LBDserver - a Federated Ecosystem for Heterogeneous Linked Building Data

Jeroen Werbrouck [a,c,*], Pieter Pauwels [b], Jakob Beetz [c] and Erik Mannens [d]
[a] *Department of Architecture and Urban Planning, Ghent University, Belgium*
E-mail: jeroen.werbrouck@ugent.be
[b] *Department of the Built Environment, TU Eindhoven, The Netherlands*
[c] *Chair of Design Computation, RWTH Aachen, Germany*
[d] *IDLab - Internet Technology and Data Science Lab, Ghent University, Belgium*

Abstract. As the application of Linked Data technologies in the Architecture, Engineering and Construction (AEC) industry gains momentum, prospects for an interdisciplinary, Web-based BIM practice become more and more realistic. Although several modular Linked Building Data (LBD) ontologies have already been developed and interlinked to address specific topics in digital buildings, infrastructures to actually test and use them in an accessible, 'BIM-like' fashion, are scarce. In this paper we propose the LBDserver, an extendable web-framework for management of federated, heterogeneous (building) project data. In contrast with current-day centralised Common Data Environments, this decentralised solution is stakeholder-oriented, where disparate project information resides with the stakeholder, to be shared in a fine-grained way with other consortium members, and linked with other heterogeneous datasets on the Web. We combine the Solid initiative for Web decentralisation with the recent industry standard *Information Container for linked Document Delivery* (ICDD), proposing a 'federated CDE' infrastructure for serving heterogeneous AEC project datasets in a federated, access-controlled and scalable manner. We validate the proposed framework for completeness using an existing project in Ghent, Belgium.

Keywords: Linked Building Data, Solid, Multi-models, Module federation, Web-based BIM

1. Introduction

1.1. The AEC industry as a patchwork of stakeholders

Contrary to many industries, most 'end products' in the AEC industry (Architecture, Engineering and Construction) are unique: a building project always lives in the real world, is designed for or adapted to a specific use case, affected by its natural surroundings and human infrastructure. People from many professional domains interact with this asset in all of its life cycle phases, ranging from direct stakeholders such as architects and engineers, commissioners, facility managers, and inhabitants to indirect partners such as governmental agencies and product manufacturers. A construction project consortium is thus a multidisciplinary patchwork of actors, where each one contributes context, knowledge, and resources in order to realise and maintain the eventual building. Therefore, and because every stakeholder will most likely be involved in multiple projects at the same time, the AEC practice can be seen as a double-layered patchwork.

As so many disciplines are involved, specialised software packages and dedicated data models exist for different tasks and phases. Reaching a reasonable degree of interoperability between those data models is one of the main

[*]Corresponding author. E-mail: jeroen.werbrouck@ugent.be.

challenges in current BIM research. The terms 'Closed BIM' and 'Open BIM' refer to the current solutions for handling such data exchange [1]. Closed BIM means that an integrated ecosystem provided by one vendor is used, thus developing a project based on a single data schema (or at least a set of heavily integrated schemas). Depending on the size of a project and the envisaged usage of the digital model, this can be a viable solution. However, as the project grows larger or when the BIM model is to be used in other phases than design and construction (maybe for purposes that are yet unknown at the start of the project), one can easily see that the entire software stack used in a project will most likely be an eclectic mix of tools: different stakeholders will use different software packages from various vendors, depending on their discipline, resources and the specific tasks they have been assigned. In such cases, the paradigm of 'Open BIM' will be the best choice to enable collaboration between stakeholders: collaboration and data exchange takes place using a neutral and open data format that can be exported and imported by various software tools. Sub-models, often structured according to the industry standard IFC (Industry Foundation Classes [2]), can now be compared and checked for consistency with one another and the project ambitions in general.

Even in an Open BIM environment, however, IFC has a limited scope: it is a fixed-size set of classes and properties, oriented towards the description of a built asset. The integration of topics or datasets that are not easily described in IFC in a machine-readable way is still very difficult; an integrated, interoperable knowledge base of asset data with contextual information such as sensor streams, user data, geospatial information, and governmental regulations remains largely fictitious. The use of such data- (and Web-)based building models is often situated in the 'final' stage of BIM, corresponding with BIM Level 3 in the iconic wedge of the 'BIM levels of Maturity' (Figure 1) [3]. The reason is evident: one focus of BIM Level 3 lies on expanding the use of BIM towards data management in the entire Building Life Cycle (BLC), using a network of smaller, interoperable services rather than 'monolithic' BIM authoring tools - communications between such services preferably taking place in a (semi-)automatic fashion. For such an environment to work, the use of open data models is essential - after all, services need to be able to work on many distinct datasets - as well as a paradigm shift towards data-based BIM instead of document-based BIM. This does not automatically mean that the use of 'documents' is entirely out of question, but rather that the data they contain is interoperable with other project data. In other words, that documents should be less thought of as 'data silos' and more as permeable collections that can reference each other's contents and can likewise be referenced in other datasets living on the Web. Depending on the use case, it may be necessary to go extremely fine grained (e.g., in terms of referencing individual asset objects and semantic properties) or it could be that referencing a binary blob is enough (e.g., when linking point clouds).

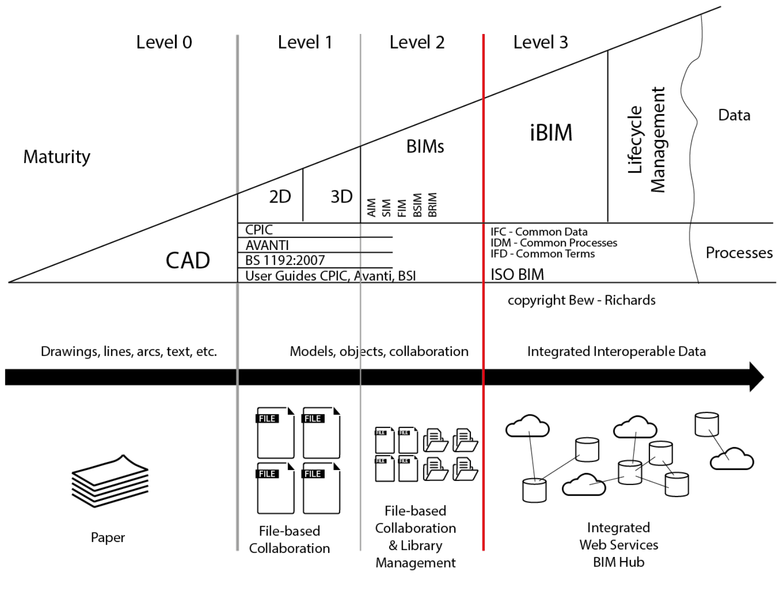


Fig. 1. The BIM maturity levels as defined by the British Standards Institution (BSI) (source: PAS 1192-2:2013)

1.2. Implementing BIM Level 3 using semantic web technologies

Often, Semantic Web and Linked Data technologies are identified as apt technologies to address the challenges posed by implementing BIM level 3, as interoperability, interdisciplinary data mapping, and data federation are amongst their core benefits [4]. The Semantic Web is a domain-agnostic data representation layer on top of the 'classic' World Wide Web, using the Resource Description Framework (RDF)[1]. Specific *Web ontologies* (data models) can be created to describe specific knowledge in a federated fashion (i.e., data snippets can be related independent of the server that hosts this data), allowing to connect to information that is described by other ontologies on the Web. The potential of Semantic Web technologies for describing the built environment is becoming widely acknowledged, resulting among others in a Semantic Web-based representation of the IFC schema (ifcOWL) [5, 6]. However, ifcOWL's complexity is as high as the 'classic' IFC representations based on Express (IFC-SPF) and XSD (IFC-XML), or even higher. This makes it quite difficult to understand, use, and implement - even more when connecting to adjacent disciplines (which often involves a mapping of terminological concepts). Recent research in the use of Semantic Web technologies for the AEC industry is therefore focusing more on developing small, modular ontologies, each one defining a limited set of classes and properties: focused on a particular topic, but easily linked to one another to address specific use cases [7–10]. These data models are often called 'Linked Building Data' (LBD) ontologies. Because of their limited scope, connecting with adjacent data models becomes more straightforward: for example, with geospatial data [11], product manufacturer data [8], sensor [12] and IoT data [13, 14], or governmental regulations [15]. At the same time, expressing knowledge in RDF (hosted in triple (or quad) stores) often increases its complexity and processing time, and sometimes adds little value to the overall data. This is, for example, the case with non-structured (or semi-structured) datasets, such as imagery, point clouds, and geometry. This inherent heterogeneity is acknowledged by both academia and industry, but efficient solutions for *dynamic* management of such resources are currently lacking.

Such approach, which integrates real-time serving and manipulation of project data by specialised databases (triple store, noSQL, SQL, ...), contrasts with *static* management of project resources, which is addressed by the Information Container for Linked Document Delivery (ICDD) [16]. ICDD uses RDF to offer a file-based 'multi-model' [17] of the project: a static ZIP file containing all the available project information. However, this information will not synchronise anymore with changes to the project data. In phases where the design changes rapidly, e.g., during the design stages, this can become problematic. For example, to just move a wall might seem harmless at first sight, but whenever a change is made in one sub-model, a cascade of actions in the other project files needs to be triggered to ensure data consistency. Enabling this trigger is only possible in a *dynamic*, highly integrated project environment. We thus identify the need for an open-source one-stop, data-based CDE that allows active discovery, creation, manipulation, and linking of heterogeneous project data, with a focus on the use of Linked Building Data.

1.3. Towards fully federated CDEs

Whilst such an ecosystem would function as intended when taking the form of a centralised Common Data Environment (CDE), the use of Semantic Web technologies allow us to take a *decentralised* approach, which effectively addresses the dichotomy of the double patchwork that persists in current-day central CDE solutions: in order to collaborate on the 'single source of truth' for the project, it is required that offices upload their contributions to a project-specific central cloud server, which acts as a middle-man to make project information available to the other stakeholders. Because CDEs are project-oriented and not stakeholder-oriented, this has as a consequence that offices may need to work in different CDEs for the different projects they work in. Apart from the fact that most AEC offices will not have the resources to master all these ecosystems, and the data ownership issues related to centralising decentrally produced information on a single service provider, uploading your projects to different ecosystems essentially means that they are wrapped between diverse APIs (making it difficult to integrate data from multiple projects) and that separate user accounts for each CDE need to be maintained. Moreover, on a service level, third party apps need to comply to the APIs of CDE providers, creating dependencies on proprietary software on the one hand and stealing time and resources that could be spent on improving the actual service.

[1]https://www.w3.org/RDF/

The technologies for a 'federated CDE' are available, though [18]. In such an environment, Linked Data technologies allow stakeholders to self-host all their contributions to all the projects they participate in, and fine-grainedly control who may access certain project information (e.g., other stakeholders, the public, product manufacturers, etc.). Upcoming open Web specifications such as WebID-OIDC[2] allow the creation of a decentral, self-hosted Web identity - in other words, a username for the entire Web, to be used to prove access rights to specific data on the Web. In our case, this means an office can authenticate to the servers of other project participants to access their (filtered) project information, but also that online services with correct access rights can combine private project information from various stakeholders with open datasets on the Web and present end-users with powerful ways to interact with this data - without mediation of a central, project-external CDE provider or the need for its permissions to use their API. Note that outsourcing the hosting of data to an external firm (e.g., for security or data redundancy) remains an option - key is that data storage and services can be separated, to (1) allow a situation where 'anyone can say anything' about the asset in a machine-readable format and (2) drive 'permissionless innovation' for third party services - independent from the actions allowed or implemented by the APIs of commercial CDE platforms.

1.4. Research objectives

Research towards the use of Semantic Web and Linked Data technologies in the AEC domain is well-established. Its focus has been largely on the development of domain models, and - although the potential of the Semantic Web for the largely federated AECO industry is widely acknowledged [4] - the majority of the work in Linked Building Data is based on more 'central' strategies and thus follows the patterns of 'traditional' CDEs: local triple stores, single servers hosting all project information or comprehensive, file-based project containers. These solutions centralise *project-specific* information on one hand, but do not prevent that information of multiple projects is scattered across multiple disparate ecosystems. In this paper, we aim to address this by proposing a stakeholder-centric approach in data storage, whilst using Web federation technologies to organise a construction project as a network of access-controlled and public resources. By enabling an ecosystem that follows such an approach, construction projects become by default *federated multi-models*, with resources that can be aggregated flexibly, depending on the use case and query parameters. To achieve this, the ecosystem needs to respond to the following system requirements:

1. The CDE shall be federated - a stakeholder data vault can be initiated at any server with access to the internet.
2. Authentication and authorization shall take place in a decentral manner.
3. The focus of the ecosystem shall be entirely on metadata structures - as it should not make any assumptions on the data models (e.g., ontologies) that structure project information.
4. The ecosystem shall support multiple database environments (e.g., triple stores, timeseries databases, SQL stores, key-value stores, etc.) to serve heterogeneous datasets in a manner that fits the datatype.
5. The ecosystem shall support *sub-document* linking and alignment of heterogeneous resources to other data on the Web.

We propose an infrastructure for a *federated* CDE, which combines access-controlled, heterogeneous project information and (open) contextual information into a federated knowledge graph. Stakeholder offices keep full control over the datasets they produce and make available, over the issues they communicate to one another and over the services they use to streamline their tasks and workflows within the project. The ability to establish sub-document links in heterogeneous datasets is essential to freely enrich project data, whether expressed using RDF or not. We describe both 'static' data patterns and 'dynamic' technological implementations. Based on the novel Solid ecosystem[3] and its specifications for decentralised identity and storage, metadata structures are devised for real-time serving and sub-document linking of heterogeneous, federated project-specific and contextual data. A prototypical code-base will be made available under the name 'LBDserver'.

[2]https://github.com/solid/webid-oidc-spec
[3]https://solidproject.org/

1.5. Paper overview

Section 2 includes an overview of background technologies, previous work, and related initiatives. The envisaged larger components of the LBDserver framework itself are introduced in Section 3 with a focus on data discovery in a federated network, and the aggregation of individual contributions ('partial projects') of stakeholders into a 'federated project'. The envisaged internal data structures of a partial project are explained in Section 4. The framework and interactions are then demonstrated in Section 5, using a real-world example. The paper finishes with a discussion of the proposed ecosystem, including the challenges that lie ahead (Section 6).

2. Related Work

The situation that was outlined in the introduction has been the scene of many technical implementations and investigations. The concept of Linked Data has recently been a key set of technologies of relevance, but also other concepts like federated access control, data catalogs, and Read-Write data storage such as Solid Pods are important to the infrastructure we propose in Section 3, and will hence be covered prior to the discussion on the LBDserver organisational patterns. For a general introduction to the basic principles of the Semantic Web, the reader is referred to [19, 20]. As Linked Data for the AECO industry ("Linked Building Data", LBD) gains importance in the field and naturally aligns with a federated CDE, this section starts with a brief overview of the foundations and logics behind LBD (Section 2.1). Next, we review existing technologies for aggregating and relating heterogeneous datasets (Section 2.2), as well as the core concepts behind the Solid ecosystem (Section 2.3). Finally, Section 2.4 reviews related initiatives with aims similar to ours.

RDF prefixes used throughout this paper and their corresponding namespaces are listed in Listing 1. Definitions in context of this paper will be structured using a custom LBDserver vocabulary (prefix 'lbds').

Listing 1: Namespaces used in this publication and their prefix abbreviations

```
@prefix lbds: <https://w3id.org/lbdserver#> #The LBDserver vocabulary

@prefix acl: <http://www.w3.org/ns/auth/acl#> .
@prefix bot: <https://w3id.org/bot#> .
@prefix cdc: <https://w3id.org/cdc#> .
@prefix cdo: <https://w3id.org/damagemodels/cdo#> .
@prefix dcat: <http://www.w3.org/ns/dcat#> .
@prefix dot: <https://w3id.org/dot#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix ldp: <http://www.w3.org/ns/ldp#> .
@prefix owl: <https://www.w3.org/2002/07/owl#> .
@prefix pim: <http://www.w3.org/ns/pim/space#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix schema: <https://schema.org/> .
@prefix solid: <http://www.w3.org/ns/solid/terms#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
```

2.1. Linked Building Data

Research in the field of Linked Data for architecture and construction has maintained a direct link with standards such as IFC [5, 21], eventually resulting in a BuildingSMART-approved IFC schema 'ifcOWL' [6, 21]. However, as IfcOWL is designed to be as compatible with 'standard' IFC as possible, the complexity commonly associated with IFC persists. Therefore, focus for developing domain ontologies has recently shifted towards small, modular ontologies, as advocated by the W3C Linked Building Data Community Group (Figure 2) [22]. The cornerstone of this modular approach is the *Building Topology Ontology* (BOT) [7], containing only the definitions to describe topological relationship of a building: a `bot:Site` contains one or more `bot:Building` instances (prefix `bot:hasBuilding`), which can be further decomposed into `bot:Storey` and `bot:Space`, eventually containing instances of `bot:Element`. Furthermore, it is possible to define instances of `bot:Interface` to represent interfaces between zones and elements (e.g., for heat transfer, indoor navigation, etc.).

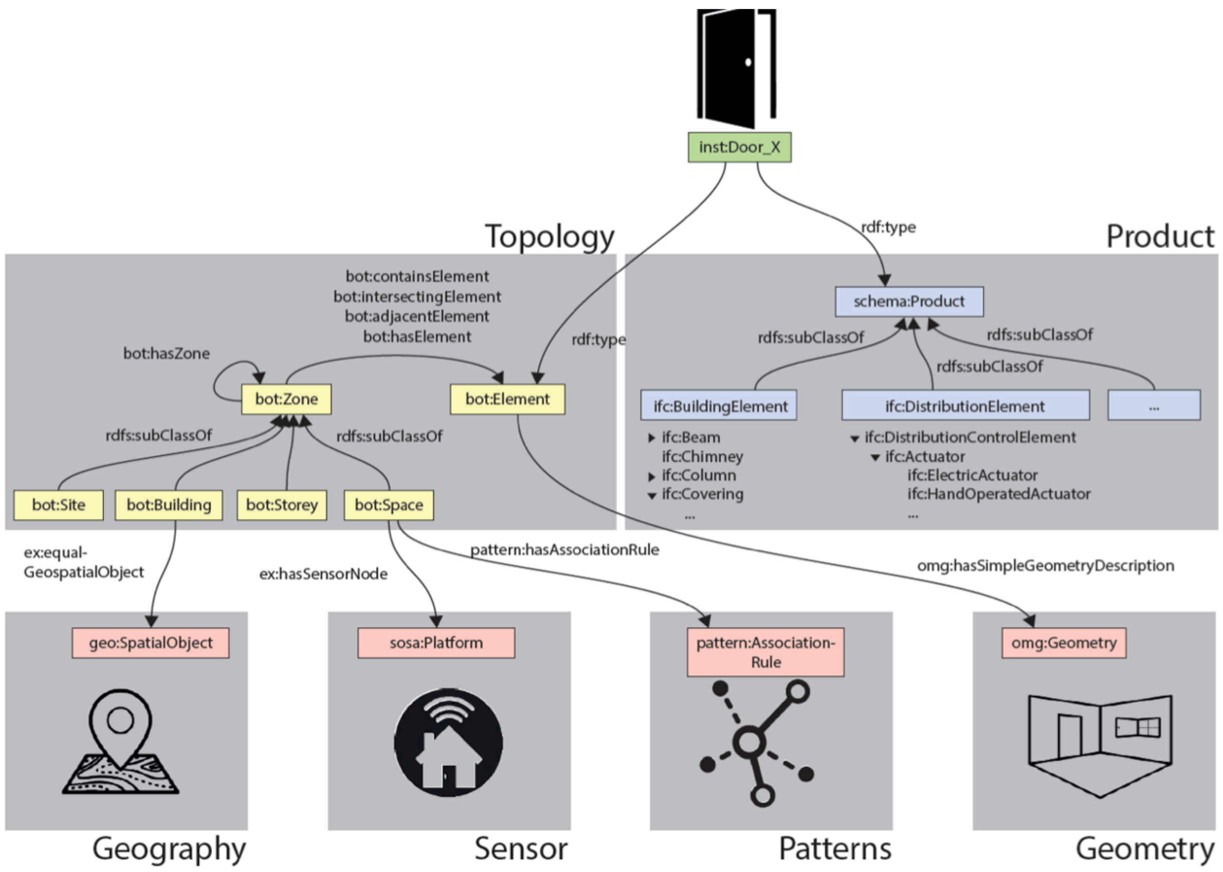


Fig. 2. Connecting modular Linked Building Data ontologies (source: [22])

With the topological description as the spine of the model, other ontologies can further enrich it, depending on the use case. For example, regarding element classification there are the *BuildingElement Ontology*[4] (BEO) and the *DistributionElement Ontology*[5] (MEP), respectively based upon the IfcBuildingElement and IfcDistributionElement subtrees in the IFC specification. Other classification systems can be used in parallel, such as the Getty Art and Architecture Thesaurus[6] (AAT) or Wikidata[7]. Building component description can be done using the *Building Product Ontology*[8] (BPO) [8], while damage classification is covered by the *Damage Topology Ontology*[9] (DOT) [10] and

[4]https://pi.pauwel.be/voc/buildingelement/index-en.html
[5]https://pi.pauwel.be/voc/distributionelement/index-en.html
[6]http://vocab.getty.edu/aat/
[7]https://www.wikidata.org
[8]https://www.projekt-scope.de/ontologies/bpo/
[9]https://alhakam.github.io/dot/

sensor data can be linked to the project using SSN[10] [23] and SAREF[11] [13, 14]. It is clear that this approach of combining modular ontologies is much less bound to the limits of a single 'monolithic' data model - as the data model can be extended whenever the use case requires this. Moreover, as RDF is based on URIs and URLs, the use of modular LBD data models naturally fits the idea of a federated building project graph that semantically connects information on the Web.

2.2. *Connecting Federated, Heterogeneous Datasets*

In a digital construction project, many resources may actually refer to one and the same object. A window in the building may find itself semantically described in graphs produced by different stakeholders in the project, and be represented in multiple images, point clouds and geometries (2D and 3D). The overall set of resources representing a digital building model is often denoted as a 'multi-model' [17]. Several approaches to interrelate the heterogeneous resources in such multi-models exist; some are AEC specific - others are generic specifications for the Semantic Web. In this section, we briefly introduce three specifications, each one aimed at discovering and connecting data on a different granularity level.

2.2.1. *Information Container for linked Document Delivery (ICDD)*

ICDD is an industry-initiated standard (ISO 21597) for exchanging heterogeneous asset data during its life cycle, in the form of multi-models. It can be situated on the edge between BIM Level 2 and 3, as it makes use of Linked Data, but at the same time recognises the fact that the industry is still largely file-oriented. Data is structured in a ZIP-compressed folder structure (.icdd), containing project files (i.e., the 'Payload documents' folder), RDF-based metadata about those files (i.e., the 'Payload triples' folder), an *index.rdf* graph, semantically relating project resources to one another, and a folder containing the vocabularies used within the ICDD multi-model. Note that an ICDD container is essentially a static collection of project data, it is not a database itself and represents the asset documentation at a fixed point in time. However, the ontological patterns for sub-document linking are independent from the data storage patterns. These patterns describe how to link resources together on a sub-document level (e.g., relating an IFC element to a spreadsheet cell), while leaving the content-type of these files open. This is done via `ls:Link` instances, which aggregate `ls:LinkElements`. A `ls:LinkElement` on its turn links to an `ls:Identifier` (identifying the sub-document snippet) and the document of interest (`ls:hasDocument`). Finally, the `ls:Identifier` instance may be enriched with additional information as how to interpret the identifier (not specified by the standard), and finally a literal value or URI representing the identifier (Figure 3).

2.2.2. *Data Catalog Vocabulary*

Where ICDD is an industry-oriented standard aimed at connecting datasets at sub-document level, the Data Catalog Vocabulary (DCAT)[12] is *an RDF vocabulary designed to facilitate interoperability between data catalogs published on the Web* [25]. DCAT defines a generic (domain-agnostic) way of aggregating federated datasets in a `dcat:Catalog`. A catalog (`dcat:Catalog`) aggregates datasets (`dcat:Dataset`); a `dcat:Dataset` instance defines the metadata about a dataset and may point in turn to several distributions (`dcat:Distribution`) - which essentially represent the same dataset but differ from one another, e.g., concerning their data type. Once a `dcat:Distribution` is identified, the actual data may be retrieved by following either `dcat:downloadURL` (retrieve a dump of the dataset) or `dcat:accessURL` (access the dataset via a database endpoint, e.g., SPARQL) (Figure 4). Such endpoints can be further described via `dcat:Service` instances, e.g., to indicate whether the service conforms to SPARQL standards (`dct:conformsTo`) or the datasets provided by the service (`dcat:servesDataset`).

DCAT features a decentralized approach to publishing data catalogs, and hence federated search for datasets across catalogs on multiple endpoints is possible using the same query mechanism and structure. Ontological extensions for using the DCAT vocabulary to describe construction projects have been proposed in [26], such as the Construction Dataset Context (CDC) ontology[13].

[10]https://www.w3.org/TR/vocab-ssn/
[11]https://saref.etsi.org/core/
[12]https://w3c.github.io/dxwg/dcat/
[13]https://mathib.github.io/cdc-ontology/

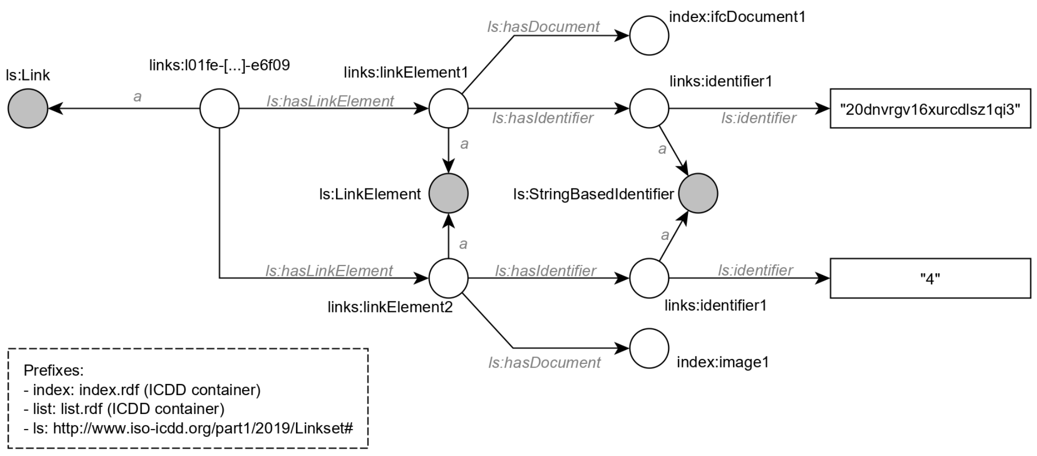


Fig. 3. Relating sub-document geometric IDs to a semantic object using ICDD - graph representation [24]

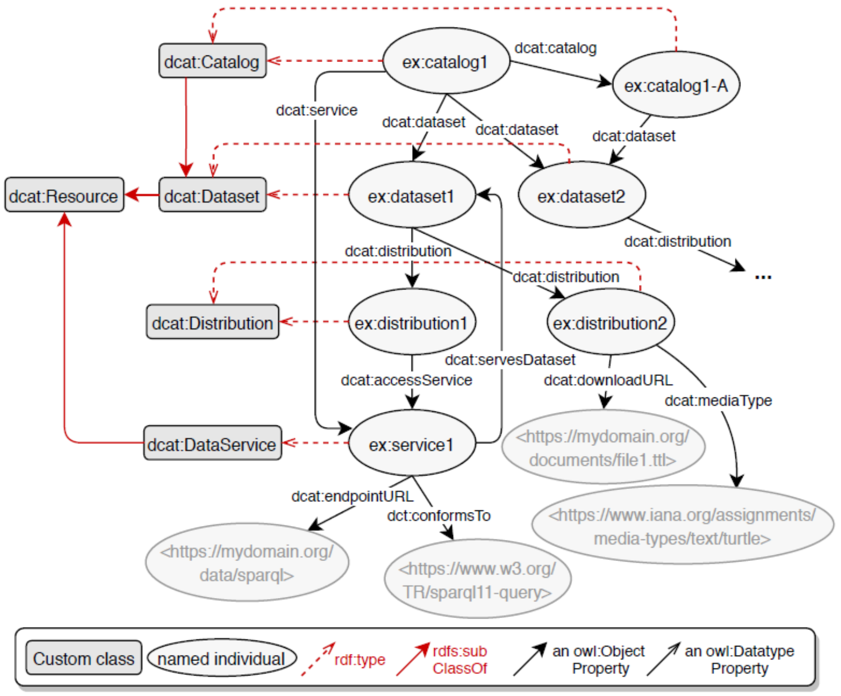


Fig. 4. Core DCAT concepts, figure from [26]

2.2.3. *Linked Data Platform*

While in DCAT the focus lies on describing metadata of datasets, it does not describe how to access or modify this metadata. The Linked Data Platform specification[14] (LDP), on the other hand, presents guidelines for storing of and interaction with heterogeneous Web resources, and presents a basis for a read-write Web of data using HTTP. Based on the type of container, membership of resources and containers can

[14]https://www.w3.org/TR/ldp/

be either predefined (`ldp:BasicContainer`) or left to the implementer (`ldp:DirectContainer` and `ldp:IndirectContainer`) to offer more (domain-specific) flexibility in defining custom relationships. LDP can be compared with a graph-based file system, where folders contain pointers to where the datasets are stored rather than containing the datasets themselves. The use of the LDP specification in conjunction with ICDD as a domain specific standard has been studied in [27]. Potential alignment of LDP and DCAT, combining the benefits of both, is proposed in [28].

2.3. The Solid Ecosystem

When datasets are not openly accessible on the web, but only available to selected agents, an authentication mechanism needs to be in place. For example, Web-based construction projects will most likely consist of restricted project-specific data and open contextual datasets. Most common techniques over the web rely on authentication mechanisms that are added to the middleware and/or backend of a web service implementation. These authentication mechanisms shield and secure the databases (SQL, NoSQL, triple stores, document stores, etc.) from random access, while only allowing authenticated users through, for example, tokens (2-way handshake). Most developers, also Semantic Web developers, are expected to set up such authentication systems themselves using traditional means. Different is the Solid project for Web decentralisation [29], which is based on the LDP specifications but adds an authentication layer on top of it. This authentication layer is based upon the concept of a WebID; an URL that uniquely identifies an agent on the Web. Consequently, this WebID can be used as a decentral 'username'. Most commonly, a WebID dereferences to a self-descriptive, public RDF document (the 'card'), which enriches the WebID with basic information about the represented agent (Listing 2).

Listing 2: Example solid "card" (Turtle syntax). The WebID is a specific resource in the card that can be semantically enriched with RDF

```
<https://pod.inrupt.com/example/profile/card#me>
        foaf:name                 "Max Mustermann" ;
        rdf:type                  schema:Person , foaf:Person ;
        pim:preferencesFile       <https://pod.inrupt.com/maxmustermann/settings/prefs.ttl> ;
        pim:storage               <https://pod.inrupt.com/maxmustermann/> ;
        solid:account             <https://pod.inrupt.com/maxmustermann/> ;
        solid:publicTypeIndex     <https://pod.inrupt.com/maxmustermann/settings/publicTypeIndex.ttl> ;
        solid:privateTypeIndex    <https://pod.inrupt.com/maxmustermann/settings/privateTypeIndex.ttl> ;
        solid:oidcIssuer          <https://broker.pod.inrupt.com/> ;
        ldp:inbox                 <https://pod.inrupt.com/maxmustermann/inbox/> ;
        lbds:hasProjectRegistry    <https://pod.inrupt.com/maxmustermann/lbd/> .

<https://pod.inrupt.com/maxmustermann/profile/card> #if the resource is actually the same as the document, often <> is used .
        foaf:primaryTopic   <https://pod.inrupt.com/maxmustermann/profile/card#me> ;
        foaf:maker          <https://pod.inrupt.com/maxmustermann/profile/card#me> ;
        rdf:type            foaf:PersonalProfileDocument .
```

Building upon this WebID concept, the WebID-OIDC protocol, which is developed in the context of Solid, combines the decentral flexibility of the WebID with the well-established standard OpenID Connect[15] (OIDC). Standard OIDC allows to outsource authentication to an external 'identity provider' (IDP) (e.g., Google, Facebook, Github, Autodesk, etc.), so one may authenticate to multiple services with only one account hosted by an identity provider. The Solid specifications and protocols expand on this, and allow to set up *a personal identity provider* and authenticate to any Solid-enabled service - hence identity verification can happen without large companies acting as a middle-man.

As Solid is based on the LDP specification, a Solid server may host both RDF and non-RDF data. Metadata and access control are expressed using RDF ontologies. Access control documents ('.acl') relate specific `ldp:Resources` and `ldp:Containers` to specific (groups of) actors via their WebID, using the Web Access Control[16] (WAC) specification and ontology. In the default ACL vocabularies, four access modes are defined: `acl:Read`, `acl:Write`, `acl:Append`, and `acl:Control`, where `acl:Control` means the ACL resource itself may be manipulated. An example ACL document is given in Listing 3.

[15]https://openid.net/connect/

[16]https://github.com/solid/web-access-control-spec

Listing 3: Example ACL document, regulating access to a specific LDP container and its resources (unless these have a dedicated ACL document of their own)

```
# This folder and its resources are readable by the public
<#public>
    a acl:Authorization;
    acl:agentClass foaf:Agent;
    acl:accessTo <./>;
    acl:default <./>;
    acl:mode acl:Read.

# The owner has full access to every resource in their pod.
# unless specifically authorized in other .acl resources.
<#owner>
    a acl:Authorization;
    acl:agent <http://pod.myoffice.org/profile/card#me>;
    acl:accessTo <./>; # Set the access to the root storage folder itself
    acl:default <./>; # All resources will inherit this authorization, by default
    acl:mode acl:Read, acl:Write, acl:Control. # The owner has all of the access modes allowed
```

The second Solid-specific resources are metadata resources (.meta). Metadata resources are related to specific LDP containers; dereferencing the container URL yields the .meta resource, with pointers (`ldp:contains`) to the resources and subcontainers they aggregate and indications about modification time (`dc:modified`). At the moment of writing, the structure of metadata descriptions is still under development. Current Solid implementations like the Solid Community Server[17] actively restrict direct editing of .meta files as to be fully compatible with the specifications and prevents external agents to directly edit metadata files. However, in the case of federated AEC projects, the use of .meta resources seems a promising way of adding metadata about a resource such as publication status, topic, etc. in a straightforward fashion. Therefore, in our proposed LBDserver architecture and proof-of-concept, we bypass this restriction - with this remark that we only perform operations which modify (write/delete) custom statements.

2.4. Related platforms and implementations

This paper is the continuation of previous research on enabling technologies for a federated CDE. It was proposed in [18] to use the Solid infrastructure as a means to create a federated Common Data Environment, and to map the specific ‘decentral’ nature of the AEC industry to the available technologies offered by the Solid specifications. This research was extended in [30], which describes a methodology for pattern-based access control to AEC datasets. In contrast with username- or group-based approaches, this approach allows to specify the properties someone should have to access data (e.g., ‘the architect of the project’), based on the SHApes Constraint Language (SHACL)[18] standard. An initial proposal for patterns to connect heterogeneous datasets in a federated ecosystem was described in [31], based on data conversion patterns described in [32]. However, the proposed framework still depends on a *central project Pod*, which hosts a stakeholder graph and a general mapping registry. Furthermore, it mainly deals with file-based storage of datasets. While such file-based ‘dumps’ can be downloaded, parsed and queried client-side, this comes at a performance cost, which will rapidly rise when the project grows larger. Moreover, the sub-document linking patterns in [31] focus on linking geometry, but do not provide a generic way to link heterogeneous information. While describing the LBDserver architecture in the following sections, certain aspects of the ecosystem described in [31] will be revisited in a more generic way, as to provide a more scalable approach. Other frameworks and initiatives as listed below have overlapping ambitions with the LBDserver framework discussed in this paper.

2.4.1. BIMserver

The BIMserver[19] [33] is an IFC model server, exposing IFC-compliant BIM models not as files but in an object-oriented approach. Using IFC as its data model, the BIMserver is able to deal with multiple use cases in the present AEC industry. However, including and semantically combining non-IFC compliant, heterogeneous resources is not possible - neither is a federated management of IFC models. Third party tools (so-called ‘BIM bots’) can be developed on top of this BIMserver, to perform specific actions on the shared BIM model. Interoperability between

[17]https://github.com/solid/community-server

[18]https://www.w3.org/TR/shacl/

[19]https://github.com/opensourceBIM/BIMserver

those bots is guaranteed because of the shared data model, in this case IFC. This concept of small, AEC-oriented microservices (the 'BIM bots') that act as specialised windows on a project information, rather than enclosing their own 'closed' databases, directly influences the architecture of the LBDserver.

2.4.2. *Project SCOPE*

An initiative which does use micro-services powered by Linked Building Data is the SCOPE project[20]. In SCOPE, a system architecture is proposed that allows micro-services to interact with full RDF-based BIM models stored in a triple store [34]. These micro-services (e.g., a Revit-to-RDF exporter, a rendering service) are then bundled in a Docker[21] network and exposed via a dedicated API gateway so users can access them in a unified way, e.g., to be used for structural analysis in other tools. This work is very similar to the proposed LBDserver, yet relies on traditional means for the implementation (no Solid, standard authentication means, single programming language, etc.).

2.4.3. *DRUMBEAT Platform*

In the DRUMBEAT project, which finished in 2017, an infrastructure is devised for publishing and linking BIM-related data on the 'Web of Building Data' [35]. 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. The main semantic models for DRUMBEAT are IFC and IfcOWL - the concept of modular LBD ontologies was still very young when the DRUMBEAT project ended. Since the finishing of this project, several novel technologies and new standards were established, both on a domain-agnostic level and AEC-specific.

3. Discovery and Aggregation of Federated Construction Projects

As mentioned in Section 1, with the LBDserver project, we propose a framework for the organisation of asset data in a federated manner, effectively creating a federated Common Data Environment. Such environment contrasts with traditional 'centralised' CDE's in that both layers of the patchwork can be taken into consideration. Firstly, a *stakeholder-centric layer* is maintained because any participating actor can self-organise their project data, and allow others to refer to this data when relevant. Using the semantically rich relationships provided by RDF and OWL technologies, abstraction is made from the fact that data may reside on different servers – the construction project effectively becomes a federated Linked Data graph, a dereferenceable catalog accounting for the *project-centric layer* in the patchwork.

In this Section, we will identify the components we deem necessary for such framework to function properly on macro-level. Firstly, we introduce 'aggregators' as a generic way to bundle and discover federated project information on the Web, using the LDP specification. Then, we add an authentication layer on top of it by combining the concept of aggregators with infrastructure offered by the Solid ecosystem. 'Micro-level' organisation of projects, i.e., how we propose project data to be stored in a Pod-based project repository, will be covered in Section 4.

3.1. *Aggregating Project Data on the Web*

When information about a project is federated over multiple servers, and the boundaries of this project may change depending on the use case, a first requirement is to be able to discover project information. To allow such discovery in a scalable way, we propose the use of *aggregators*: a collection of pointers to relevant datasets on the Web. This collection may be automatically generated, based on specific parameters or queries, or manually curated. For now, we only focus on discovery and make abstraction from the storage of project data: this will be the subject of Section 3.2.

To semantically construct such aggregators, we suggest the use of LDP containment triples (`ldp:contains`). Although mostly used for resources that are *effectively* hosted by the container on a specific server (as is the case

[20]https://www.projekt-scope.de
[21]https://www.docker.com/

with current Solid implementations), the use of URIs implies that LDP resources can be 'virtually' contained as well. Because LDP containers are also LDP resources, it is possible that aggregators (virtually) aggregate other aggregators - collections of collections. This 'matryoshka' principle enabled by the LDP specification gives us a simple yet powerful way to discover collections of information in a scalable way. Note that access to the referenced datasets is still the responsibility of the dataset's owner: an aggregator only references resources by their URL.

Figure 5 illustrates this with the following example: a project P1 has contributions from two separate offices A and B. Every stakeholder can create a 'personal' aggregator for the project, which allows a client to find and query project information based on this single 'project access point', e.g., a folder on their Solid Pod (Section 3.2).

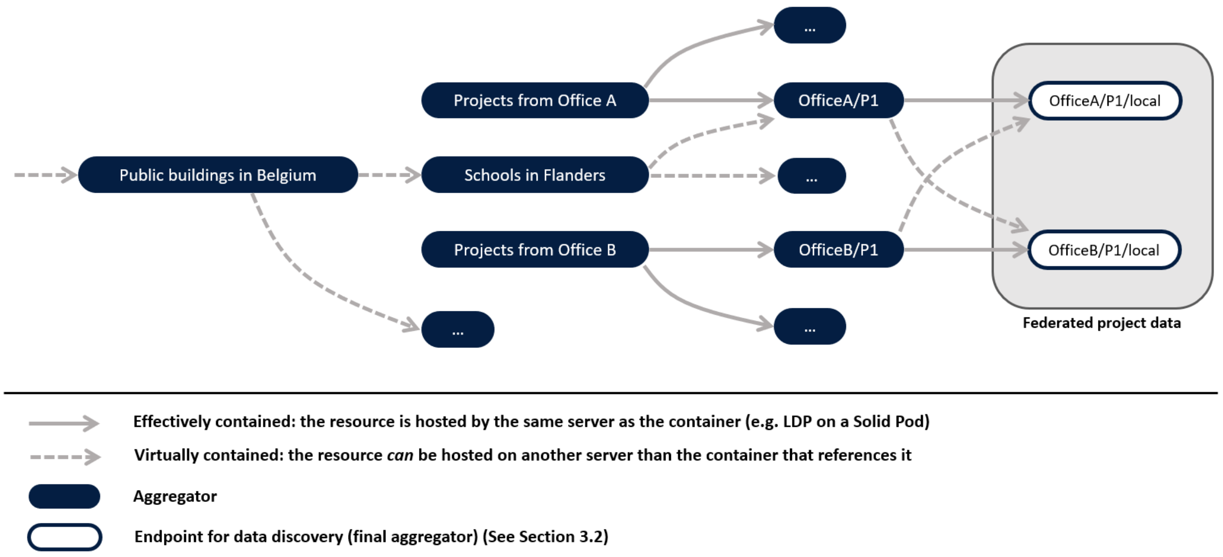

...
Projects from Office A
OfficeA/P1
OfficeA/P1/local
Public buildings in Belgium
Schools in Flanders
...
Projects from Office B
OfficeB/P1
OfficeB/P1/local
...
...
Federated project data
Effectively contained: the resource is hosted by the same server as the container (e.g. LDP on a Solid Pod)
Virtually contained: the resource can be hosted on another server than the container that references it
Aggregator
Endpoint for data discovery (final aggregator) (See Section 3.2)

Fig. 5. Discovering relevant datasets via (nested) aggregators

We implement this project access point as an LDP container which *effectively* contains this data (i.e., the 'local' folder, being the final endpoint for data discovery, is stored on the same Pod as the aggregator). However, it also references the other office's information relevant for this project - by *virtually* containing the contributions of the other office. Note that we can flexibly support situations where a consortium can decide to have only one access point, e.g., hosted by the project manager, and fine-grained situations where individual stakeholders can choose to aggregate information from their own subcontractors or individual employees as well. In the first situation, individual stakeholder aggregators then just point to this consortium-wide access point, their aggregator functions effectively as a shortcut. In the second situation, there is one fewer degree of nesting: the project access point then (1) contains contributions from the stakeholder to the project and (2) points to contributions from other partners.

Stakeholder-maintained aggregators are just one example. External aggregators may also aggregate these project access points (or one of them); in Figure 5 this is illustrated by the 'Schools in Flanders' aggregator. Such public aggregators can be just RDF graphs on the Web. Furthermore, aggregators can be public, but that does not mean their sub-aggregators are public as well - the decision making on access control still happens with the maintainer of the endpoint for data discovery.

3.2. *Organising Project Access Points using the Solid Ecosystem*

The above-described system of aggregators can perfectly function without Solid: they are regular LDP containers which describe a cascading system of where to find datasets that are relevant according to a specific set of parameters - in our case these parameters will boil down to 'datasets that are part of the same project'. As the Solid ecosystem hosts data based on LDP, we may use Solid's system of decentral access control seamlessly on top of our aggregator infrastructure. The 'virtual container' layer used by the aggregators does not exist in the Solid specifications, but we did not identify major conflicts either.

In [18], it is mentioned that an AEC office may set up their own Solid infrastructure, to host their own Pods and WebIDs. Specifically, this means their self-hosted Solid Identity Provider (IDP) allows the office to be identified as an actor on the Web, and provides the means to validate the WebIDs it provides to any other Solid-enabled Server via the WebID-OIDC protocol. In this paper, we assume the IDP and the Solid server to be combined in one service, although that is not strictly required by the specifications. Other offices host their own IDPs and Pods as well. In the following section we will dwell upon how the infrastructure of the Solid Pods can be used to support federated AEC projects.

In Section 3.1, we indicated that project data is easy to discover by dereferencing known LDP containers (aggregators). In some cases, these aggregators may be public registries. In most cases, however, they will be pod-specific, listing the projects relevant to the authenticated stakeholder. As this stakeholder's WebID is known through the authentication process, clients may start the discovery of this personal aggregator by dereferencing this WebID (Fig. 6).

For each individual stakeholder, it is thus required to define a property (`lbds:hasProjectRegistry`) which points to the aggregator (`ldp:Container`) containing (`ldp:contains`) the individual project repositories. This is shown in Listing 4 and simply links a personal identity card with a project registry. Only by doing so can it be tracked which projects are owned and managed by an actor/office (Listing 5).

Listing 4: An LBD project repository can be found via the WebID of a stakeholder

```
<#me> lbds:hasProjectRegistry <https://pod.my-office.org/lbd/> .
```

Listing 5: An LBD project repository contains access points to the projects the stakeholder participates in.

```
<https://pod.my-office.org/lbd/> ldp:contains
        <9320812f/>, # short ID for readability
        <80c6aa41/> .
```

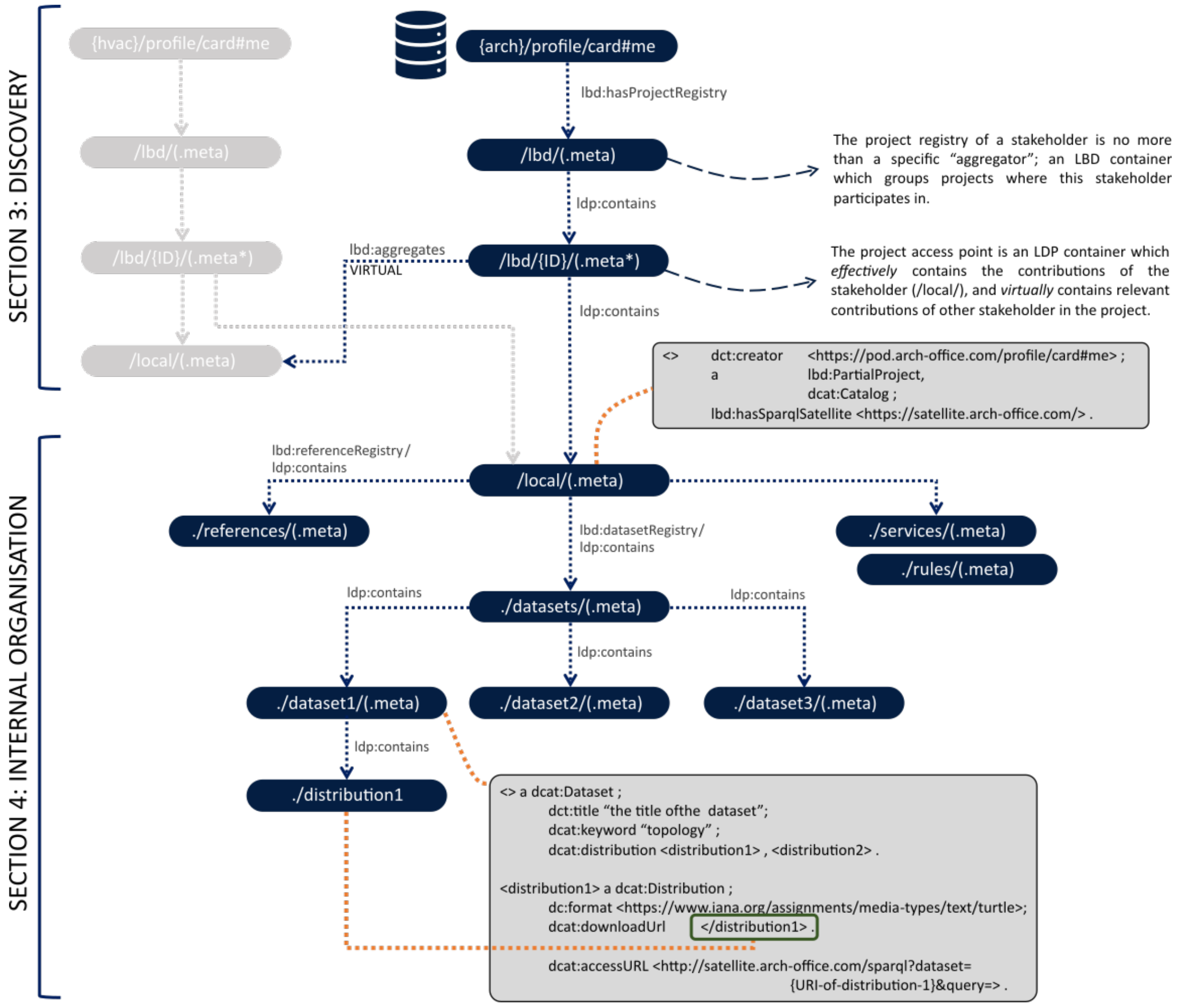
Using the Solid infrastructure, an office can create a single LDP container for every project they participate in. In this LDP container, they may define local access points - again aggregators, but now both *effectively* containing a subfolder with the contributions of the office (a 'partial project', here identified with <*local/*>), and *virtually* pointing to the partial projects of other offices in the network (Listing 6). For discovery of project data, the distinction between real and virtual containers is trivial, as a client will just follow the indicated URLs and retrieve the next container in the property chain in order to reconstruct the project graph. For resource hosting, however, it is not: a stakeholder is only sure of the access rights they have to their own Pod, and in this case to their local partial project. The effectively hosted <*local/*> container, which hosts the resources provided by the office, thus ends the discovery chain.

Listing 6: An LBD project repository contains access points to the projects the stakeholder participates in.

```
<https://pod.my-office.org/lbd/9320812f/>
    ldp:contains <local/>, #URL of the partial project where the stakeholder stores project contributions
    ldp:contains <https://pod.other-office.org/lbd/9320812f/local> .
            #virtual link to contributions of other stakeholder
```

3.3. Inviting Stakeholders to a Project

The infrastructure of aggregators and partial projects will change dynamically during the lifecycle of the project and may even be even task dependent. To have a stable mechanism to communicate the project's existence to new contributors is therefore no luxury. This can be done by first granting the newcomer access to an existing aggregator of the project and then notify them. This notification must at least include a pointer to this known



```
<>      dct:creator     <https://pod.arch-office.com/profile/card#me> ;
        a               lbd:PartialProject,
                        dcat:Catalog ;
        lbd:hasSparqlSatellite <https://satellite.arch-office.com/> .
```

```
<> a dcat:Dataset ;
        dct:title "the title ofthe  dataset";
        dcat:keyword "topology" ;
        dcat:distribution <distribution1> , <distribution2> .

<distribution1> a dcat:Distribution ;
        dc:format <https://www.iana.org/assignments/media-types/text/turtle>;
        dcat:downloadUrl    </distribution1> .

        dcat:accessURL <http://satellite.arch-office.com/sparql?dataset=
                                    {URI-of-distribution-1}&query=> .
```

Fig. 6. Data discovery patterns for the LBDserver. A stakeholder's project registry is an aggregator which groups all the projects the stakeholder contributes to. These project access points are themselves also aggregators, pointing to both their own contributions in the <local/> subcontainer as well as contributions of other people deemed relevant by the maintainer of the aggregator (in this case, the owner of the POD).

aggregator, so they can create their own partial project based on this template. A detailed description of this process is outside the scope of this paper, but it can be noted that this process can be automated by using RDF-based (hence machine-readable), form-validated LDP notifications[22] sent to stakeholders using their Solid inbox ldp:inbox, which is typically mentioned in the WebID profile card.

With the project access point and its local and remote contained partial projects, the minimally required information is available to establish a project graph. Now that the discovery patterns are clear, the next section will focus on the contents of the partial project repositories, i.e., the internal organisation of the <*local/*> container.

[22]https://www.w3.org/TR/ldn/

4. Internal Organisation of a Partial Project

As discussed in Section 3, the <local/> container dereferences to an RDF graph which describes itself as a partial project (`lbds:PartialProject`). This graph points to the elementary subcomponents of an LBDserver partial project, namely datasets, references and services (Listing 7). Datasets include the metadata and content (*distributions*) of separately created resources for the project. The service registry references (semi-)autonomous services managed on the office's LBDserver instance, whilst the reference registry describes the sub-document links between (federated and local) project resources. In the following sections, we will provide a detailed description of the structure and purpose of each of these subcomponents.

Listing 7: Project access point offers pointers to local information and the means to discover federated project data

```
<https://pod.my-office.org/lbd/9320812f/local/> #partial project contained by the project repository in Listing 6
    a lbds:PartialProject;
    lbds:partOfProject "9320812f";
    rds:label "iGent" ;
    dct:creator <https://pod.my-office.org/profile/card#me> ;

    # elementary pointers to find local project information
    ldp:contains <datasets/>, <services/>, <references/> ;
    lbds:hasDatasetRegistry <datasets/>;
    lbds:hasServiceRegistry <services/>;
    lbds:hasReferenceRegistry <references/>.

    # indicating an authenticated SPARQL endpoint for querying the project
    lbds:hasSparqlSatellite <https://query.my-office.org/9320812f/query?query=>
```

4.1. Datasets and Distributions

This section continues to describe how project datasets can be organised in a way that allows easy retrieval and querying of the content of these datasets within the project, by means of the `lbds:hasDatasetRegistry`. In the remainder of the paper, we use the DCAT terminology, where a 'dataset' identifies the metadata of a resource and a 'distribution' contains its actual content, using a specific data format. The `lbds:hasDatasetRegistry` property allows to point from a partial project folder to a container that includes the datasets present in the partial project (Listing 7). In the example given in Listing 8, this includes dataset 1, 2, and 3. Organising project resources as a flat list of metadata containers ensures their URLs remain stable, while the metadata descriptions allows an external Web API or a browser client to virtually restructure them in a way that matches the use case (e.g., based on topic, on publication stage (ISO 19650) or based on project tasks they are relevant to).

For the description of datasets (i.e., metadata of actual distributions), we rely on the DCAT vocabulary (section 2.4), as this vocabulary has higher semantic flexibility to describe metadata. Since Solid-based "folders" are per definition LDP containers, and there are overlapping uses of LDP (as used in Solid) and DCAT (as an extension to add metadata), we use these in parallel. This means that an LBDserver dataset will also be an LDP container and that distributions of this dataset (`dcat:distribution`) will also be LDP Resources (`ldp:contains`), effectively hosted by this container in the Solid environment (Figure 6).

Listing 8: The dataset Registry contains the pointers to relevant individual datasets. The datasets are ideally stored separate, as shown in Fig. 6; yet, dataset1 is shown here in one snippet to show how one points to the other.

```
# contents of the /local/datasets/ container
<> ldp:contains
    <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;
    dct:format <https://www.iana.org/assignments/media-types/text/turtle> ;
    dcat:downloadUrl <dataset1/distribution1> .
```

Although the dataset registry describes metadata of the different resources that are available in a project, it does not describe the actual dataset content, i.e., the distributions. Distributions can be stored in multiple ways. The most straightforward way is to host a distribution in a file-based Solid Pod. In this case, the resource is hosted in the container with the dataset metadata, its URL corresponding with the folder structure of the Pod. However, sometimes dedicated data stores have better performance or offer more extensive functionality. For example, to the knowledge of the authors, no Solid Server implementation currently supports (access-controlled) exposure of a SPARQL endpoint - an important requirement to the LBDserver: given the amount of triples of RDF-based BIM models, querying multiple graphs client side would become quite expensive. On the other hand, implementing an authentication middleware around an existing SPARQL endpoint is feasible, and essential for data security. To facilitate this 'pod-external' hosting, we propose the use of 'satellite' services, which are discussed in the next Section.

4.2. Satellites and the Service Registry

We propose the term 'satellite' services to identify highly trusted pod-specific services with divergent functionality. Essentially, they have in common that they provide a service layer on top of a very specific (group of) Pod(s). In order to be compatible with the main Pod, they need to access the ACL documents protecting certain datasets - these are quite extensive access rights - a reason to self-host them alongside the Office Pod and not outsource them to third parties. Satellites act as gatekeepers to a more 'intelligent' or more dynamic data access than allowed by a regular Solid Pod - they could be considered Pod-specific digital assistants. A satellite will either have its own WebID to authenticate to the POD, or use the WebID of its owner. The former is more cumbersome to set up, but allows a fine-grained approach as to what a satellite is allowed to do with the data on the Pod. We can define an 'LBDserver instance' as the combination of an office-based Solid Pod and its attached satellites. However, a Solid Pod without attached services will still suffice for supporting the LBDserver framework, as it contains the basic features of decentral storage and access control, and a primary (downloadable) distribution for every dataset.

The concept of satellites is very broad. Example satellite services are the following:

1. Specialised databases that grant authenticated access to distributions of project datasets. This allows to define, for example, a pod-external time-series database, a document store for geometry and a quad store for RDF-based data.
2. Web APIs that allow the client to create complex data and file structures on their POD. In the case of the LBDserver, we consider such proxy API a viable option to create and maintain the infrastructure for projects, datasets, distributions, and references (see Section 4.3), compared with a fully browser-based approach that can directly interact with a Pod.
3. Satellites that extend the basic access control specifications as implemented by Solid. For example, someone may not have full Read access to a dataset, but may be given the opportunity to ask yes-or-no questions (SPARQL ASK), or have a curated list of queries they may run on the Pod. Moreover, a visitor can be granted access to a resourceif they can proof they comply to a specific pattern [30].
4. Notification satellites can be configured to watch a project repository and notify certain agents when a complex event occurs. While notifications for non-complex events can be handled by technologies like pub/sub Web Socket protocols, notifications for complex events may be the result of RDF reasoning on combined datasets.
5. Synchronisation satellites: several existing distributions of the same satellite may require regular synchronisation, especially in phases where they are regularly subject to changes. Alternatively, when distributions are considered only derived views of the same 'core data', these views can be dynamically generated by specialised satellites. For example, by providing on-the-go conversions (e.g., there is a glTF geometric distribution available, but an STL distribution is needed) or generating specific versions of a model based on a core geometry resource described using the OMG[23]/FOG[24] ontologies.

[23]https://www.projekt-scope.de/ontologies/omg/
[24]https://mathib.github.io/fog-ontology/

4.2.1. Storage satellites

Present Solid implementations like the Community Solid Server already allow to internally route data storage based on their type (e.g., RDF is served in a triple store, non-RDF using a classic file-system). However, the endpoints of these systems are effectively hidden to the client, which makes it difficult to point to these services and APIs *at a data level* to enable discovery. Furthermore, this would introduce a dependency on the Solid Server implementation used, which we wish to avoid here. Therefore, it was chosen to decouple dataset distributions (Section 4.1) from the default storage mechanisms of the Solid Server where the datasets reside, and instead explicitly state where these distributions can be retrieved. This way, the option to serve them in specialised databases, which maintain an intimate relationship with the Pod, remains possible, in addition to 'default', file-based distributions stored on the Pod.

In the LBDserver ecosystem, we define a property `lbds:hasSatellite` that indicates the endpoint where a specific project can be queried (See Listing 7). Given the tree-like organisation of a Solid Pod, this can be generalised when linking a satellite to an `ldp:Container`, indicating the container and its sub-containers are queryable on this satellite.In this way, a SPARQL satellite can offer query functionality on a single knowledge graph that is the union of the resources in the container tree. In the context of a project, this allows quick retrieval of resources based on specific parameters. For example, to discover all resources related to the architectural design (querying the metadata). As such union graph can still maintain the original tree structure, for example as named graphs, access control on query results is still possible.

Listing 9: The SPARQL satellite must be able to check access rights to read query results before sending these results back to the client. It could transform the query behind the scenes or only support a subset of SPARQL that includes the source (e.g. FROM, FROM NAMED, GRAPH)

```
SELECT ?dist ?g
WHERE {
    GRAPH ?g {
        ?ds a dcat:Dataset;
            a cdc:ArchitecturalDesignDS ;
            dcat:distribution/dcat:downloadURL ?dist .
    }
}
```

Of course, the flexibility of serving distributions separate from datasets introduces a few additional complexities, mainly related to access control. For example, if the distribution itself is not hosted *on the Pod* but on a satellite, where does the access control document reside and how is it linked with the distribution? Although this topic goes beyond the scope of the paper, a few remarks can be made.

In the proposed ecosystem, access to resource metadata (i.e., the `dcat:Dataset`) is still compliant to the Solid specs: as metadata is organised as LDP containers hosted in a file-oriented Pod, their access control will be regulated by either dedicated or fallback ACL resources. The same holds for Pod-internal distributions. However, when (non-public) distributions are served outside the Solid Server, several questions on access control remain. Generally speaking, Pod-external distributions will need to be governed by their own access control mechanisms. However, when we only consider distributions served by Pod-specific satellites, access control rules *may be 'centrally' stored on the Pod*, given the strong dependency of the satellite on the Pod. Also, in most situations, distributions of the same dataset will have the same access rules. Therefore, in the current ecosystem we propose for access control to regulated for the entire dataset (container). A satellite can check on the (default) Pod-hosted distribution if (read) access can be granted, possibly using more complex access control mechanisms [30]. The satellite should be able to extract a visitor's WebID and have the rights to check a dataset's access rights.

However, it is perfectly possible for a stakeholder to incorporate *public* resources that are maintained outside the project and outside the ecosystem structure. This to allow aggregation of, for example, governmental and geospatial datasets in the set of project resources. To be able to include this external data into the ecosystem, stakeholders can create a new dataset document (metadata) in their Pod, mentioning the remote `dcat:downloadURL` or `dcat:accessURL`. They need to make sure that these URLs remain accessible.

4.2.2. *Functional Satellites*

Functional satellites can be set up to facilitate interaction with the LBDserver. While many of these interactions (e.g., the creation of a project or the registration of datasets and distributions) can be carried out also directly in the browser, a dedicated satellite is likely to improve performance. However, more 'continuous' tasks like data synchronisation and validation should per default be running as back-end services. To handle this process in a separate Web API decouples this from the Solid Server implementation and offers a user- and developer-friendly alternative to communicate with the LBDserver, based on well-known HTTP requests for CRUD (Create-Read-Update-Delete) operations.

Functional satellites can as well offer an intermediary layer that allows to match the LBDserver storage patterns to alternative 'views'. After all, in the LBDserver, all datasets are stored in a flattened list, to allow keeping resource URLs stable and generate 'virtual' views on this data based on metadata queries. Examples in AEC context would be virtual views based on the BCF API[25] for BIM-related issue management or the folder structure specified in the ICDD standard.

4.2.3. *The Service Registry*

The DCAT specification allows semantic definition of 'services' (`dcat:DataService`) (Figure 4). As such, services may be categorized according to their case (`dct:type`), conformance with specific standards (e.g., SPARQL) can be indicated (`dct:conformsTo`), and endpoints can be described and referenced (`dcat:endpointDescription` and `dcat:endpointURL`). As a way to link the Pod with its satellites, the LBDserver infrastructure includes a Service Registry (See Listing 7). Listing 10 gives an example for the description of a satellite that offers RDF storage and SPARQL querying of distributions. The property `dcat:servesDataset` indicates which datasets have distributions that can be retrieved using this service.

Listing 10: Description of a SPARQL satellite service in the Service Registry

```
# The service registry itself is an LDP container as well.
# It may directly mention its content but can also host them as individual LDP resources.
<> ldp:contains
        <#service1> ,
        <#service2> ,
        <#service3> .

<#service1> a dcat:DataService ;
        dct:conformsTo  <https://www.w3.org/TR/sparql11-query/> ;
        dcat:endpointURL <http://fuseki.arch-office.com/query> ;
        dcat:servesDataset dataset:dataset1 , dataset:dataset5 .
```

The link between services and datasets can be described on the side of the dataset as well: distributions served by a service indicate this relationship with the property `dcat:accessService`, alongside their `dcat:accessURL`. We can thus add a few triples to the dataset described in Listing 8, creating a distribution of this dataset served via the service described in Listing 11.

Listing 11: Description of a SPARQL satellite service in the Service Registry

```
<dataset1#distribution2> a dcat:Distribution;
    dcat:accessService <https://pod.my-office.org/lbd/9320812f-b587-4f0b-ba0c-36988e6b8a31/services#service1 ;
    dcat:accessURL <https://fuseki.arch-office.com/query> .
```

4.3. *The Reference Registry*

4.3.1. *Abstract concepts and Sub-document Linking*

The documented registries together with the project organisation for project discovery offer a way to allocate datasets and their content in a distributed catalog. Other mechanisms are needed, however, to align and reference information from different datasets on a sub-document level. To be able to semantically enrich heterogeneous re-

[25]https://github.com/BuildingSMART/BCF-API

sources, using modular vocabularies, is an explicit ambition of the LBDserver. This is particularly difficult in a decentralized approach, where access control is distributed and self-organised, and where various kinds of data can be included (triples, images, point clouds, etc.).

The Reference Registry is built to handle such links across resources within the proposed ecosystem. The design of this Reference Registry is largely based on the patterns provided by the ICDD standard, although we will re-interpret some of the ICDD definitions to make more sense in the context of distributed digital twins and deviate from the standard where we deem it necessary. Still, conversions of LBDserver sub-document references to standard ICDD structures should be possible in a straightforward fashion.

We can make a few extensions to the ICDD data patterns to increase semantic expressivity. For example, if someone identifies a pixel zone on an image, which represents a damaged area, it should be possible to use this pixel zone as a means to further enrich the damaged area ("caused by erosion"), but also to comment on the pixel zone itself ("The damage is larger than the zone you identified - please extend"). The former can be easily done using ICDD; the latter is more difficult. As we will see in Section 4.3.3, this is achievable when references can be referenced themselves. This means that references (in ICDD: link elements) can also be concepts (in ICDD: links) at the same time - something that is not allowed in ICDD. Lastly, in Section 4.1 we argued for the separation of datasets (metadata) and distributions (content). ICDD does not make this difference, as it just defines `ls:hasDocument` to indicate the origin of a sub-document identifier.

Considering the above, in the following paragraphs we will propose a data structure based on but not identical to ICDD.

4.3.2. Structure of the Reference Registry

Like the other registries (Sections 4.1 and 4.2), the Reference Registry is simply available from any Partial Project using the `lbds:hasReferenceRegistry` predicate. It is also structured as a `dcat:Dataset`, which can have multiple distributions and accessURLs. To enable representing links between concepts, those concepts first need to be identified. We identify the abstract concepts as instances of `lbds:Concept` (similar to `ls:Link` in ICDD, but with a focus on the instance being an abstract concept rather than merely a link). Semantic enrichment of those concepts happens via references (`lbds:hasReference`, similar to `ls:hasLinkElement` in ICDD).

Previously, we discussed the benefits of using `dcat:Datasets`, described in metadata resources, the main access points to `dcat:Distributions` of a particular dataset. In ICDD, a document can be referred to directly in a Link Element (`ls:hasDocument`), because ICDD is meant to function as a file-based container with a snapshot of the project - not intended to serve datasets which may still undergo active changes. In other words, ICDD features only one distribution per dataset - a federated CDE may have multiple. Therefore, a `lbds:Concept` refers to the `dcat:Dataset` rather than to individual distributions, using the property `lbds:inDataset`. Because the form of identifiers depends on a particular distribution of this dataset, identification of a distribution (`lbds:inDistribution`) happens at the level of the identifier instead of at the level of the reference (or the `LinkElement`). A reference indicates its identifier via `lbds:hasIdentifier`, which indicates its value with `schema:value`. Figure 7 illustrates these patterns.

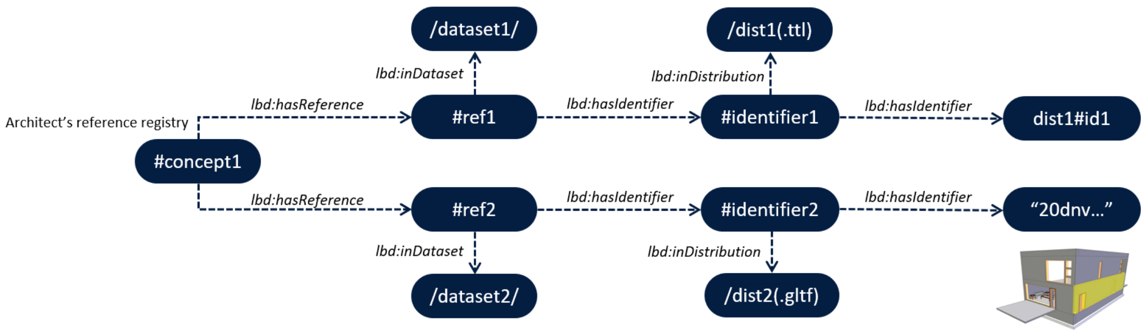


Fig. 7. RDF patterns from abstract concept to sub-document identifier. A concept can be referenced by multiple resources (datasets and distributions) on the web.

4.3.3. Concept Enrichment in a Federated Environment

Enrichment of abstract `lbds:Concepts` should never take place directly - a reference should always be used as a proxy 'to say something about the concept'. Apart from the data structure displayed in Figure 7, which is essentially meant to link references to concepts, there are two exemptions on this embargo:

1. Alignment with federated aliases of the same concept.
2. Commenting (referencing) on other references and datasets *as datasets*.

Firstly, since every partner in the federated project will maintain a reference registry, free to identify and semantically enrich any `lbds:Concept`, a mechanism should be present to align those references over the entire project. The first stakeholder to identify a concept in the project is not necessarily the most suitable one to choose a unique ID for this concept (which will most likely be related to the URL of their Pod). To avoid such racing conditions, and to allow gradual alignment of resources (e.g., partial models) while the project proceeds, every stakeholder may define concepts and associated references within the scope of their own Pod (using their own Reference Registry). Whenever a remote version of the same concept is identified, an `owl:sameAs` relationship must be established, allowing discovery and registration of the remote alias.

Figure 8 illustrates this. In the situation depicted in this figure, the architect of the project indicates that a geometric element ("20dnvw6f90Nfh9rP1dlXrr") in their architectural partial model (with specific dataset and distribution) corresponds with `arch:concept1`, and in another (RDF-based) dataset one says that `arch:concept1` is an instance of `beo:Wall`, using a URI-based identifier as a proxy (in Figure 8, <#id1>). The geometry and the semantics are now established as representations of the same abstract concept, within the Pod of the architect office. At the same time, the engineering office also has some statements to make about this wall instance - in their own Pod. Therefore, a local alias of this concept should be created in their own reference registry, and by doing so, the engineer's local partial project can keep functioning independently from other actors and their enrichments in the project. New sub-document identifiers can now be linked to this concept (`eng:concept8`), e.g., to indicate that the element (in Figure 8, <#id4>) has a specific U-value.

The final alignment of these abstract object aliases can happen in multiple ways, using web interfaces or (semi-) automatic mapping services. For example, if the scale of the project does not allow the creation of a separate energy-oriented 3D model, the engineers may directly visualise the geometry of the architectural model in a Web GUI and select the geometry of interest, as a proxy to create a local alias of the concept and directly map it to the alias defined by the architecture office. This approach also supports the situation where stakeholders work independently from one another, keeping their project data in-office during the 'work-in-progress' stage of data publication (ISO 19650). When each of them has a local project with multiple resources that are already (internally) mapped to one local abstract concept (e.g., 3D model, structural semantics, energy semantics, schedules, etc.), mapping to other stakeholders' datasets and their sub-document identifiers (e.g., during the 'shared' phase) can be limited to visualising one reference from both stakeholders in a Web GUI and indicate that they reference the same 'thing' in reality.

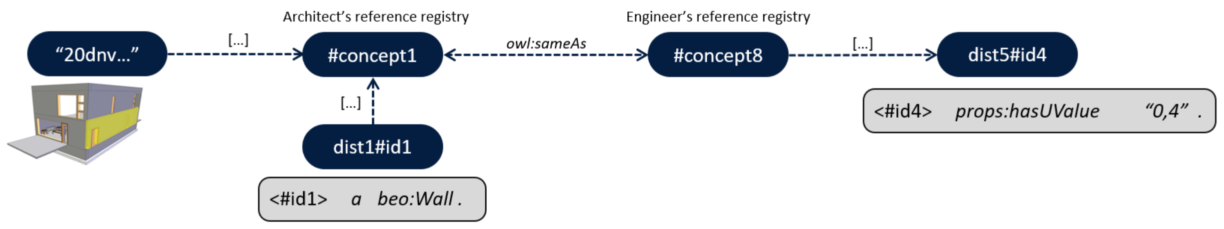


Fig. 8. An owl:sameAs relationship allows a client to query multiple federated Reference Registries and infer when people are actually enriching the same concepts.

The creation and registration of aliases can happen in multiple ways. A functional satellite can be set up on each LBDserver instance in the project consortium, which can be notified (and notify others) about the creation of an alias for an existing concept. This way, a backlink can be made (`owl:sameAs` is bi-directional), either automatically or after check and approval of the satellite owner. In an environment without satellites, stakeholders can allow each other `acl:Append` rights to their reference registry, or to a dedicated RDF resource containing aliases. Usage of `owl:sameAs` is known to be delicate [36]. However, as the goal is indeed to confirm that these concepts are effectively the same thing, and, if correctly used, direct semantic enrichment of abstract concepts will not happen, we consider the use of `owl:sameAs` appropriate.

4.3.4. Referencing existing References

Apart from semantic enrichment of 'real' abstract concepts, which was the topic of Section 4.3.3, it should be possible to reference datasets, documents and sub-document references as well, *in their capacity of being digital documents*. This is, for instance, of use to create issues concerning modelling practice, to allow comments on due project deliverables, etc. Within the proposed data structure for sub-document linking, it is thus allowed to create a `lbds:hasReference` chain, or to create concepts and accompanying reference patterns that indicate equality (`owl:sameAs`) with existing dataset URLs, distribution URLs or reference URIs (Figure 9).

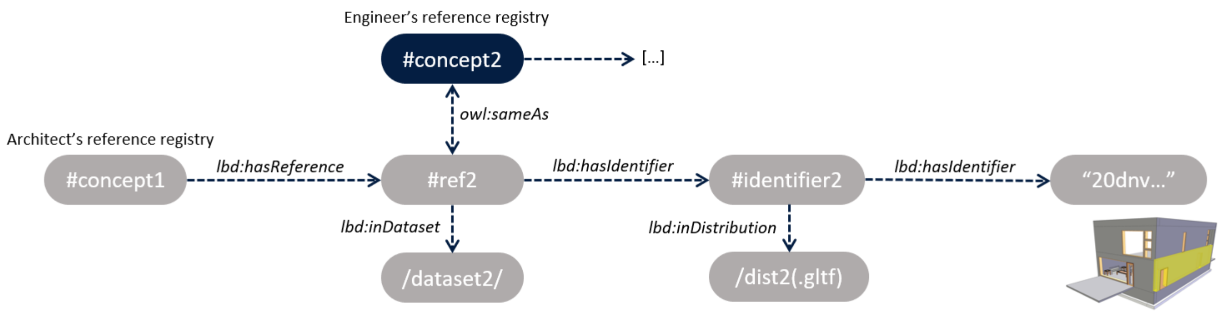


Fig. 9. A reference of a concept can be enriched as well, either by having its own references (`lbds:hasReference`) or by creating an alias using `owl:sameAs`.

4.4. Alternative Registry Representations

Thus far, we have structured the Reference Registry using RDF ontologies, as this allows us to easily make the bridge with concepts of ICDD and use existing concepts such as `owl:sameAs`. However, since the actual semantic freedom required by the reference registry is very limited, other distributions (e.g., JSON/BSON) are feasable and might have advantages over an RDF-based approach. For example, to serve references in a document store instead of a SPARQL store is likely to increase performance, especially considering the total amount of concepts present in a project.

5. Proof of Concept

While the above sections explain the proposed ecosystem in considerable theory, this section explains the Proof of Concept that was created to evaluate the validity, feasibility, and overall (qualitative) performance of the proposed system. Firstly, we briefly present the components of the prototypical codebase of the LBDserver. Secondly, we present the actual building case and stakeholder constellation, then we explain which datasets and distributions have been created, and how access was created and generated to all the diverse parts of the ecosystem. A demonstration is given for the semantic enrichment of the datasets to finalise this proof of concept.

5.1. The LBDserver prototype

In the previous sections, data patterns as well as services related to the LBDserver have been laid out. As indicated, the minimal requirements for this environment to function are (1) that every participant in the project is identifiable with a WebID, and that (2) every office maintains a Solid Pod, either locally hosted or outsourced to a third party. A Pod, however, is just the storage provider. Interactions that are more high-level than classic CRUD operations need to be initiated by pod external actors, such as satellites (Section 4.2) or browser clients. A prototypical codebase was developed within the scope of this project to make abstraction of the low-level interactions, and provide developers with classes and methods to easily create applications on top of the LBDserver infrastructure: Projects, Datasets, Distributions and References. This codebase has been published as an npm module (lbdserver-client-api)[26]. Furthermore, a SPARQL satellite was configured, regulating access to an Apache Jena Fuseki[27] SPARQL endpoint via an Express.js REST API [28]. The satellite is able to extract and verify the WebID of requesters, and can itself make WebID-authenticated requests to synchronise with its Pod. A similar Express.js satellite has been created to serve non-RDF (binary) distributions in a blob store, connected to a MongoDB database[29]. Even without the setup of such satellites, the framework will still function: every dataset has a default (downloadable) distribution on the Pod. Figure 10 illustrates the relationships between the above-mentioned components and actors.

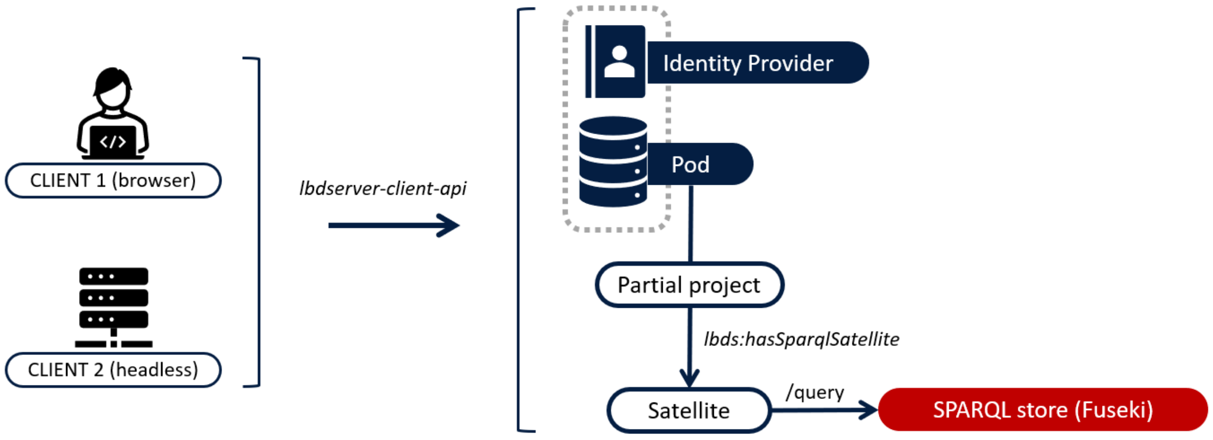


Fig. 10. Schematic of the LBDserver prototype implementation

5.2. *Case description and Setup*

To illustrate the data patterns discussed in Section 4, we consider the following example, featuring federated datasets of the iGent tower in Ghent, Belgium (EVR Architecten). We consider the operational phase of the building. As a preparation step, the original Autodesk Revit partial models for Architecture (modelled by Bureau Bouwtechniek (BE)), Engineering, Electricity, and HVAC (all three modelled by Arcadis (BE)) were converted to IFC and then to a semantic model for RDF (Turtle) and a geometric model (glTF).

5.2.1. *Organisation*

In this simulation, we assume every stakeholder maintains their own Pod and associated WebID, with an instantiated project registry and project access point to the iGent project. We consider three stakeholders: Bureau Bouwtechniek, Arcadis, and the Facility Management office of Ghent University, as it concerns a University Building. Each of them stores their own contributions to the project in their Pod, as a local *partial project*. Corresponding with their IPR, Bureau Bouwtechniek hosts the architectural model, Arcadis the three others. At this ‘initial’ point, the FM office does not host anything yet, but optionally maintains already a *project access point* with virtual pointers to the federated models. This may be part of an *aggregator* maintained by the FM office, which, for example, contains

[26]https://github.com/ConSolidProject/lbdserver-client-api
[27]https://jena.apache.org/documentation/fuseki2/
[28]https://github.com/ConSolidProject/sparql-satellite
[29]https://github.com/ConSolidProject/satellite-mongo

references to all university buildings. Additionally, a Web API satellite, SPARQL store, and Document store are set up at each LBDserver instance in the federated network, following the setup described in Section 5.1.

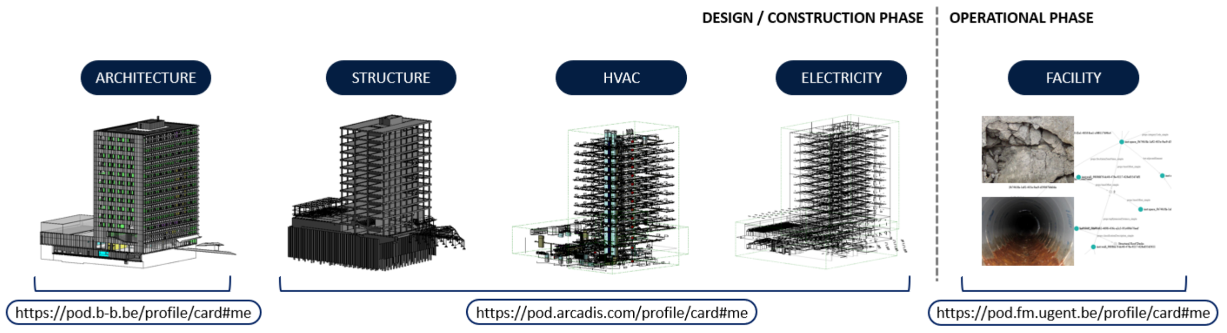

DESIGN / CONSTRUCTION PHASE
OPERATIONAL PHASE
ARCHITECTURE
STRUCTURE
HVAC
ELECTRICITY
FACILITY
https://pod.b-b.be/profile/card#me
https://pod.arcadis.com/profile/card#me
https://pod.fm.ugent.be/profile/card#me

Fig. 11. Use case setup and model IPR

5.2.2. *Datasets, distributions and satellites*

Each converted semantic RDF model and geometric glTF model is stored as a distribution with attached metadata (dataset) on the creator's Pod. They are as well mirrored to SPARQL and document satellites, discoverable via `dcat:accessURL`.

5.2.3. *Access Control*

Access control to project data is regulated by giving the office that created and owns a certain model full access control rights (`acl:Read`, `acl:Write`, `acl:Control`), and doing the same for their satellites. Other project partners get reading rights (`acl:Read`). Only reading rights are needed, as every contribution or comment they make will be stored on their own LBDserver instance, even when referring to data on other LBDservers in the project.

5.2.4. *References*

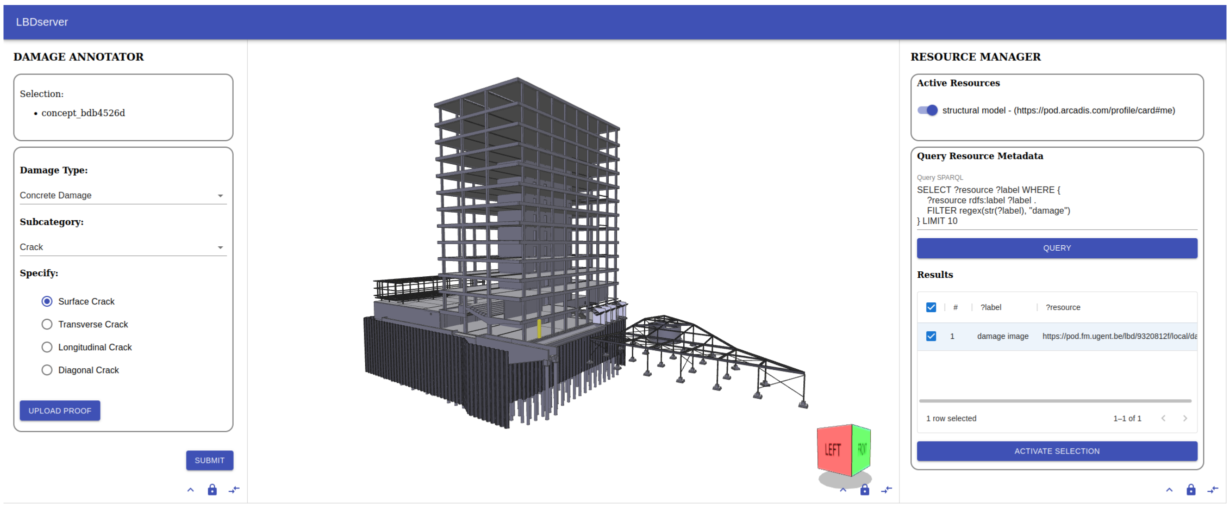
Upon data upload, the Reference Registry of each of the four stakeholders can be populated with abstract concepts that link corresponding entities in the semantic and geometric models, automatically generated based on the fact that in both conversions, the original GUIDs remain referenced, as described in [32]. However, in current industry practice, partial BIM models are created as separate models within the same coordinate system, but cross-document links are generally lacking. The elements in one partial BIM model are thus not yet connected with those in another one, but such semantic relationships can now be created by project stakeholders whenever necessary.

5.3. *Semantic Enrichment for a damage case*

The use case for this particular proof of concept includes the discovery and documentation of damage on an element in the federated model. We will discuss a situation where a Facility Manager localises damages on a column in the building and documents it in one's Pod, thereby referencing the external federated models of other stakeholders (in this case, Arcadis as creator of the structural model) and semantically enriching the overall project. Below, we describe the raw data structures created by these steps. However, an agent will never initiate these interactions directly - they can be largely simplified using a dedicated UI (e.g., Figure 12).

5.3.1. *Creation of Damage Graph by the FM*

The first step for the Facility Manager to undertake in the process of documenting the damages, is to create a dataset on one's Pod, referencing an RDF-based distribution. In our example, we will create the semantics in this new graph using the Damage Topology Ontology (DOT) and one of its extensions, the Concrete Damage Ontology (CDO) [10]. Note again that the LBDserver framework is independent of any choice of RDF vocabularies - it acts entirely on metadata level (Listing 12).


LBDserver

DAMAGE ANNOTATOR

Selection:
- concept_bdb4526d

Damage Type: Concrete Damage

Subcategory: Crack

Specify:

- [✓ Surface Crack]
- Transverse Crack
- Longitudinal Crack
- Diagonal Crack

UPLOAD PROOF

SUBMIT

RESOURCE MANAGER

Active Resources

structural model - (https://pod.arcadis.com/profile/card#me)

Query Resource Metadata

Query SPARQL

```
SELECT ?resource ?label WHERE {
   ?resource rdfs:label ?label .
   FILTER regex(str(?label), "damage")
} LIMIT 10
```

QUERY

Results

✓	#	?label	?resource
✓	1	damage image	https://pod.fm.ugent.be/lbd/9320812f/local/da

1 row selected 1–1 of 1

ACTIVATE SELECTION

Fig. 12. Example UI to navigate and enrich LBDserver resources. Included modules in the depicted setup are a 3D Viewer (based on the Xeokit SDK), a Query Module for finding and activating federated project datasets and a Damage Enrichment Module. Since data and applications are fundamentally separated, specialised UI modules can be created and combined depending on the use case.

Listing 12: Triples in the distribution of the Damage dataset, maintained by the FM. The metadata graph will be similar to the one listed in Listing 8.

```
@prefix distFM: <https://pod.fm.ugent.be/lbd/9320812f/local/datasets/damages/distribution1#>.

distFM:theDamagedElement dot:hasDamageArea distFM:theDamageArea ;
    a dot:ClassifiedDamage; cdo:SurfaceCrack ;
    cdo:crackWidth "35" .
```

To be able to reference them in other resources later on, both the damage and the damaged element need to get a local ‘abstract concept’ in the FMs Reference Registry. In this example, this is done with respectively `<#concept_bdb4526d>` and `<#concept_c2427dd9>` (Listing 13). These (relative) URIs represent the real objects and form the primary points for enrichment via a federated ´digital twin’. Using the patterns described in Section 4.3, these abstract concepts are mapped to the identifiers `distFM:theDamagedElement` and `distFM:theDamageArea`, in the dataset and distribution listed in Listing 12.

5.3.2. *Enrichment of project repository with damages (reference registry)*

After defining the damages, the Facility Manager can start mapping the digital references needed to enrich the original project with the newly asserted damage data.

Listing 13: Lifting the local context of the Semantics to a ‘global concept’ in the Artefact Registry of the stakeholder

```
@prefix refFM: <https://pod.fm.ugent.be/lbd/9320812f/local/references/>.
@prefix distFM: <https://pod.fm.ugent.be/lbd/9320812f/local/datasets/damages/distribution1#>.

refFM:concept_bdb4526d lbds:hasReference refFM:ref_a80854ae .
refFM:ref_a80854ae lbds:hasIdentifier refFM:id_c69531c5 ;
    lbds:inDataset <https://pod.fm.ugent.be/lbd/9320812f/datasets/damages/> .
refFM:id_c69531c5 lbds:inDistribution <https://pod.fm.ugent.be/lbd/9320812f/datasets/damages/distribution1> ;
    schema:value ''https://pod.fm.ugent.be/lbd/9320812f/local/datasets/damages/distribution1#theDamagedElement''^^xsd:anyURI .

refFM:concept_c2427dd9 lbds:hasReference refFM:ref_f1f2704e .
refFM:ref_f1f2704e lbds:hasIdentifier refFM:id_27801276 ;
    lbds:inDataset <https://pod.fm.ugent.be/lbd/9320812f/datasets/damages/> .
refFM:id_27801276 lbds:inDistribution <https://pod.fm.ugent.be/lbd/9320812f/datasets/damages/distribution1> ;
    schema:value ''https://pod.fm.ugent.be/lbd/9320812f/local/datasets/damages/distribution1#theDamageArea''^^xsd:anyURI .
```

5.3.3. *Enrichment of sub-document identifiers: pixel region*

The FM office now needs to document the damage with a photograph, and does so by creating a new metadata description for this photograph (`dcat:Dataset`), with the image (image/jpeg) as a distribution. As only part of the image shows the damage, the location of the damage on the image is indicated with a pixel zone, as a sub-document identifier. We can do so using the relationships listed in Listing 14. A similar enrichment can be made for a pixel zone that represents the damaged object. These relationships are expressed in the FM's local Reference Registry. Again, this can be easily done with a dedicated user interface (Figure 13).

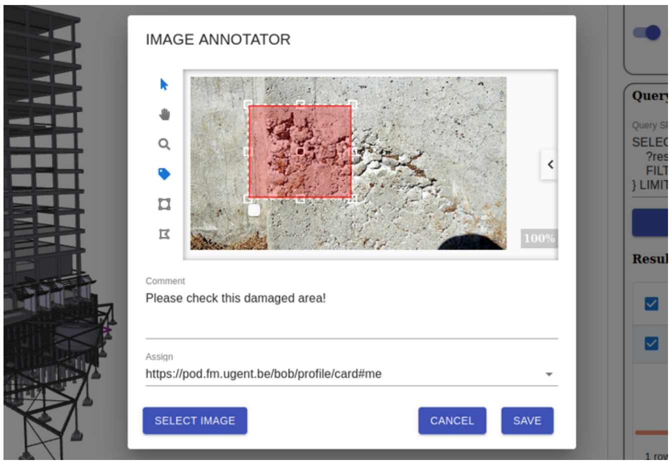
Listing 14: Defining an image's pixel zone as a specific representation of an abstract concept

```
@prefix refFM: <https://pod.fm.ugent.be/lbd/9320812f/local/references/>.

refFM:concept_bdb4526d lbds:hasReference refFM:ref_ba0fe231 .
        #this abstract concept is the same as the one mapped to the damaged element,
        #effectively linking the image zone with the element.

refFM:ref_ba0fe231 lbds:hasIdentifier refFM:id_1d2eee0e ;
    lbds:inDataset <https://pod.fm.ugent.be/lbd/9320812f/datasets/damagePicture1/> .
refFM:id_1d2eee0e lbds:inDistribution <https://pod.fm.ugent.be/lbd/9320812f/datasets/damagePicture1/jpegDistribution> .
schema:value ``210 480 190 40''^^xsd:string. #image region identified as a rectangle with x y w h.

[...]
```



Comment
Please check this damaged area!
Assign
https://pod.fm.ugent.be/bob/profile/card#me

Fig. 13. An image region annotator is part of the LBDserver Damage Enrichment Module (see Fig. 12)

5.3.4. Linking with remote models

At this moment, the `refFM:concept_bdb4526d` has two references in two different resources. However, these statements currently only exist within the local project of the FM. To effectively combine this information with relevant data from the other stakeholders, there are two options. Either the other stakeholders already have references to this object, or they have not. In the first case, we want to link the damage assertions with the actual object, a specific column as modeled by Arcadis. As we described earlier, every office created references based on the glTF model and LBD semantics, so the FM may just open the 3D model provided by Arcadis, 'access' the (existing) abstract element via the 3D model and indicate equivalence between Arcadis' abstract object and the one he just created to say it was damaged. A backlinking satellite may now inform the network of the new alignment, so their local Reference Registries can register the new alignment - `owl:sameAs` is transitive, so the newly mapped concept can directly be aligned to all the other aliases of this concept as well. In the case Arcadis has not created any references yet, the FM may create a new reference locally, referring to the remote dataset and distribution.

Listing 15: Alignment of (federated) concept aliases using `owl:sameAs`

```
@prefix refFM: <https://pod.fm.ugent.be/lbd/9320812f/local/references/>.
@prefix refArcadis: <https://pod.arcadis.com/lbd/9320812f/local/references/>.

refFM:concept_bdb4526d owl:sameAs
    refArcadis:concept_1756cb77 .
```

Listing 16: The alias of the concept in the Reference Registry of the Architect Office

```
@prefix refArcadis: <https://pod.arcadis.com/lbd/9320812f/local/references/>.

refArcadis:concept_1756cb77 owl:sameAs refFM:concept_bdb4526d ; # the backlink
    lbds:hasReference refArcadis:ref_5b54d29b , refArcadis:ref_a031ee99 .

refArcadis:ref_5b54d29b lbds:hasIdentifier refArcadis:id_f636296e ;
    lbds:inDataset <https://pod.arcadis.com/lbd/9320812f/local/datasets/arch_geom/> . #the geometric architectural model
refArcadis:id_f636296e lbds:inDistribution <https://pod.arcadis.com/lbd/9320812f/local/datasets/arch_geom/gltfDistribution> ;
    schema:value ``product-982f59b0-f2e1-485f-8ce1-c9f6117b9d44-body''^^xsd:String.

refArcadis:ref_a031ee99 lbds:hasIdentifier refArcadis:id_6cf00e25 ;
    lbds:inDataset <https://pod.arcadis.com/lbd/9320812f/local/datasets/arch_lbd/> . #the LBD graph of the architectural model
refArcadis:id_6cf00e25 schema:value ``https://example.org/model1/column_982f59b0-f2e1-485f-8ce1-c9f6117b9d44''^^xsd:anyURI .
        #local baseURI of the concept in the distribution, e.g. where it is classified and enriched.
        #not necessarily related to the URL of the Pod.
```

Once this mapping is done, a client can query the set of federated project resources [37] to search for product information of damaged elements. The semantic graphs are used for querying; the reference registry delivers the corresponding abstract concepts and their identifiers, where the non-semantic resources (e.g., 3D models, imagery) can be used to visualise the results. For example, to find the necessary information to order a replacement product.

Finally, a reference can be made not of the concept represented by the pixel zone (i.e., the damage) but of the pixel zone reference itself. This can be done by any actor in the network (Listing 17 and 18) and can take any form. It involves creating a reference of a reference, which can then be enriched further in just the same way as concept references.

Listing 17: Reference registration of a comment on another reference

```
@prefix refFM: <https://pod.fm.ugent.be/lbd/9320812f/local/references/>.
@prefix refArcadis: <https://pod.arcadis.com/lbd/9320812f/local/references/>.

refFM:ref_ba0fe231
    lbds:hasReference refArcadis:ref_f7196667 .
refArcadis:ref_f7196667 lbds:hasIdentifier refArcadis:id_d51b0800 ;
    lbds:inDataset <https://pod.arcadis.com/lbd/9320812f/local/datasets/comments/> .
refFM:id_d51b0800 schema:value "https://example.com/locallyDefinedComment1"^^xsd:anyURI .
[...]
```

Listing 18: The actual contents of the comment

```
<https://example.com/locallyDefinedComment1> rdfs:comment "The crack in the column runs farther than this polygon indicates" .
```

6. Discussion and Future Work

Considering the above Sections, we identify a need for the following components to be present in a federated ecosystem for the built environment:

1. ‘Aggregators’, i.e., RDF graphs (`ldp:Container`) that point to multiple access points, in order to group them according to specific parameters.
2. ‘Project access points’, i.e., RDF graphs (`ldp:Container`) that allow to retrieve the necessary information to discover the project, resolving to ‘partial projects’ provided by individual stakeholders.
3. ‘Identity Providers’, i.e., services capable of providing and verifying decentral identities via their WebID, as defined in the Solid specifications.
4. Solid servers, i.e., the basic hosting services and primary access points for project discovery.
5. ‘Satellite’ services, i.e., pod-specific services that may interact with the Pod in a fully trusted manner - for example, to serve and synchronise distributions in the most fit database (document store, quad store, etc.) or to interact with complex data structures on the Pod.

These components, and their interactions, form the conceptual backbone of the LBDserver infrastructure. However, there are several topics that need to be addressed in future research to make this ecosystem more mature and useful in practice.

Firstly, this paper did not discuss the interaction of third party services with the LBDserver infrastructure. In order to do their job, services (so, also any BIM tool) may need to either fetch information from a single federated project, or aggregate information from many projects simultaneously. Examples are simulation services (fire, building performance, cf. the BIMserver's BIM bots), but also planning services or services that provide access to contextual (geospatial, governmental) datasets. The interaction of these services with the network are an important future task. The prototypical NodeJS library (*lbdserver-client-api*) mentioned in Section 5 already makes abstraction of the necessary steps to communicate with the networ. A lot of research is currently going into the development of APIs to interact with Solid Pods and a Solid SDK. Combined with the discipline-specific requirements of the AEC industry, these APIs will provide the building blocks for a service-based infrastructure on top of federated building projects, including the interaction with native BIM authoring tools.

In Section 5, we converted an IFC model to Linked Building Data and geometry. This workflow is appropriate in an operational phase, when one or more as-built BIM models are available. These models can be considered semi-static; they will be enriched much more frequently than they will be modified. Synchronisation with commercial BIM tools is also from this perspective an important research topic: future research should determine how the infrastructure will deal with rapidly changing information during the design (or renovation) phases, and how it will keep track of provenance and data handover.

7. Conclusion

In this paper, we proposed an architecture for organising federated datasets for the AEC sector. Using the container structures provided by the LDP, we have shown that project data hosted by different stakeholders can be discovered using 'aggregators', i.e., containers referencing the relevant datasets according to the maintainer of a specific aggregator. This introduces a flexibility to fine-grainedly include relevant datasets at the level of individual project partners (e.g., subcontractor datasets, visitor datasets, etc.), while at the same time allowing a scalable approach towards the data integration of multiple projects at the same time: for example, public aggregators can be created and aggregate large sets of projects according to parameters like typology (public bridges, libraries, ...) or building phase (construction, demolition, ...).

In such a federated context, where every stakeholder can define the convenient extents of the project(s) from their point of view, those sub-document identifiers should be kept decentrally as well, so datasets and their linked identifiers can be aggregated at once. We argued that every stakeholder should be able to construct these links, independent from other participants in the project, in a local Reference Registry. However, to be able to (2) refer to data hosted by other actors in the network and (2) indicate concept equivalence in a gradual way, we propose the creation of local aliases of links created by others via `owl:sameAs`. This way, clients that have access to a project can easily combine those aliases into a set of links, all referring to the same abstract 'thing'. This 'thing' corresponds with the ICDD concept of Links, but may be interpreted not as merely a link between documents but as the main abstract concept that is *indirectly* represented via digital resources.

This combination of federated data organisation and heterogeneous enrichment of abstract things was illustrated with a use case building in Ghent. As a proof of concept, we listed some typical interactions between actors in this project.

We illustrated that, once an access point is known (e.g., discovered via the public WebID of the data owner), a client can find relevant datasets in a straightforward manner. While the Solid infrastructure forms the core of the project, we extended this infrastructure with the concept of satellites. While the use of such pod-specific services is currently not covered by the Solid specifications, it aligns with ongoing research efforts regarding decentral networks in other disciplines. A deviation from the current Solid specifications was introduced by implementing direct enrichment of *.meta* resources - an approach that allows a semantically richer description of dataset metadata (DCAT) compared with the default (restricted) use of .meta files to indicate LDP containment triples.

We believe this research will contribute to demystifying the concept of 'Web- and data-based BIM' (BIM maturity level 3), and offer a deeper understanding of the technology stack that allows the AEC community to work in a federated, interdisciplinary and interoperable context.

8. Acknowledgements

This research is funded by the Research Foundation Flanders (FWO), as a Strategic Basic Research grant (grant no. 1S99020N). We thank Bureau Bouwtechniek and Arcadis for granting permission to use their models in the use case of this paper.

References

[1] A. Borrmann, M. König, C. Koch and J. Beetz, *Building Information Modeling: Why? What? How?*, in: *Building Information Modeling: Technology Foundations and Industry Practice*, Springer International Publishing, 2018, pp. 1–24, Chapter 1. ISBN 978-3-319-92862-3. ISBN 978-3-319-92862-3. doi:10.1007/978-3-319-92862-3_1.

[2] Industry Foundation Classes (IFC) for data sharing in the construction and facility management industries, Standard, International Organization for Standardization, 2018.

[3] Specification for information management for the capital/delivery phase of construction projects using building information modelling, Standard, British Standards Institution, 2013.

[4] P. Pauwels, S. Zhang and 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] J. Beetz, J.P. Leeuwen, van and B. Vries, de, An Ontology Web Language Notation of the Industry Foundation Classes, in: *Proceedings of the 22nd CIB W78 Conference on Information Technology in Construction*, R.J. Scherer, P. Katranuschkov and S.-E. Sconfke, eds, Technische Universität Dresden, 2005, pp. 193–198.

[6] P. Pauwels and 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.

[7] M.H. Rasmussen, P. Pauwels, M. Lefrançois, G.F. Schneider, C.A. Hviid and J. Karlshøj, Recent changes in the Building Topology Ontology, in: *5th Linked Data in Architecture and Construction Workshop (LDAC)*, Dijon, France, 2017. doi:10.13140/RG.2.2.32365.28647. https://hal-emse.ccsd.cnrs.fr/emse-01638305.

[8] A. Wagner and U. Rüppel, BPO: The Building Product Ontology for Assembled Products, in: *7th Linked Data in Architecture and Construction Workshop (LDAC), CEUR Workshop Proceedings*, 2019, pp. 106–119. http://ceur-ws.org/Vol-2389/08paper.pdf.

[9] M. Bonduel, M. Bassier, M. Vergauwen, P. Pauwels and R. Klein, Scan-to-BIM Output Validation: Towards a Standardized Geometric Quality Assessment of Building Information Models Based on Point Clouds, in: *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, Vol. XLII-2/W8, Hamburg, Germany, 2017, pp. 45—52. doi:10.5194/isprs-archives-XLII-2-W8-45-2017.

[10] A.-H. Hamdan, M. Bonduel and R.J. Scherer, An ontological model for the representation of damage to constructions, in: *7th Linked Data in Architecture and Construction Workshop (LDAC), CEUR Workshop Proceedings*, 2019, pp. 64–77. http://ceur-ws.org/Vol-2389/05paper.pdf.

[11] K. McGlinn, A. Wagner, P. Pauwels, P. Bonsma, P. Kelly and D. O'Sullivan, Interlinking geospatial and building geometry with existing and developing standards on the web, *Automation in Construction* **103** (2019), 235–250. doi:10.1016/j.autcon.2018.12.026.

[12] K. Janowicz, A. Haller, S.J. Cox, D. Le Phuoc and M. Lefrançois, SOSA: A lightweight ontology for sensors, observations, samples, and actuators, *Journal of Web Semantics* **56** (2019), 1–10.

[13] L. Daniele, F. den Hartog and J. Roes, Created in close interaction with the industry: the smart appliances reference (SAREF) ontology, in: *International Workshop Formal Ontologies Meet Industries*, Springer, 2015, pp. 100–112.

[14] M. Poveda-Villalón and R. Garcıa-Castro, Extending the SAREF ontology for building devices and topology, in: *Proceedings of the 6th Linked Data in Architecture and Construction Workshop (LDAC 2018), Vol. CEUR-WS*, Vol. 2159, 2018, pp. 16–23.

[15] R. Buyle, L. De Vocht, M. Van Compernolle, D. De Paepe, R. Verborgh, Z. Vanlishout, B. De Vidts, P. Mechant and E. Mannens, OSLO: Open standards for linked organizations, in: *Proceedings of the international conference on electronic governance and open society: Challenges in Eurasia*, 2016, pp. 126–134.

[16] Information container for linked document delivery , Standard, International Organization for Standardization, 2020.

[17] M. Gürtler, K. Baumgärtel and R.J. Scherer, Towards a workflow-driven multi-model bim collaboration platform, in: *Working Conference on Virtual Enterprises*, Springer, 2015, pp. 235–242.

[18] J. Werbrouck, P. Pauwels, J. Beetz and 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. https://biblio.ugent.be/publication/8633673.

[19] T. Berners-Lee, J. Hendler, O. Lassila et al., The Semantic Web, *Scientific American* **284**(5) (2001), 28–37. doi:10.1038/scientificamerican0501-34.

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

[21] J. Beetz, J. Van Leeuwen and B. De Vries, IfcOWL: A case of transforming EXPRESS schemas into ontologies, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AI EDAM* **23**(1) (2009), 89.

[22] E.A. Petrova, AI for BIM-Based Sustainable Building Design: integrating knowledge discovery and semantic data modelling for evidence-based design decision support (2019).

[23] A. Haller, K. Janowicz, S.J. Cox, M. Lefrançois, K. Taylor, D. Le Phuoc, J. Lieberman, R. García-Castro, R. Atkinson and C. Stadler, The modular SSN ontology: A joint W3C and OGC standard specifying the semantics of sensors, observations, sampling, and actuation, *Semantic Web* **10**(1) (2019), 9–32.

[24] J. Werbrouck, M. Senthilvel, J. Beetz and P. Pauwels, Querying heterogeneous linked building datasets with context-expanded graphql queries, in: *7th Linked Data in Architecture and Construction Workshop*, Vol. 2389, 2019, pp. 21–34.

[25] P. Winstanley, A. Perego, R. Albertoni, A.G. Beltran, S. Cox and D. Browning, Data Catalog Vocabulary (DCAT) - Version 2, W3C Recommendation, W3C, 2020, https://www.w3.org/TR/2020/REC-vocab-dcat-2-20200204/.

[26] M. Bonduel, A Framework for a Linked Data-based Heritage BIM, 2021. $$Uhttps://lirias.kuleuven.be/retrieve/618662$$DPhD_dissertation_MathiasBonduel.pdf[freelyavailable].

[27] M. Senthilvel, J. Oraskari and 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*, CEUR-WS. org, 2020.

[28] R. Albertoni, D. Browning, S. Cox, A. Gonzales Beltran, A. Perego, P. Winstanley and M. Dekkers, Data Catalog Vocabulary (DCAT) - revised edition, W3C Editor's Draft, W3C, 2019, https://agreiner.github.io/dxwg/dcat/#ldp.

[29] E. Mansour, A.V. Sambra, S. Hawke, M. Zereba, S. Capadisli, A. Ghanem, A. Aboulnaga and 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.

[30] J. Werbrouck, R. Taelman, R. Verborgh, P. Pauwels, J. Beetz and 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.

[31] J. Werbrouck, P. Pauwels, J. Beetz and E. Mannens, Data patterns for the organisation of federated linked building data, in: *LDAC2021, the 9th Linked Data in Architecture and Construction Workshop*, 2021, pp. 1–12.

[32] A. Malcolm, J. Werbrouck and P. Pauwels, LBD server: Visualising Building Graphs in web-based environments using semantic graphs and glTF-models, in: *5th Symposium Formal Methods in Architecture*, Springer, 2020.

[33] J. Beetz, L. van Berlo, R. de Laat and P. van den Helm, BIMserver. org–An open source IFC model server, in: *Proceedings of the CIP W78 conference*, 2010, p. 8.

[34] T.-J. Huyeng, C.-D. Thiele, A. Wagner, M. Shi, A. Hoffmann, W. Sprenger and U. Rüppel, An approach to process geometric and semantic information as open graph-based description using a microservice architecture on the example of structural data, in: *EG-ICE 2020 Workshop on Intelligent Computing in Engineering*, L.C. Ungureanu and T. Hartmann, eds, Universitätsverlag der TU Berlin, Berlin, 2020. http://tubiblio.ulb.tu-darmstadt.de/121623/.

[35] N.V. Hoang and S. Törmä, Drumbeat platform—a web of building data implementation with backlinking, in: *eWork and eBusiness in Architecture, Engineering and Construction*, CRC Press, 2017, pp. 155–163.

[36] H. Halpin, P.J. Hayes, J.P. McCusker, D.L. McGuinness and H.S. Thompson, When owl: sameas isn't the same: An analysis of identity in linked data, in: *International semantic web conference*, Springer, 2010, pp. 305–320.

[37] R. Taelman, J. Van Herwegen, M. Vander Sande and R. Verborgh, Comunica: a modular SPARQL query engine for the web, in: *International Semantic Web Conference*, Springer, 2018, pp. 239–255.