Mapping Federated AEC projects to Industry Standards using dynamic Views

Jeroen Werbrouck[1,3], Pieter Pauwels[1,2], Jakob Beetz[3] and Erik Mannens[4]

[1]*Department of Architecture and Urban Planning, Ghent University, Ghent, Belgium*
[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

Web-based construction projects are rapidly becoming commonplace. Domain-specific collaboration platforms, the so-called Common Data Environments (CDEs), facilitate complex interactions between the various stakeholders participating in a project. CDEs are developed and maintained by the large BIM companies allowing deep integration with BIM authoring tools. Notwithstanding the benefits such integration offers, usage of proprietary tools, data models and platforms holds the risk of a vendor lock-in and creates dependencies on platform APIs as the sole funnels trough which project data can be accessed - even when using open data formats. Recently, technologies for re-decentralising the Web are under increasing interest, as they allow decoupling data storage from applications. The Solid initiative bundles these technologies in a domain-independent way. In previous work we have already discussed data patterns for the AEC industry, using these technologies - the LBDserver. In this paper, we demonstrate how these very generic data organisation patterns can be aligned with organisational structures of some common industry standards: ISO 19650, ISO 21597 and the BCF API specifications. To achieve full alignment with all three standards will be out of scope for this paper. Rather the aim is to demonstrate a Linked Data-based, federated environment such as the LBDserver is compatible with existing (centralised) approaches while maintaining the benefits of organising digital projects in a federated way.

Keywords

BIM Level 3, Solid, DCAT, Linked Data Platform, Common Data Environment

1. Introduction

1.1. Context

In recent years, the construction industry has been digitising rapidly. One of the driving forces behind this digitisation are advancements in Web-based collaboration, another one is data interoperability - achieved through usage of open standards. Nowadays, Web-based collaboration is often managed via centralised platforms, called Common Data Environments (CDEs). The most widely used CDEs are commercial platforms, tightly integrated with traditional (proprietary) BIM authoring tools. This integration offers an as seamless-as-possible workflow,

LDAC 2022: 10th Linked Data in Architecture and Construction Workshop, May 29, 2022, Hersonissos, Greece
jeroen.werbrouck@ugent.be (J. Werbrouck)
0000-0002-4972-1189 (J. Werbrouck); 0000-0001-8020-4609 (P. Pauwels); 0000-0002-9975-9206 (J. Beetz); 0000-0001-7946-4884 (E. Mannens)

but also holds the risk of a vendor lock-in, both regarding the use of proprietary formats and the storage of project data on centralised platforms. Only reachable via specific API endpoints, the possible interactions of project stakeholders with their own models lies in the hands of third-party software vendors.

At the same time, domain-agnostic Web technologies aim towards a re-decentralisation of data on the Web. Essentially this means that there is no one-on-one relationship between an application and the data it uses: data can be federated on the Web, aggregated by a client that knows how to discover, retrieve and combine this information in a meaningful way. Important here is the Solid project [1] and the Solid specifications[1], which rely on the Semantic Web and its core standard the Resource Description Framework (RDF)[2] to accomplish this. In Solid, every agent (human or digital) has its own 'decentral' identity on the Web ('WebID'), and a data storage point ('Pod') where access-regulated information can be disclosed to specific people (using their WebIDs) or made public[3]. A WebID is a URL that resolves to an RDF document (the 'card') which contains public information about the agent and which can be used as a 'decentral username' on the Web, thereby facilitating decentral authentication and authorisation mechanisms. A global perspective on applying the Solid specifications on the needs of the (very 'federated') AEC industry is given in [2], while data patterns for a 'federated CDE' based on Solid are discussed in [3], in context of the LBDserver project.

Aligning digital project organisation with the federated nature of the industry is only one of the advantages commonly associated with the application of Semantic Web technologies for the AEC industry. The large majority of research in this area is concerned with data interoperability and linking cross-domain information on a data level [4]. Specific domain models for the AEC industry are the official RDF-based representation of the Industry Foundation Classes (IFC), ifcOWL [5], and the modular stack of ontologies covered by the W3C Linked Building Data Community Group, among which the Building Topology Ontology (BOT)[4], the Damage Topology (DOT)[5] and the Ontology for Property Management (OPM)[6].

However, using the Semantic Web for federated data storage is decoupled from using the same technologies for structuring AEC datasets. This means that the benefits of federated storage are available without disrupting a notoriously traditional and largely file-based industry: it is perfectly possible to wire a federated network of heterogeneous project data (including proprietary file formats) with a client still handling it *as if* the data is centralised [6]. At the same time, it is clear that ontology-based projects ease the integration and coupling with other federated datasets: using scoped, modular domain ontologies like the ones mentioned above can be rather easily connected to adjacent domain knowledge such as GIS, governmental datasets, Facility Management information or historical knowledge - all most likely to live on the Web on different servers (and hence part of a federated knowledge graph), in contrast with a centralised approach.

[1] Solid specifications, https://github.com/solid/solid-spec, visited 15/02/2022
[2] Resource Description Framework, http://www.w3.org/1999/02/22-rdf-syntax-ns#, visited 15/02/2022
[3] https://github.com/solid/webid-oidc-spec
[4] Building Topology Ontology, https://w3id.org/bot#, visited 16/02/2022
[5] Damage Topology Ontology, https://w3id.org/dot#, visited 16/02/2022
[6] Ontology For Property Management, https://w3id.org/opm#, visited 16/02/2022

1.2. Research Questions and Paper Overview

In this paper, we investigate compatibility of the federated data structures of the LBDserver ecosystem (proposed in earlier papers) with existing industry practices. We will demonstrate by example how a Linked Data-based, federated environment such as the LBDserver is compatible with existing (centralised) approaches, while maintaining the benefits of organising digital projects in a federated way. Hence, the research questions discussed in this publication can be formulated as follows:

- RQ1: Does a metadata-based organisation of federated AEC project data allow flexible mapping and filtering of this data in query-based 'virtual views'?
- RQ2: Can such virtual views be used to present federated project data in standard-compliant ways?

In our handling of these research questions, we base upon established standards rather than proprietary formats. National and international standards such as ISO 19650, ISO 21597, PAS 1192:2013 (UK) and the BCF API specifications[7] provide specifications for digital collaboration in the Architecture, Engineering and Construction (AEC) industry, in an effort to enable information exchange between disparate CDEs. This is achieved by devising common API structures, containerisation of project files and shared data formats. After a brief overview of related technologies (Section 2), we present a methodology for metadata-based generation of virtual views on federated projects, which present project data to the client complying with these standards. We evaluate and illustrate this methodology with virtual views on:

- The stages of data publication as defined in ISO 19650;
- Generating a project container based on ISO 21597 (ICDD);
- Exposing project issue data ('Topics') in a way compatible with the BCF API specifications)
  .

The former two can be achieved entirely client-side (Section 3), the latter focuses on how a middleware service can be used to present the data in a standard-compliant way (Section 4). The paper concludes with a discussion and overview of future work.

2. Related Work

In this section, we introduce the main technologies on which this work will rely. Describing the basics of the Semantic Web is out of scope for this paper; for the reader in need we refer to [7], which gives an elaborate introduction to the topic. Regarding decentral access control in a Solid-based environment, the comprehensive background is provided by the Solid and OpenID Connect (OIDC)[8] specifications, while pattern- and role-based extensions for the AEC industry are described in [8].

[7] https://github.com/BuildingSMART/BCF-API
[8] OpenID Connect, https://openid.net/connect/, visited 16/02/2022

In the following sections, we briefly cover resource containment with the Linked Data Platform (LDP) (Section 2.1), metadata resources using the DCAT vocabulary (Section 2.2), and the data patterns used in the LBDserver (Section 2.3), upon which this paper will base its contribution.

2.1. Linked Data Platform

As the main specification for serving resources, Solid bases upon the Linked Data Platform Specification (LDP)[9]. In LDP, resources are aggregated (`ldp:contains`) in `ldp:Container`-s. This containment structure can be used to mimic a folder-based structure in a RESTful API, allowing a read/write Web of data. LDP has proven very useful as a straightforward mechanism for organising data containers in Solid Pods. However, its expressive power is limited, which means that other frameworks need to be used to express semantically rich metadata.

2.2. Metadata and Data Catalogs on the Web

When we model a digital construction project as a federated multi-model, a data model is needed to allow discovery and aggregation of its resources. The Data Catalog vocabulary (DCAT)[10] offers domain-independent definitions for this purpose. In DCAT, a `dcat:Catalog` references 'datasets' (`dcat:Dataset`); metadata descriptions of resources. Datasets may, in turn, have multiple 'distributions' (`dcat:Distribution`), which contain the actual information. As DCAT is RDF-based, catalogs can be federated - which makes them an ideal fit to describe building datasets [9]. Listing 1 provides an example of such dataset description.

Listing 1: A dcat:Catalog can reference multiple dcat:Dataset-s; each dataset can have multiple dcat:Distribution-s with specific properties

```
# contents of the /local/datasets/ container
<partialCatalog1> a dcat:Catalog;
    lbds:DatasetRegistry; #see section 2.2
    dcat:dataset <dataset1/>, <dataset2/>, <dataset3/> .

#contents of the /local/datasets/dataset1/ container
<dataset1/> a dcat:Dataset ;
    dct:title "the title of the dataset" ;
    dcat:keyword "damage" ;
    dcat:distribution <dataset1#distribution1>,
                      <dataset1#distribution2> .

<dataset1#distribution1> a dcat:Distribution;
    dcat:mediaType <https://www.iana.org/assignments/media-types/text/turtle> ;
    dcat:downloadUrl <dataset1/distribution1> .
```

2.3. Project Discovery and Data Patterns of the LBDserver

In [3], data patterns are described to manage federated construction projects using Solid, LDP and DCAT. A cascading structure of 'aggregators' allows the discovery of 'project access points', i.e. containers referencing (`lbd:aggregates` the contributions of other stakeholders, each one stored in (remote) container or `lbds:PartialProject`. Knowing the WebID of a project partner is then enough to discover their projects and contributions in an access-controlled

[9] Linked Data Platform, http://www.w3.org/ns/ldp#, visited 17/02/2022

[10] Data Catalog vocabulary, https://www.w3.org/TR/vocab-dcat-2/, visited 17/02/2022

way, as illustrated by the SPARQL query in Listing 2. Using a query engine that supports link traversal [10], dynamic data discovery is possible by sending an initial query to the stakeholder's WebID. However, the query can also be split up into separate sub-queries to discover relevant datasets one by one, in a more classic approach.

Listing 2: SPARQL query to discover LBDserver project access points and their constituent partial projects.

```
SELECT DISTINCT ?partial WHERE {
    #initial source: http://localhost:5000/test/profile/card#me
    <http://localhost:5000/test/profile/card#me> lbds:hasProjectRegistry ?reg .

    #link traversed source: ?reg
    ?reg ldp:contains ?proj .

    #link traversed source: ?proj
    ?proj lbds:aggregates ?partial .
}
```

An `lbds:PartialProject` contains several registries, of which the 'dataset registry' is the most important in this context: it contains a flat list of metadata containers, describing project resources as `dcat:Dataset`-s. A query for discovering datasets is given in Listing 3. Filtering project resources in Section 3 will be based on this flat list.

Listing 3: SPARQL query to discover project datasets on a partial project.

```
SELECT DISTINCT ?ds WHERE {
    # a partial project
    <http://localhost:5000/test/lbd/51fdc267-0dfd-46b8-902e-335fb2e73ad0/local/>
            lbds:hasDatasetRegistry ?dsreg .

    #link traversed source: dataset registry
    ?dsreg ldp:contains ?ds .
}
```

A partial project may indicate an alternative endpoint (SPARQL, MongoDB, SQL ...) where this data is served, as an alternative to the file-based approach common in current Solid implementations (a 'satellite').

2.4. Standards considered in this paper

This paper tests the flexibility of LBDserver dataset storage by applying virtual views and re-alignments on three existing industry standards. These standards are:

- ISO 19650: the main ISO standard for organising information about an asset during its life cycle. It focuses on a file-based information exchange in a shared Common Data Environment. In this paper, we will only focus on a very small part of the standard for demonstrative purposes, which defines the stages of publication for a resource: 'work-in-progress', 'shared', 'published' and 'archived'.
- ISO 21597: The Information Container for linked Document Delivery (ICDD) defines a container format to exchange and archive heterogeneous 'multi-models' of a built asset, based on RDF ontologies. ICDD containers, which can be considered a dump repository of the project, consist of a root 'index.rdf' document, a subfolder with RDF-based resources ('Payload Triples'), one with non-RDF-based resources ('Payload documents') and finally a subfolder containing the used ontologies ('Ontology resources'). A comparative study on the structures of ICDD and LDP is covered in [11].

- BCF API: The BIM Collaboration Format (BCF) API[11] is a member of the OpenCDE Foundation API specification family, a buildingSMART-covered initiative for facilitating information exchange between CDEs on the market. It defines a series of HTTP endpoints to retrieve project issue data ('Topics') in a tree-like fashion, using standardised response bodies.

In Section 3, the metadata stored in LBDserver datasets will serve as the input to query the federated project and generate views according to parameters put forward by the above-mentioned standards.

3. Client-side views on Federated Project Data

When an LBDserver project has been discovered, a union of all partial projects forms the knowledge graph of the project. Federated queries can be executed on this list of project resources. Depending on the setup of the Solid Servers in the project, such queries can be carried out completely client-side (using a client-side query engine and source discovery as described in Section 2.3) or in a hybrid way - when partial projects are served via an access controlled SPARQL endpoint (`lbds:hasSparqlSatellite`). In this section, these queries are concerned with filtering and organising resources in a specific way, and hence have metadata resources, i.e., DCAT datasets, as their main sources. We propose the use of Linked Data Platform (LDP) Containers (`ldp:Container`) to exchange sets of project resources adhering to the query - in other words, filters on the available datasets will be presented as if they were default LDP Containers. Applications relying on Solid data structures will thus experience no difference between query-based (virtual) containers and effectively hosted Solid containers. A SPARQL CONSTRUCT query can immediately generate such virtual container. In the following section, we illustrate the flexibility of this approach by remapping the LBDserver datasets structure to common (standardised) data patterns. Note that this does not change the resources themselves: they are only grouped (filtered) in an alternative way, as a list of resource URLs aimed at discovery.

3.1. UC 1: stages of publication according to ISO 19650

The example in Listing 4 yields an `ldp:Container` that references all project resources labelled as 'shared'. This aligns with a first use case mapping, where the ISO 19650 stages of data publication can be dynamically generated in a federated project: virtual containers can be created for 'work-in-progress', 'shared', 'published' and 'archived' tags.

Listing 4: SPARQL query to filter a project datasets and return their distributions as LDP containers.

```
CONSTRUCT {?virtualContainer ldp:contains ?downloadURL }
WHERE {
  #_____Dataset discovery starting from Project Access Point____
  #initial source: project access point
  ?aggr lbds:aggregates ?partial .

  #link traversed source: partial project
```

[11] The BCF API, https://github.com/BuildingSMART/BCF-API, accessed 23/02/22

```
  ?partial lbds:hasDatasetRegistry ?dsr .

  #link traversed source: dataset registry
  ?dsr ldp:contains ?ds .

  #_____Subquery for dataset filtering____
  #link traversed source: dataset
  #aggregate all resources with status "shared" (example ontology)
  ?ds ex:publicationStatus "shared" ;
    dcat:distribution/dcat:downloadURL ?downloadURL.

  BIND(UUID() as ?virtualContainer)
}
```

Listing 5: Example result of a dynamically generated LDP container, based on the query in Listing 4. The project ID allows identification of the project on the different stakeholder Pods. Short IDs are used in this listing to keep URLs brief.

```
<> ldp:contains
    <https://pod.architect-office.com/lbd/458107b5/local/d44db838/ds1/mainDist ,
    <https://pod.architect-office.com/lbd/458107b5/local/8c6d7317/ds8/mainDist ,
    <https://pod.engineering-office.de/lbd/458107b5/local/5c08ddb2/ds25/mainDist ,
    <https://pod.architect-office.com/lbd/458107b5/local/34ed9b8d/ds12/mainDist ,
    <https://pod.hvac-studiebureau.be/lbd/458107b5/local/9edbfe3c/ds14/mainDist .
```

3.2. UC 2: Virtual view templates for ICDD containers (ISO 21597)

A similar approach can be taken for generating an ICDD-compliant view of the project.

Listing 6 illustrates the query to extract the resources for the 'Payload Triples' folder. The query for 'Payload Documents' is identical, with the exception of the mediatype ('?mt') filter being changed to 'NOT IN'. The query to retrieve the ontologies is given in Listing 7.

Listing 6: SPARQL query to discover project datasets on a partial project.

```
CONSTRUCT {?virtualContainer ldp:contains ?download }
WHERE {
  [...dataset discovery patterns starting from Project Access Point (cf. Listing 4)]

  ?ds dcat:distribution ?dist .
  ?dist dcat:downloadURL ?download;
    dcat:mediaType ?mt .

  FILTER(?mt IN (
    <https://www.iana.org/assignments/media-types/text/turtle>,
    <https://www.iana.org/assignments/media-types/application/rdf+xml>,
    <https://www.iana.org/assignments/media-types/application/ld+json>
    ))

  BIND(UUID() as ?virtualContainer)
  }
```

Listing 7: SPARQL query to map the ontologies used on the project.

```
  CONSTRUCT {?virtualContainer ldp:contains ?vocabulary }
  WHERE {
    [...dataset discovery patterns starting from Project Access Point (cf. Listing 4)]

    ?ds void:vocabulary ?vocabulary .

      BIND(UUID() as ?virtualContainer)
  }
```

Next is the linkset, which connects sub-document identifiers in an ICDD container. This corresponds with the Reference Registry in the LBDserver, which is similar to ICDD but incorporates metadata and distributions on the one hand and differentiates between enrichment of 'real objects' and enrichment of digital files on the other hand. As the LBDserver approach

for sub-document linking can be considered a superset of ICDD [3], mapping these to ICDD is straightforward (Listing 8).

Listing 8: SPARQL query to generate ICDD links from LBDserver references

```
CONSTRUCT {
    ?concept a ls:Link ;
        ls:hasLinkElement ?le .
    ?le a ls:LinkElement ;
        ls:hasDocument ?distribution ;
        ls:hasIdentifier ?id .
    ?id ls:identifier ?identifier .
}
WHERE {
  [...reference registry discovery patterns (lbds:hasReferenceRegistry)]

  #link traversed source: Reference registry
  ?concept a lbds:Concept ;
    lbds:hasReference ?le .
  ?le lbds:hasIdentifier ?id ;
    lbds:hasDocument ?distribution .
  ?id lbds:hasIdentifier ?identifier .
}
```

The index.rdf file bundles information about the project with a listing of the contained resources and basic metadata (e.g. format, creation date, label and original file name) - information that will be generally stored in the LBDserver metadata as well (Listing 9). Note that the result of this query will not be an LDP container but a single resource directly following the ICDD links.rdf patterns.

Listing 9: SPARQL query (partial) to generate index.rdf for the ICDD container.

```
CONSTRUCT {
    ?index ct:containsDocument ?dist ;
    ct:creator ?creator ;
    ct:description ?projectDescription .

    ?dist a ct:InternalDocument ;
        ct:description ?dsDescription ;
        ct:format ?format ;
        ct:filename ?filename .
}
WHERE {
  [...dataset discovery patterns starting from Project Access Point (cf. Listing 4)]

  #link traversed source: dataset registry
  ?dsr ldp:contains ?ds .
  ?ds dct:creator ?creator ;
    rdfs:comment ?dsDescription ;

  BIND(UUID() as ?index)
  BIND(replace(str(?mt), str("https://www.iana.org/assignments/media-types/"),
                                                    str("")) as ?format)
}
```

3.3. Notes on virtual container views

Of course, metadata can only be filtered when the triple patterns in the query are actually available. However, requirements to metadata can be easily configured (standardised, project-wide or stakeholder-specific) using RDF validation languages such as the SHApes Contstraint Language (SHACL)[12]. Upon creation of datasets, adherence to these shapes should be validated. A discussion on specific workflows for such validation-based resource posting is beyond the scope of this paper.

[12]SHACL, https://www.w3.org/TR/shacl/, visited 18/02/2022

Although the use of LDP containers allows to deliver the data in a standardised format, it is still up to the client to interpret this information, further modify it and present it to the end-user in a meaningful way. However, the use of such containers goes beyond the interfacing aspect. For example, a middleware API can use these mappings as an intermediary mapping to expose the project as an ICDD dump to the outside world. Another example is dynamic access rights management. For example, if a resource is still 'work-in-progress', a *satellite* on top of the LBDserver can set the access rights so only employees of the producing office are allowed to view it. When the publication status changes to 'shared', it can be opened to other stakeholders of the project. The example in Section 4 will further discuss these middleware-oriented mappings.

4. Mapping LBDserver datasets to standardised APIs

In the previous section, client-based views were constructed on LBDserver projects. Although these views can be used for multiple purposes, such as generating a standard-compliant ICDD archive or presenting project information to the user in a specific hierarchy, more can be done when these views are constructed by intermediary agents - headless servers - to expose federated project data as if the project were centralised (Figure 1). This way, backwards compatibility with existing tools and standards can be more easily achieved.

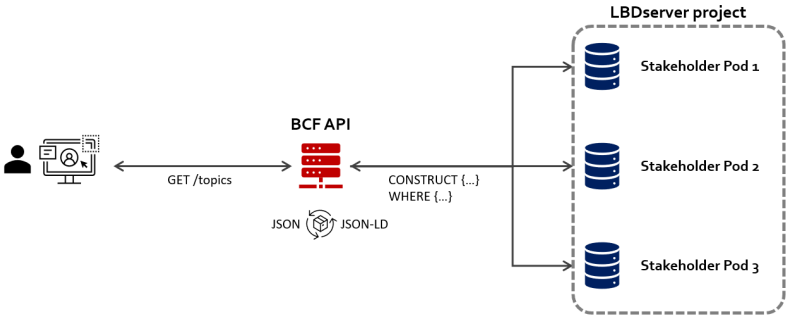

Figure 1: Middleware APIs facilitate the mapping of LBDserver storage patterns to standardised HTTP endpoints and responses, such as the BCF API specification.

4.1. UC3: serving federated project data conform the BCF API

The BCF API proposes a tree-like HTTP endpoint structure to retrieve project issue data. Mirroring this tree to a Solid/LDP-based environment is the topic of [12]. The approach in [12] eliminates the need for an intermediary API, but hard-wires the data storage patterns of BCF into the Pod structures, based on the bcfOWL ontology, an RDF-based version of the BCF data structures[13]. In the below paragraphs we will explore an alternative approach, based on virtual views. This will require an extra middleware service to be set up, but also allows for a more independent way of storing project data. An intermediary service accesses project data on the Pod (exposed via LDP) and remoulds this data in the desired shape via its own REST API. Note that the service is *not* integrated with the Solid Server hosting the Pod - from this perspective, it is both a client (fetching remote data) and a server (exposing data on the Web).

Because the LBDserver allows storage of all kinds of project information, mapping to the BCF API involves creating a subset of the available datasets: e.g., only those which are tagged as `bcfowl:Topic` are relevant in this case. We will consider an exemplary request of the BCF API specification, namely GET TOPICS. Other endpoints for the BCF API can be constructed in a similar way. A standard BCF API response for this request is given in Listing 10.

Listing 10: Example response for a GET TOPICS request according to the BCF API (from https://github.com/BuildingSMART/BCF-API)

```
GET /bcf/{version}/projects/{project_id}/topics

[...otherTopics,
    {
        "guid": "A211FCC2-3A3B-EAA4-C321-DE22ABC8414",
        "server_assigned_id": "ISSUE-00078",
        "creation_author": "Architect@example.com",
        "title": "Example topic 2",
        "labels": ["Architecture", "Heating", "Electrical"],
        "creation_date": "2014-11-19T14:24:11.316Z"
    }
]
```

Additionally, the BCF API relies on JSON for serving its information. The JSON-based RDF format JSON-LD allows us to attach a context to a JSON object, which means that the response can be at the same time compliant to the (contextless) BCF API and avoid losing semantic information. Upon request for '/topics', a BCF server that relies on an LBDserver ecosystem for data storage thus first queries the project with the query listed in Listing 11 and then easily converts the resulting RDF graph to JSON-LD using the context given in Listing 12. If required, further alignment can rely on JSON-LD framing and the JSON-LD framing API[13]. As in this use case the focus does not lie on discovery, but on presenting actual information, the use of 'virtual' LDP containers (cf. Section 3) is omitted. An alternative approach is to just filter the datasets that are `bcfOwl:Topic`-s, using the LDP containers indeed, and then retrieve the entire dataset as JSON-LD.

Listing 11: SPARQL query retrieve `bcfowl:Topic`-s in the federated project.

```
CONSTRUCT {
    ?ds dct:identifier ?identifier ;
        dc:creator ?creator ;
        dct:title ?title ;
        rdfs:label ?label ;
        dct:created ?creationDate .
} WHERE {
  [...dataset discovery patterns starting from Project Access Point (cf. Listing 4)]

    ?ds a bcfOwl:Topic ;
        dct:identifier ?identifier ;
        dc:creator ?creator ;
        dct:title ?title ;
        rdfs:label ?label ;
        dct:created ?creationDate .
}
```

Listing 12: Context to make the results compatible with BCF API response without losing semantics.

```
{
  "@context": {
        "guid": "http://purl.org/dc/terms/identifier",
        "creation_author": {
          "@id": "http://purl.org/dc/terms/creator",
          "@type": "@id"
```

[13] JSON-LD framing, https://w3c.github.io/json-ld-framing/, visited 07/03/2022

```
            },
        "title": "http://purl.org/dc/terms/title",
        "labels": "http://www.w3.org/2000/01/rdf-schema#label",
        "creation_date": "http://purl.org/dc/terms/created"
    }
}
```

5. Discussion and Conclusion

In this paper, we showed how datasets on multiple Solid Pods (the LBDserver ecosystem) can be re-structured to be backwards compatible with existing industry standards. This was done using a programming language-agnostic approach, relying on the properties of SPARQL, RDF and JSON-LD. As illustrated by the use cases, these mappings can serve a mere interface-oriented purpose, e.g. to allow different views on the same datasets (through the use of virtual LDP containers), or a more infrastructural one, hiding LBDserver complexities behind a middleware service layer. We have demonstrated that the data storage patterns of the LBDserver are flexible enough to support basic mappings to divergent standards. However, future research should determine the limits of more complex mappings, e.g., involving the relationships of LBDserver datasets with other datasets. For example: when a BCF *Topic* references its *Comments* and *Viewpoints*, all these instances may be separate LBDserver datasets hosted on different stakeholder Pods. In this case, the use of the LBDserver Reference Registry (not covered in this paper) is expected to offer solace - but this should be tested more thoroughly. Additionally, in such situations, the difference between metadata and actual content is more ambiguous: isn't an issue always metadata on some larger subject? This as well should be clarified in future research.

Nevertheless, this research shows that data storage on Solid Pods does not need to exactly follow a specific storage pattern: the LDP-compliant REST API as implemented in Solid may be only the first layer in a series of mappings. When resources are described by their metadata and their URLs bear no implicit semantics or tags, as is the case in the LBDserver, URLs will be much more stable as they do not need to reflect an ever-changing folder structure (a common challenge for Solid Pods in general). The flattened list of project resources in an LBDserver can be considered 'pluripotent', since creating a new view or folder-structure upon this data or structure is not the responsibility of the Pod (or the LDP resource tree) anymore: these views are just a virtual layer that can be applied at every point in the flow of data exchange. By illustrating this concept holds for multiple industry standards, we hope this research can lower the threshold for further integration of Web- and data-based Building Information Management practices.

Acknowledgments

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

References

[1] E. Mansour, A. V. Sambra, S. Hawke, M. Zereba, S. Capadisli, A. Ghanem, A. Aboulnaga, T. Berners-Lee, A demonstration of the solid platform for social web applications, in: Proceedings of the 25th International Conference Companion on World Wide Web, 2016, pp. 223–226. doi:10.1145/2872518.2890529.

[2] J. Werbrouck, P. Pauwels, J. Beetz, L. van Berlo, Towards a decentralised common data environment using linked building data and the solid ecosystem, in: 36th CIB W78 2019 Conference, 2019, pp. 113–123.

[3] J. Werbrouck, P. Pauwels, J. Beetz, E. Mannens, Lbdserver - a federated ecosystem for heterogeneous linked building data, Semantic Web Journal (submitting).

[4] P. Pauwels, S. Zhang, Y.-C. Lee, Semantic Web Technologies in AEC industry: A Literature Overview, Automation in Construction 73 (2017) 145—-165. doi:10.1016/j.autcon.2016.10.003.

[5] P. Pauwels, W. Terkaj, EXPRESS to OWL for Construction Industry: Towards a Recommendable and Usable ifcOWL Ontology, Automation in Construction 63 (2016) 100—-133. doi:10.1016/j.autcon.2015.12.003.

[6] R. Verborgh, Reflections of knowledge: Designing web apis for sustainable interactions within decentralized knowledge graph ecosystems, 2021. URL: https://ruben.verborgh.org/blog/2021/12/23/reflections-of-knowledge/.

[7] J. Hendler, F. Gandon, D. Allemang, Semantic Web for the Working Ontologist: Effective Modeling for Linked Data, RDFS, and OWL, Morgan & Claypool, 2020.

[8] J. Werbrouck, R. Taelman, R. Verborgh, P. Pauwels, J. Beetz, E. Mannens, Pattern-based access control in a decentralised collaboration environment, in: Proceedings of the 8th Linked Data in Architecture and Construction Workshop, CEUR-WS. org, 2020.

[9] M. Bonduel, A framework for a linked data-based heritage bim, 2021. URL: https://lirias.kuleuven.be/handle/123456789/674476.

[10] R. Verborgh, R. Taelman, Guided link-traversal-based query processing, arXiv preprint arXiv:2005.02239 (2020).

[11] M. Senthilvel, J. Oraskari, J. Beetz, Implementing information container for linked document delivery (icdd) as a micro-service, in: EG-ICE 2021 Workshop on Intelligent Computing in Engineering, Universitätsverlag der TU Berlin, 2021, p. 66.

[12] J. Oraskari, O. Schulz, J. Werbrouck, J. Beetz, Enabling interoperable issue management in a federated building and construction sector, in: EG-ICE 2022 Workshop on Intelligent Computing in Engineering, submitting.

[13] O. Schulz, J. Oraskari, J. Beetz, bcfowl: A bim collaboration ontology (2021) 142–153.