Data Patterns for the Organisation of Federated Linked Building Data

Jeroen Werbrouck[1,3][0000−0002−4972−1189], Pieter Pauwels[2][0000−0001−8020−4609], Jakob Beetz[3][0000−0002−9975−9206], and Erik Mannens[4][0000−0001−7946−4884]

[1] Department of Architecture and Urban Planning, Ghent University, Ghent, Belgium jeroen.werbrouck@ugent.be
[2] Department of the Built Environment, TU Eindhoven, Eindhoven, The Netherlands
[3] Faculty of Architecture, RWTH Aachen, Aachen, Germany
[4] Department of Electronics and Information Systems, Ghent University – imec, Ghent, Belgium

Abstract. Few industries are as fragmented as the building sector: during the life cycle of an asset, countless stakeholders are involved, ranging from direct stakeholders such as the architect, the owner and the facility manager towards indirect data providers like governments or geospatial institutions. This 'federated' reality contrasts with the concept of a 'centralised' cloud server; a 'Common Data Environment' (CDE). In this paper, we propose a basic infrastructure for a 'federated CDE', using domain-agnostic Web specifications for (access-controlled) data federation. This infrastructure is illustrated via a proof-of-concept implementation, based on the open-source Community Solid Server.

Keywords: Linked Building Data · microservices · BIM · Semantic Web · Solid

1 Introduction

1.1 Background

Construction projects are 'decentral' by nature: there is no single entity that centralises the decision-making over the entire life cycle of a building. Instead, multiple independent stakeholders interact with it over time - going from design over maintenance or occupation to demolition and material reallocation. However, as they all relate to the same physical asset, intense collaboration between those stakeholders is often required. The rise of digital technologies such as Building Information Management (BIM) brought a lot of benefits for cross-stakeholder collaboration. Lately, such collaboration activities move more and more towards specialised cloud-based ecosystems, the so-called 'Common Data Environments' (CDE) (ISO 19650). Those CDEs are mostly provided by large BIM tool vendors, and offer an as-seamless-as-possible integration with these (often commercial) BIM authoring tools. Yet, the bigger the scope of the project, the higher chances are that those stakeholders do not all rely on the same ecosystem. After all, they represent different disciplines, so chances are the data models

they use are not aligned. To cope with these data exchange challenges, the use of Semantic Web technologies for information management during the Building Life Cycle (BLC) has been gaining traction over the last years [11]. Using the infrastructure of the 'classic' World Wide Web, the Semantic Web provides a domain-agnostic framework for defining data models, the Resource Description Framework (RDF)[5]. In RDF, atomic data entities can be related to one another in unambiguous ways, using 'Web ontologies', resulting in a 'knowledge graph': a directed graph where the nodes and relationships have unique identifiers on the Web. Data structured in RDF does not need to reside on a central server. Information hosted by different Web servers can be connected to form a *federated* network of dereferenceable information. For example, an RDF graph of the asset model could just point to open geospatial information [10], governmental data [4] or data from product manufacturers [17]; all exposed as open data on the Web. Objects and properties in the project graph itself may be 'federated' as well, as a digital mirror of the real-life consortium. Each stakeholder office then organises their project data on an office server, allowing the others to access relevant data for their respective tasks in (certain phases of) the project. When semantic links are established between the servers of these offices, the project is a network of (interoperable) data rather than a set of files on a single server. Depending on the use case, this data pool can be dynamically extended to include external information as well, e.g. contextual data provided by governmental or scientific institutions, product manufacturers and others.

1.2 Research topics and Paper Structure

In this paper, we discuss a data-based setup for a federated construction project. The focus will hereby lie on discovery of and interaction with federated project data hosted by multiple stakeholders. This infrastructure, and its implementation in the decentralised Solid ecosystem, forms the core of the ConSolid project. Although examples will be given that implement specific RDF ontologies, the underlying concepts should be generally applicable. First, we discuss some background technologies and related work on information management in a Semantic Web context (Section 2). In Section 3, structures for a federated CDE are outlined. A brief proof-of-concept is described in Section 4. The paper concludes with a discussion and an indication of future work (Section 5).

Listing 1.1. Prefixes used in this paper

```
@prefix cs: <https://consolid.org/vocabulary#>.
@prefix dct: <http://purl.org/dc/terms/>.
@prefix dcat: <http://www.w3.org/ns/dcat#> .
@prefix ldp : <http://www.w3.org/ns/ldp#>.
@prefix me: <http://localhost:5000/example/profile/card#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
```

[5] `https://www.w3.org/RDF/`

2 Technological Foundations and Related Work

2.1 Exposing Federated Data

The Linked Data Platform[6] (LDP) is a W3C recommendation for performing read-write operations on Web Resources via HTTP. Those resources may or may not be based on RDF (`ldp:RDFSource` and `ldp:NonRDFSource`), and can be contained in an `ldp:Container`, itself a `ldp:RDFSource` which describes its content and metadata.

The LDP specification is a good starting point for organising heterogeneous federated building data. However, it does not deal with authentication and authorisation of agents, which is necessary to enable the setup of a federated digital asset model, where data resides with the responsible stakeholder instead of on a centralised CDE server. Such access-control layer is provided by the Solid specifications [9]. Solid is an evolving set of specifications and Web standards focused at the 'decentralisation' of data: anyone is free to choose where their data resides and who (or which service) can access this data. This means that information is decoupled from the applications that use it [15]. In Solid, decentral identities are assigned a 'WebId'[7], which serves as a dereferenceable identifier to any agent (human or digital). A WebId is a URL that represents the actor on the Web; it can thus serve as a username as well. It is hosted by an 'identity provider', which, in most but not all cases, also hosts the personal data storage ('Pod') of the actor. An identity provider can be a specialised company, but the Solid specs allow to self-host your data and WebId as well (so you can be your own identity provider). To manage access to resources, Solid uses the WAC[8] (Web Access Control) vocabulary, where access rights are managed via '.acl' RDF graphs.

Although the original use case of Solid focuses on social networking, the specifications are domain-agnostic and can hence be used within various sectors, among which the AEC industry [18]. An open-source implementation of the Solid specifications is the upcoming Community Solid Server[9] (CSS), which is now in its beta stage of development. Built in a modular fashion, individual components can be programmed separately and 'injected' in a custom configuration [14] of the CSS. This modularity makes it well-suited for development and testing, but also to configure a CSS setup to address specific needs.

2.2 Federated Multi-Models in AEC

Because of the federated nature of the AEC industry and the diversity of the activities present in the BLC, a digital building model often consists of multiple heterogeneous resources. A multi-model [8] is an interlinked collection of BIM data which acknowledges this and focuses on the combination of BIM data with other disparate datasets. Fuchs and Scherer [5] propose a multimodel approach

[6] `https://www.w3.org/TR/ldp/`
[7] `https://www.w3.org/2005/Incubator/webid/spec/identity/`
[8] `https://solid.github.io/web-access-control-spec/`
[9] https://github.com/solid/community-server

for loose-coupling independent resources into an over-arching multi model. Sub-document connections are made using ID-based Link Objects, grouped into larger Link Models. The recent standard *Information Container for linked Data Delivery* (ICDD, ISO 21597) heavily builds upon these concepts. ICDD is oriented towards the delivery of data containers of the entire model, but since it bases upon RDF as the 'semantic glue' between resources, there is no reason this data cannot be federated on the Web. For example, mappings are possible between ICDD and LDP containers [13].

2.3 Resource Metadata

The Solid specifications do not (yet) include a way to express resource metadat a[10]. However, expressing certain statements about an RDF or non-RDF resource, rather than about the content of this resource, is an important aspect of finding your way in a federated project. The distinction can be made here between generic metadata (e.g., last-modified, content-type) and specific metadata (e.g., topic, description, tags). While the former can be expressed easily using HTTP Headers, the latter needs a more static approach such as attaching a `.meta` auxiliary to each resource - especially concerning domain-specific metadata. One domain-agnostic approach to describe datasets on the web is the DCAT 2.0 specification[11], which relates a `dcat:Catalog` to multiple separate `dcat:Datasets`, which are *collections of data, published or curated by a single agent, and available for access or download in one or more representations.* A `dcat:Dataset`, in turn, links to instances of `dcat:Distribution`. A distribution description may contain information on the resource's content-type (`dct:type`), description (`dct:description`) and title (`dct:title`) but will also point to URLs for downloading the dataset (`dcat:downloadURL`).

2.4 Common Data Environments

Section 1.1 mentioned the concept of a CDE. A CDE exposes contributions by specific stakeholders to other members of the consortium, and is often integrated in an existing (commercial) BIM ecosystem. Information resides 'in the cloud', to be accessed via a provider-specific ID. CDEs often provide additional functionality to enable stakeholders to communicate about sub-model collisions, inconsistencies, or general change of plans. CDEs need to handle the double patchwork present in the AEC industry: each project has different stakeholders, and each stakeholder has multiple projects. A CDE is mostly project-specific, but an office cannot maintain subscriptions to different CDEs for every project. A notable initiative here is the ongoing OpenCDE project (BuildingSMART), which provides a set of APIs that allow a client to communicate with multiple CDEs. All OpenCDE APIs share a limited amount of services and conventions, which are

[10] `https://github.com/solid/specification/issues/102`
[11] `https://www.w3.org/TR/vocab-dcat-2/`

bundled in the Foundation API[12]. Currently, the most active OpenCDE API is the BCF-API[13], which supports exchange of BCF (BIM Collaboration Format) 'topics' between software applications via a RESTful API; a 'BCF server' as a global access point for the project. BCF API furthermore supports, amongst others, the retrieval and modification of existing projects, BIM snippets, comments and entire project files.

2.5 Related Work

The **SCOPE project** (Semantic Construction Project Engineering)[14] [7, 16] is a research project that fully embraces the use of Semantic Web technologies for the construction sector. One of the project's aims is to provide a software-independent, lossless environment for building project data, with a focus on parametric building product data and the building envelope. Project information ('abstract' semantics as well as geometry etc.) is stored in triple stores in the cloud; product information from manufacturers is handled in a similar ´product cloud'. Microservices may access these datasets via SPARQL queries.

Another ongoing project, which integrates the concept of Linked Building Data with the well-known **BIMserver** [1], is the **BIM4Ren** (BIM for Renovation) project [3]. BIM4Ren envisages a one-stop access platform where different 'BIM bots' can be triggered, possibly chained to address larger business processes for renovation. Project data and BIM objects as well as BIM services are managed in a decentral way. The interaction between BIM bots on the Web and existing, standalone BIM authoring tools is a topic of interest in this joint venture between academia and industry.

The **DRUMbeat Platform** [6] is a research project towards an infrastructure for publishing and linking BIM-related data on the 'Web of Building Data'. Project data, integrated at object level, is federated over multiple DRUMbeat servers, where it can be accessed by clients through a common REST API layer. The project has provided multiple proof-of-concept implementations to demonstrate information flows in federated building projects.

3 Infrastructure for Federated Building Projects

A federated CDE is essentially a federated, access-controlled knowledge graph. Autonomous agents (both human and digital) have a clearly defined set of tasks and responsibilities in each project they are involved in. We may identify three 'types' of actors in the federated network: the 'main' stakeholders, who are directly involved in the project, the 'external' datasets, providing auxiliary information, and microservices. In the following paragraphs, we will mainly use Solid

[12] `https://github.com/buildingSMART/foundation-API`
[13] `https://github.com/BuildingSMART/BCF-API`
[14] `https://www.projekt-scope.de/en/home-en/`

terminology (Pods, WebIds, ...) to outline the structure of the network. However, as the Solid specifications base upon general Web-standards, they may be applied also in non-Solid environments.

Figure 1 shows an overview of the different components in the conSolid infrastructure. Before the different aspects of this figure will be highlighted in the sections below, some notes on services are required, as they will be regularly mentioned in the rest of this Section.

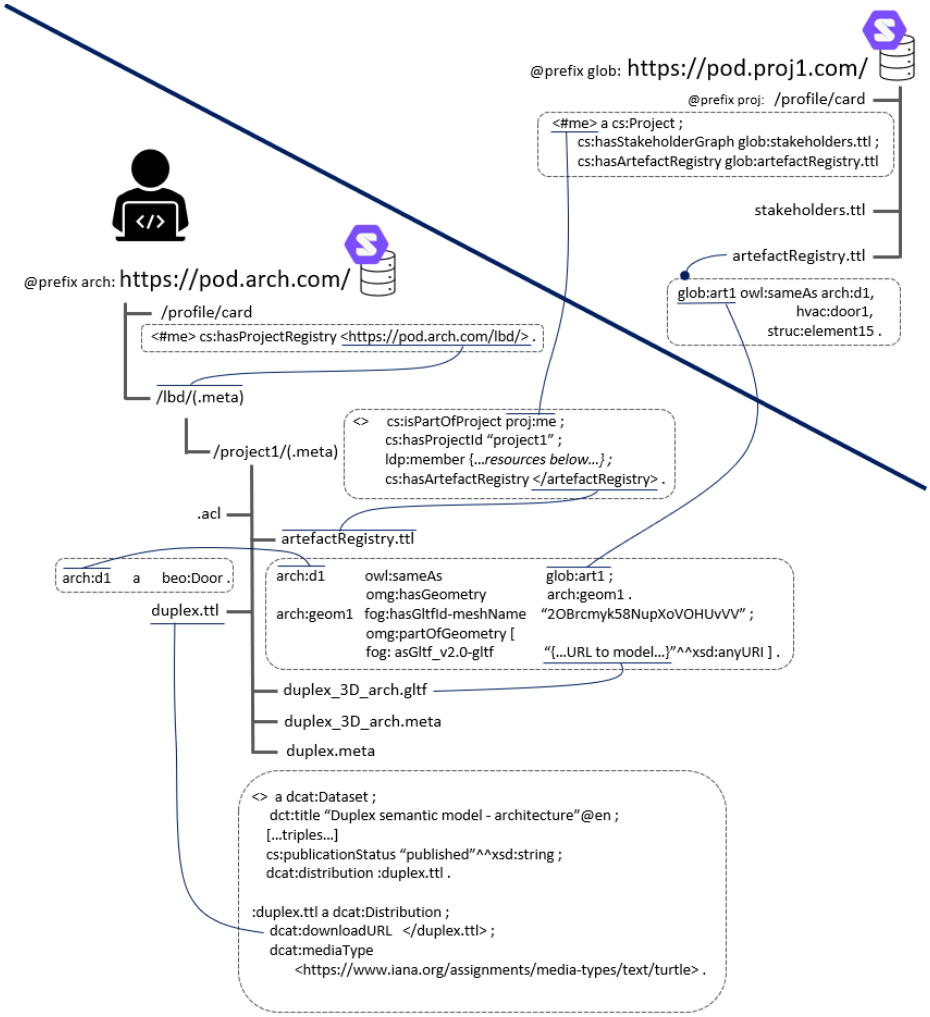

Fig. 1. Overview of the semantic links in a federated ConSolid Project.

3.1 Services

We can roughly distinguish three types of services in the network. The first type of service is project- or even Pod-exclusive. Their task is to keep the project synchronised, to expose project data in specialised databases or to generate specific viewpoints on the project data (e.g., an OpenCDE-conform API or a virtual view generator (Section 3.2)). As they will often involve managing access rights as well, these services require a very high level of trust and should be administered by the owner of the Pod or a dedicated project manager. The other services are provided by third parties, with a distinction between Open Data services and functional services. While Open Data services (e.g., governmental APIs, product manufacturers, geospatial datasets) will not require any access rights to the project, functional tools (e.g., for simulations, regulation checking etc.) need to have at least 'reading' rights. Functional tools can be compared to the 'BIM bots'-concept featured in the BIMserver project (Section 2.5).

3.2 Local Project Repository

General structure A Solid Pod bases upon the LDP specification, so it has a folder-like structure that can be accessed via HTTP URLs. Starting from the WebId of an authenticated stakeholder, a client service needs to find where LBD projects are stored. In their WebId, a stakeholder must provide a pointer (`cs:hasProjectRegistry`) to a project registry folder (`ldp:Container`). At the root of this `ldp:Container`, the stakeholder's *local* project repositories are referenced (`ldp:member`) - these will be `ldp:Container`s as well (Fig. 1). Relevant metadata about the project is attached to this `ldp:Container` via a `.meta` file, which is the default response when fetching the container resource itself. For the structure of an individual project repository, we propose not to use sub-directories but instead use a flattened list of project resources. The main reason to do this, is the persistence of URIs in the project: moving resources from one sub-folder to another one will easily break links in the network. However, this flattened list only exists at data storage level - using metadata resources (Section 3.2), 'virtual views' on this data can be created, to improve user experience. Such virtual views will be based on specific parameters (e.g. document status according to ISO 19650, virtual folder structures following ICDD (Section 2.2), document ownership, discipline, project task etc.) and can be either offered by specialised services in the network, or be rendered client-side in a web app.

Resource Auxiliaries We identify two types of auxiliaries to resources in a federated multi model. The first regulates Web Access Control: rules for authorisation are expressed in an .acl document that governs either an entire `ldp:Container` or applies only to a specific resource. The second is a metadata (`.meta`) file (Section 2.3), giving extra context about a resource. As in our infrastructure, the metadata file forms the primary access point to the actual resource, it is required that every resource in the project has their own metadata file attached. In our ecosystem, `.meta` files will follow the DCAT 2.0 specification

as described in Section 2.3. An example is given in Figure 1, with the metadata file `duplex.meta`. The example describes a dataset with a known distribution hosted in the Pod. Section 3.2 will further elaborate on making alternative distributions available via specialised databases.

Linking external databases In the previous section, we argued that the access point for every dataset is its related `.meta` resource. From there on, a client can find the distributions it needs. Exposing all distributions in a flattened `ldp:Container`, as proposed in Section 3.2 and visualised in Figure 1, complies with the LDP and Solid specifications, and is implementationwise the most straightforward one. However, sometimes specialised databases offer better performance or additional functionality. For example, an (access-restricted) triple store for RDF project resources will offer better server-side query performance or provide reasoning or rule-checking functionality, while availability risks are relatively low compared with public SPARQL endpoints. Another use case dwells upon the concept of content negotiation: while one 'main' distribution is stored on the Pod, another one may just point to a service that converts other representations on-the-fly or actively watches the main folder and caches alternative representations. In the case an (RDF-based) distribution is to be queried rather than downloaded, the property `dcat:accessURL` is used. When connecting external databases to a federated project network, a challenge is to pass on access-control rules for these services, which requires a very high level of trust. Therefore, we consider such services to be self-hosted or at least Pod-specific, i.e., exclusively working on one data vault.

3.3 Global Access Point for a Federated Project

In Section 2.4, a BCF project server was mentioned as a single point through which BCF issues may be exchanged, pointing to project files on a model server. Similarly, commercial CDEs will provide a project identifier to find the project on their servers, and file-sharing platforms use a folder structure to bundle resources related to a specific topic. In a federated environment, it makes sense to determine one or more 'global access points' as well. To enable discovery of a global access point, a stakeholder needs to indicate a part-whole relationship (`cs:isPartOfProject`) at the root of the *local* project repository (Section 3.2). From there on, each agent with the correct access rights can propagate through the federated multi model. A project access point can be set up by a principal stakeholder (e.g. the architect or the project manager) and has its own assigned WebId. Depending on the organisation of the project, the access point may expose its own project metadata resources (e.g., a graph of stakeholders or a list of abstract artefacts (Section 3.4) and host multiple services.

Now that local and global project access points can be discovered and the structure of metadata has become clear, the following section will discuss how in this federated network, resources can be linked on a sub-document level.

3.4 Referencing Abstract Elements

In a digital construction project, multiple resources may actually refer to one and the same 'real' object. A real window in the building may find itself semantically described in graphs produced by different stakeholders in the project, and represented in multiple images, point clouds and geometries. About the latter we can say that these are graphical representations which can act as proxies to talk about the real 'abstract' building element: if you select an element in a 3D viewer or mark a region in an image, you only implicitly refer to this abstract element. The same goes for resources like spreadsheets or text files, which are still very much used in industry. To semantically define such references to abstract objects, we follow the approach used in the LBD 'tandem ontologies' *Ontology for Managing Geometry*[15] (OMG) [16] and *File Ontology for Geometry formats* [16] (FOG) [2], which consider geometry a specific property and hence base upon the *Ontology for Property Management*[17] (OPM) ontology [12] to link geometric instances to abstract objects via 'property levels'. A similar approach can be followed, however, for non-geometric auxiliaries. Comparable with the multi-model approach (Section 2.2), sub-document elements can be refered to via ID-based links. When there are no identifiers in the document (e.g. because the resource is a binary image or a point cloud), semantic descriptions can be attached to the metadata, for example to identify 2D or 3D regions.

In a *central* environment, a 'master artefact registry' could be easily set up to keep track of such references. All references to abstract artefacts are then made in this registry, which acts as a 'single source of truth'. In a federated CDE, such master registry is still necessary (to avoid arbitrary local creation of new abstract elements by any stakeholder who claims the referenced element did not exist yet in the network). Several approaches may be used to set up this master registry. One of them is be to use blockchain technologies - as a fully *distributed* solution. Another one is to host the registry in the project access point, or in the Pod of a primary stakeholder (e.g. the project manager).

Although this registry will be the main reference to connect project resources and semantic instances, there are advantages to allow individual stakeholders to maintain a *local* registry as well: drafting work-in-progress, querying, offline working, etc. Moreover, it would be undesirable that each stakeholder has full writing rights to the registry at the project access point: as a design pattern in the network, a stakeholder should only make changes to resources they host in their own Pod, or be allowed to use a proxy service that may only perform specific data manipulations on remote resources.

We thus propose an in-between solution where each stakeholder may use a local reference to the abstract element in an *artefactRegistry* graph in their local project repository, indicating that there exists another URI in the master registry, which refers to the same thing (`owl:sameAs`). Network- specific services (Section 3.1 should ensure local artefact registries are regularly synchronised

[15] `https://www.projekt-scope.de/ontologies/omg/`
[16] `https://mathib.github.io/fog-ontology/`
[17] `https://w3c-lbd-cg.github.io/opm/`

with the global 'base registry' and vice versa. I.e, the `owl:sameAs` relationship in local registries should have a backlink in the master registry, so all references to a certain abstract artefact can be easily discovered: a service locates the *global* artefact registry, follows the pointers to the local artefact registries, collects the references from there and bundles these for the client to use.

4 Implementation

As a proof of concept, we recreated the components discussed in Section 3 as separate modules for the AECOstore project[18] [19]. The AECOstore is an ecosystem (in-development) to create micro-front-ends on top of heterogeneous building data. Using a Solid-enabled authentication module, agents can log in with their WebId. A setup-module allows to either create a project access point, or join a project based on open 'invitations'. An authenticated agent can now interact with her own project data as well as with the data of other stakeholders, upload resources, follow links to external data providers or enrich the overall semantic model by making use of dedicated interaction modules. Data storage happens as described in Section 3, using Community Solid Server instances as the main data- and identity providers. Auxiliary services for synchronisation and backlink creation were implemented as NodeJS servers. Figure 2 shows a prototype where several independent modules are configured to interact with the same federated multi-model. Project resources, provided by three stakeholders, can be 'activated', linked and displayed in specialised modules.

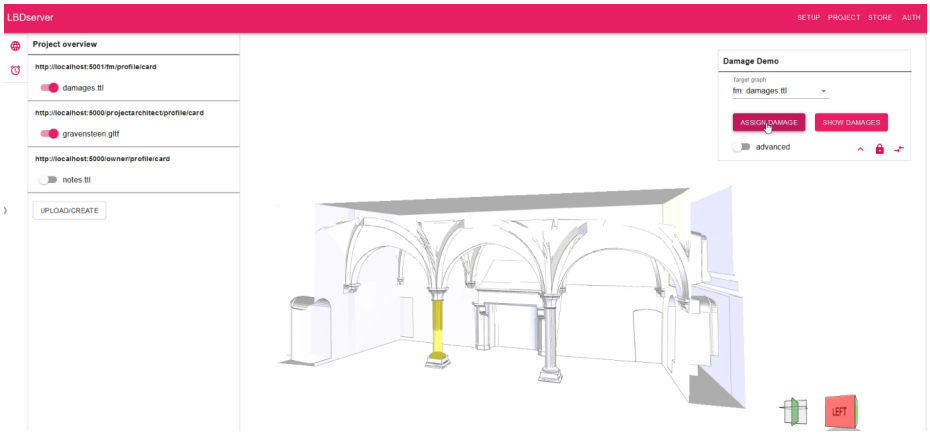

Fig. 2. Modular AECOstore GUI, featuring a Viewer, Resource Overview and Semantic Enrichment module, to interact with the federated network described in this paper.

[18] `https://github.com/ConSolidProject/`

5 Future Work and Conclusion

In this paper, we discussed a method for structuring building project data in a federated manner. Without going into detail about the nature of such data, we have shown that project contributions from different stakeholders can be hosted on different servers, yet remain semantically connected and unambiguously part of the same overall project. With multiple Community Solid Server instances featuring as back-ends and a generic dashboard interface based on the LBDserver project, we implemented a proof-of-concept showing that browser-based and headless microservices can interact with this data. Depending on the use case, such microservices may offer generic functionality or provide the means for tailor-made and discipline-specific data enrichment in different phases of the Building Life Cycle. The limited scope of this paper prohibited topics such as archiving, versioning, multi-user editing and synchronisation streams to be covered. The conceptualization of compatible data patterns for these topics is an important remaining work package.

As we proceed with this research, the integration of the data structures described in this paper ('ConSolid') with the 'AECOstore' infrastructure for micro-services and –frontends (Section 4) will become more important. The development of common API functions to interact with the data on a layered, more user-friendly level is essential to leverage the potential offered by the combination of Semantic Web technologies with modern-day web development technologies, a combination which may eventually result in tailor-made and discipline-specific data enrichment in different phases of the building life cycle.

6 Acknowledgements

This research is funded by the Research Foundation Flanders (FWO) in the form of the Strategic Basic Research grant (grant no. 1S99020N).

References

1. Beetz, J., van Berlo, L., de Laat, R., van den Helm, P.: Bimserver. org - an open source ifc model server. In: Proceedings of the 27th CIB W78 conference. p. 8 (2010)
2. Bonduel, M., Wagner, A., Pauwels, P., Vergauwen, M., Klein, R.: Including widespread geometry formats in semantic graphs using rdf literals. In: 2019 European Conference on Computing in Construction. pp. 341–350. European Council on Computing in Construction (2019)
3. Bourreau, P., Charbel, N., Werbrouck, J., Senthilvel, M., Pauwels, P., Beetz, J.: Multiple inheritance for a modular bim. Le BIM et l'évolution des pratiques: Ingénierie et architecture, enseignement et recherche p. 63 (2020)
4. Buyle, R., De Vocht, L., Van Compernolle, M., De Paepe, D., Verborgh, R., Vanlishout, Z., De Vidts, B., Mechant, P., Mannens, E.: Oslo: Open standards for linked organizations. In: Proceedings of the international conference on electronic governance and open society: Challenges in Eurasia. pp. 126–134 (2016)

5. Fuchs, S., Scherer, R.J.: Multimodels—instant nd-modeling using original data. Automation in Construction **75**, 22–32 (2017)
6. Hoang, N.V., Törmä, S.: Drumbeat platform—a web of building data implementation with backlinking. In: eWork and eBusiness in Architecture, Engineering and Construction, pp. 155–163. CRC Press (2017)
7. Huyeng, T.J., Thiele, C.D., Wagner, A., Shi, M., Hoffmann, A., Sprenger, W., Rüppel, U.: An approach to process geometric and semantic information as open graph-based description using a microservice architecture on the example of structural data. In: Ungureanu, L.C., Hartmann, T. (eds.) EG-ICE 2020 Workshop on Intelligent Computing in Engineering. Universitätsverlag der TU Berlin, Berlin (June 2020), http://tubiblio.ulb.tu-darmstadt.de/121623/
8. Katranuschkov, P., Weise, M., Windisch, R., Fuchs, S., Scherer, R.J.: Bim-based generation of multi-model views (2010)
9. Mansour, E., Sambra, A.V., Hawke, S., Zereba, M., Capadisli, S., Ghanem, A., Aboulnaga, A., Berners-Lee, T.: A demonstration of the solid platform for social web applications. In: Proceedings of the 25th International Conference Companion on World Wide Web. pp. 223–226 (2016). https://doi.org/10.1145/2872518.2890529
10. McGlinn, K., Debruyne, C., McNerney, L., O'Sullivan, D.: Integrating building information models with authoritative irish geospatial information. In: International Semantic Web Conference (Posters, Demos & Industry Tracks) (2017)
11. Pauwels, P., Zhang, S., Lee, Y.C.: Semantic Web Technologies in AEC industry: A Literature Overview. Automation in Construction **73**, 145—-165 (2017). https://doi.org/10.1016/j.autcon.2016.10.003
12. Rasmussen, M.H., Lefrançois, M., Pauwels, P., Hviid, C.A., Karlshøj, J.: Managing interrelated project information in AEC Knowledge Graphs. Automation in Construction **108** (2019). https://doi.org/10.1016/j.autcon.2019.102956
13. Senthilvel, M., Oraskari, J., Beetz, J.: Common data environments for the information container for linked document delivery pp. 132–145 (2020), http://ceur-ws.org/Vol-2636/10paper.pdf
14. Taelman, R., Vander Sande, M., Verborgh, R.: Components.js: A semantic dependency injection framework. In: Proceedings of the The Web Conference: Developers Track (Apr 2018), http://componentsjs.readthedocs.io/
15. Verborgh, R.: Designing a linked data developer experience (2018), https://ruben.verborgh.org/blog/2018/12/28/designing-a-linked-data-developer-experience/
16. Wagner, A., Bonduel, M., Pauwels, P., Rüppel, U.: Relating Geometry Descriptions to its Derivatives on the Web. In: Proceedings of the European Conference on Computing in Construction (EC3 2019). pp. 304–313. Chania, Greece (2019). https://doi.org/10.35490/EC3.2019.146
17. Wagner, A., Rüppel, U.: Bpo: The building product ontology for assembled products. In: 7th Linked Data in Architecture and Construction Workshop (LDAC), CEUR Workshop Proceedings. pp. 106–119 (2019), http://ceur-ws.org/Vol-2389/08paper.pdf
18. Werbrouck, J., Pauwels, P., Beetz, J., van Berlo, L.: Towards a decentralised common data environment using linked building data and the solid ecosystem. In: Proceedings of the 36th CIB W78 2019 Conference. pp. 113–123 (2019), https://biblio.ugent.be/publication/8633673
19. Werbrouck, J., Pauwels, P., Beetz, J., Mannens, E.: Aecostore: an infrastructure for managing heterogeneous linked building data (submitting)