. van Schaik, S.-A. Sansone, E. Schultes, T. Sengstag, T. Slater, G. Strawn, M. A. Swertz, M. Thompson, J. van der Lei, E. van Mulligen, J. Velterop, A. Waagmeester, P. Wittenburg, K. Wolstencroft, J. Zhao, B. Mons, The FAIR Guiding Principles for scientific data management and stewardship, Sci Data 3 (2016) 160018. doi:10.1038/sdata.2016.18, https://www.nature.com/articles/sdata201618.