# ConSolid: a Federated Ecosystem for Heterogeneous Multi-Stakeholder Projects

Jeroen Werbrouck [a,c,d,*], Pieter Pauwels [a,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.** In many industries, multiple parties collaborate on a larger project. At the same time, each of those stakeholders participates in multiple independent projects simultaneously. A double patchwork can thus be identified, with a many-to-many relationship between actors and collaborative projects. One key example is the construction industry, where every project is unique, involving specialists for every subdomain, ranging from the architectural design over technical installations to geospatial information, governmental regulation and sometimes even historical research. A digital representation of this process and its outcomes requires semantic interoperability between these subdomains, which however often work with heterogeneous and unstructured data. In this paper we propose to address this double patchwork via a decentral ecosystem for multi-stakeholder, multi-industry collaborations dealing with heterogeneous information snippets. At its core, this ecosystem, called ConSolid, builds upon the Solid specifications for Web decentralisation, but extends these both on a (meta)data pattern level and on microservice level. To increase the robustness of data allocation and filtering, we identify the need to go beyond Solid's current LDP-based interfaces to a Solid Pod and hence introduce the concept of metadata-generated 'virtual views', to be generated using a SPARQL interface to a Pod. Building on top of these generic interfaces, domain-specific (higher-level) interfaces can be set up. A recursive, scalable way to discover multi-Pod aggregations is proposed, along with data patterns for connecting and aligning heterogeneous (RDF and non-RDF) resources across Pods in a mediatype-agnostic fashion. We demonstrate the use and benefits of the ecosystem using minimal running examples, concluding with the setup of an example use case from the Architecture, Engineering and Construction (AEC) industry.

Keywords: Solid, Federated Data Catalogs, Collaboration Platforms, Metadata, Semantic Enrichment

## 1. Introduction

### *1.1. A Double Patchwork of Stakeholders and Projects*

Interdisciplinary collaborations often involve intensive information exchange between domain experts with various backgrounds. In many cases, a 'real world' product will be involved, which will interact with its environment for its entire lifecycle. When these interactions are to be captured digitally, e.g. to make predictions or invoke active interactions with physical surroundings, the concept of 'digital twins' comes into sight: an as-complete-as-possible digital representation of a physical object - a core concept of Industry 4.0 [1]. The example industry used in this paper is the Architecture, Engineering and Construction (AEC) industry, one of the most decentralised industries

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

[2]. In the AEC industry people from many professions interact with the asset in all of its life cycle phases, ranging from direct stakeholders such as architects and engineers, commissioners, contractors, workers, facility managers, and inhabitants to indirect partners such as governmental agencies and product manufacturers. Because most stakeholders will be involved in multiple projects at the same time, we can generally speak of a 'double patchwork', a many-to-many relation between stakeholders and collaborative projects (Figure 1).

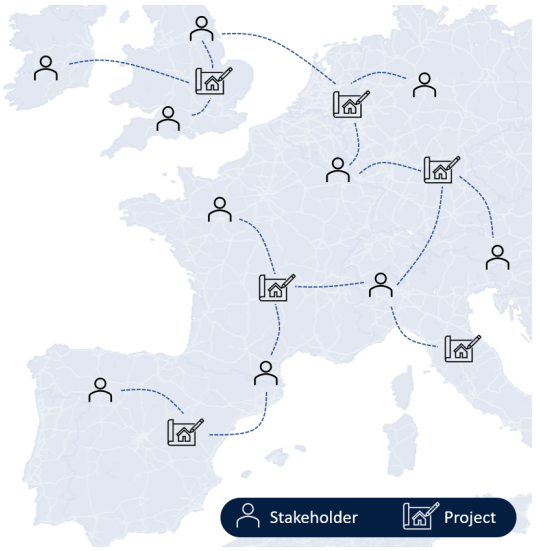


Fig. 1. A double patchwork of stakeholders and collaborative projects

Since every activity in the process requires input from different disciplines, and every discipline has its own workflows and data formats, it is impossible to determine a 'complete' data model for describing a product's digital twin at the start of the project. As a matter of fact, every model is by definition incomplete. Instead, a modular approach allows to gradually build the project model whenever input from a new discipline is required. Often, Semantic Web and Linked Data technologies are identified as apt technologies to achieve interdisciplinary data mapping, logical inferencing [3] and query-based discovery of federated data. 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, sensor streams and geometry [4]. This heterogeneity, which is inherent to large industries such as the AEC industry, is acknowledged by both academia and industry, but efficient solutions for dynamic management of such resources are currently lacking, especially given their decentral production.

The fact that the amount and nature of the disciplines involved is dynamic and depending on the use case, also implies that it is very impractical to try to centralise all this information in a single 'Common Data Environment' (CDE): especially contextual information (geospatial, governmental, historical) should not be duplicated every time, but just referenced semantically, as such information will be relevant for many projects at the same time. In essence, a 'central CDE' can thus be considered federated from the moment it refers to external contextual information - although they are mostly not created with data federation at their core, making it difficult to work with external information in an integrated way. However, also for project-specific information, uploading everything to a centralised, commercial cloud hub causes issues regarding data ownership (intellectual property) and data sovereignty ('how can this data be used legally') [5].

The technologies for a federated collaboration ecosystem are available, though. 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.) [6]. Upcoming open Web specifications such as WebID-OIDC[1] allow the creation of a decentral, potentially 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. This is one of the core enabling technologies for the specifications of the Solid[2] project [7], where WebIDs are used to control access to resources on a 'Personal Online Data storage' or, shortened, 'Pod'. In the case of multi-stakeholder, multi-resource collaborative projects, this means that a company can authenticate to the Pods of other project participants to access their (filtered) interdisciplinary 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 cloud provider or the need for its permissions to use their API. Up to this point, the main focus of Solid has been on single-Pod environments. However, multi-Pod environments require a different approach in many cases.

### *1.2. Research Questions*

We identify the need for a *federated, multi-purpose, cross-domain ecosystem for heterogeneous digital twins*. The ecosystem should be domain-agnostic at its core, but extendable to address the needs of and interoperability between specific industry domains. Hence, we can formulate the main research question of this paper as following:

**RQ**: Is it possible to devise *domain-agnostic* data patterns for a *federated, multi-purpose, cross-domain ecosystem for heterogeneous digital twins* based on the Solid specifications? What (meta)data patterns and service architecture could empower this ecosystem?

In devising our approach to answer this research question, we will take the following hypotheses and resulting objectives:

1. The ecosystem starts from the Solid specifications at its core but it will be necessary to extend the current specifications, regarding interfaces to a Solid Pod, service infrastructure and data patterns. Contrasting with known existing work on Solid, the ecosystem will treat a Solid Pod as a hybrid knowedge graph instead of as a heterogeneous file system. This is needed to address multi-purpose use of resources and aggregations of datasets living on different Pods in a secure way.
2. Security (authorization and authentication) is considered to be managed by the Solid specifications, unless a custom deviation or extension of the specifications is made. The use of federated authentication is considered an important requirement for addressing the double patchwork, to avoid the frequently occurring practice of creating a new account on different project management platforms each time a new consortium is formed with a different setup of collaborative platforms.
3. Although the resources in the project may be *heterogeneous* (i.e. RDF and non-RDF), the metadata that connects them into a larger project needs to be RDF-based in order to allow maximal semantic freedom. This way, every domain can enrich the metadata with discipline-specific information, to enhance discovery and filtering of data, according to the use case at hand.
4. Membership of one or more larger projects does not directly impact a resource, i.e. the resource should be functional in a standalone case as well as in different multi-resource aggregations. To make the ecosystem more than a collection of datasets, the ability to establish sub-document links between non-RDF datasets will be a prerequisite.
5. The ecosystem should be non-exclusive, i.e. it should be possible to integrate any dataset on the Web in the project without duplicating them, also when they are regular Web resources not originally embedded in the ecosystem.

In the following sections we will investigate the current challenges for such layered ecosystem to work. Although we will take the use case of the AEC industry to identify the requirements of the ecosystem, we aim for a solution

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

that is generally applicable. To guarantee the general applicability of the framework, we will describe and address every sub-challenge in a domain-agnostic fashion, and give examples related to the AEC industry. In the remainder of the paper, the ecosystem will be called ConSolid.

### *1.3. Relationship to earlier work*

This paper is the continuation of previous research on enabling technologies for a federated CDE. It was proposed in Werbrouck et al. [6] 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 Werbrouck et al. [8], which describes a methodology for pattern-based access control to AEC datasets. In contrast with username- or group-based approaches, this experimental approach allows to specify the properties someone should have to access data (e.g., 'the architect of the project as approved by the project owner'), based on the SHApes Constraint Language (SHACL)[3] standard. An initial proposal for patterns to connect heterogeneous datasets in a federated ecosystem was described in Werbrouck et al. [9], based on data conversion patterns described in Malcolm et al. [10]. However, the proposed data patterns still featured multiple avoidable complexities. 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, in earlier research the sub-document linking patterns focus on linking geometry, but do not provide a generic way to link heterogeneous information. The concepts of 'virtual containers' and 'virtual views', which create discipline-specific interfaces on top of a Solid Pod or multi-Pod configuration, were introduced in Werbrouck et al. [11] and will largely contribute in simplifying the early versions of the ecosystem and make it more robust, scalable and domain-agnostic. This paper organises and improves the insights and data patterns from this previous work. It introduces novel concepts with respect to this earlier research, such as considering the Pod a hybrid, metadata-based knowledge graph instead of a folder system of more or less independent researchers, scalable aggregation mechanisms and sub-document, references that are independent of mediatype and data owner.

### *1.4. Paper overview*

In the next section (Section 2), background technologies and related work will be discussed. In Section 3, we will motivate the extensions that we need to make to the current Solid infrastructure in order to achieve the goals of the envisaged ecosystem, regarding interfaces, microservices and metadata patterns. Section 4 then builds upon those extensions to propose aggregation structures for single- and multi-Pod configurations, and Section 5 devises the patterns for asynchronous sub-document linking. A minimal proof-of-concept from the AEC industry is then described in Section 6. The paper concludes with a discussion and overview of future work (Section 7). Definitions related to the data patterns used in the ecosystem are published as the ConSolid vocabulary[4]. The different software prototypes discussed in this paper are aggregated at *https://github.com/ConSolidProject/infrastructure*.

## 2. Related Work

This section will cover key technologies for data aggregation and containerisation, as well as the fundamentals of the Solid ecosystem. Finally we review some related initiatives, from the perspective of the AEC industry and perspective of the decentralised Web.

[3] https://www.w3.org/TR/shacl/
[4] https://w3id.org/consolid#

### 2.1. Containerisation of 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' [12]. A multi-model can be organised in (nested) containers or catalogs, which, together with metadata descriptions, aid in discovering the right datasets for the right task. Two approaches are hereby considered relevant: the Linked Data Platform (LDP)[5], and the Data Catalog Vocabulary (DCAT)[6] specification.

The Linked Data Platform specification (LDP) 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 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. LDP is currently the main interface to discover and retrieve information on a Solid Pod (Section 2.2.1).

The Data Catalog Vocabulary (DCAT) is *an RDF vocabulary designed to facilitate interoperability between data catalogs published on the Web* [13]. DCAT defines a domain-agnostic way of aggregating federated datasets in a `dcat:Catalog`. A catalog aggregates 'datasets' (`dcat:Dataset`); a `dcat:Dataset` instance defines the metadata about a dataset and may in turn either contain other datasets as well (a `dcat:Catalog` is an `rdfs:subClassOf dcat:Dataset`) or point to one or more 'distributions' (`dcat:Distribution`). Distributions essentially represent the actual information of the dataset, differing from one another, e.g., concerning their data type or version. Once a `dcat:Distribution` is identified, the actual data may be retrieved by dereferencing either the `dcat:downloadURL` (retrieve a dump of the dataset) or the `dcat:accessURL` (access the dataset via a database endpoint, e.g., SPARQL, timeseries etc.) (Figure 2). Such endpoints can be further described via `dcat:Service` instances, e.g., to indicate whether the service conforms to a specific standard such as SPARQL (`dct:conformsTo`) or list the datasets provided by the service (`dcat:servesDataset`). Ontological extensions for using the DCAT vocabulary to describe metadata of AEC project resources have been proposed in [14], such as the Construction Dataset Context (CDC) ontology[7].

### 2.2. Web Decentralisation Ecosystems

Over the last years, the concept of 'data sovereignty' of actors in the digital economy has known growing interest [5]. Data sovereignty is closely related to data security and protection. It is a concept to make data usage and storage subject to national privacy laws and regulations of the country where it was collected (not where the data is stored), so organisations cannot use it for whichever purpose that suits their interests. One of the core ideas is that the owner of the data remains in control [15]. The most well-known regulation for data sovereignty is the EU's General Data Protection Regulation (GDPR), which defines how personal data should be processed and moved between agents. However, the majority of web services still rely on centralised data storage, which means that it is still up to the service providers to comply to these regulations. Numerous recent initiatives therefore focus on decentralised or federated data storage, where the owner of the data ultimately gets to decide on which server their data resides and who is allowed to interact with the data. In the below paragraphs, we will briefly discuss some of these projects.

#### 2.2.1. 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 typically consist of restricted project-specific data and open contextual datasets. Most common implementations rely on authentication mechanisms that

[5] https://www.w3.org/TR/ldp/
[6] https://w3c.github.io/dxwg/dcat/
[7] https://mathib.github.io/cdc-ontology/

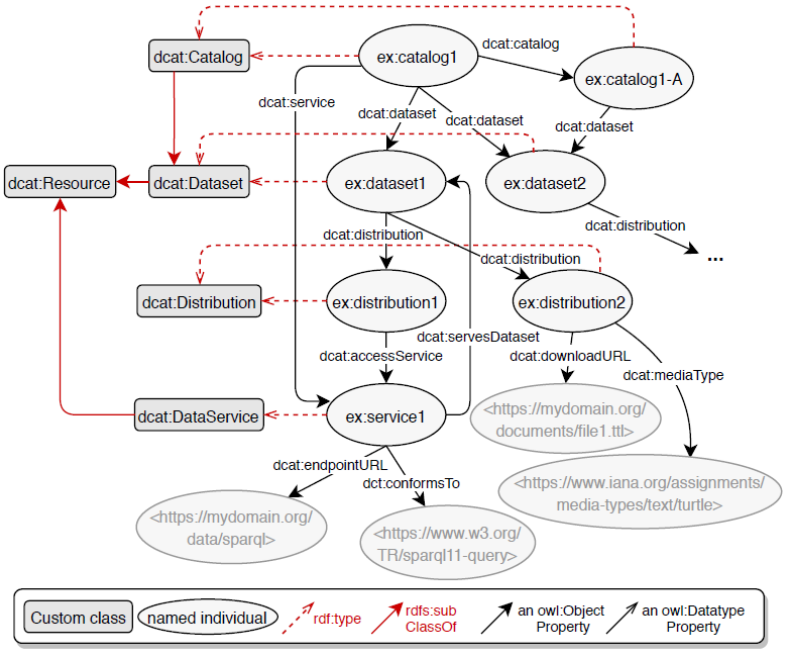


Fig. 2. Core DCAT concepts, figure from [14]

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). Different is the Solid project for Web decentralisation [7], which adds a decentralised authentication layer on top of a heterogeneous resource server, with LDP as the primary interface. 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 1).

```
<https://jeroen.werbrouck.me/pod/profile/card#me>
    foaf:name               "Jeroen Werbrouck" ;
    rdf:type                schema:Person, foaf:Person ;
    solid:oidcIssuer        <https://broker.pod.inrupt.com/> .

<https://jeroen.werbrouck.me/pod/profile/card>
    foaf:primaryTopic   <https://jeroen.werbrouck.me/pod/profile/card#me> ;
    foaf:maker          <https://jeroen.werbrouck.me/pod/profile/card#me> ;
    rdf:type            foaf:PersonalProfileDocument .
```

Listing 1: Example solid "card" (Turtle syntax). The WebID is a specific resource in the card that can be semantically enriched with RDF. Prefixes are included in Appendix A.

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[8] (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 authen-

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

ticate to any Solid-enabled service - hence identity verification can happen without third-party companies acting as a middle-man.

As Solid relies on the LDP specification, a Solid server may host both RDF and non-RDF data. Access control rules are expressed in 'Access Control Lists' ('.acl'), which relate specific `ldp:Resources` and `ldp:Containers` to specific (groups of) actors via their WebID, using the Web Access Control[9] (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 2. An upcoming replacement technology for ACL is the Access Control Policy[10] (ACP) specification, which will allow more semantic freedom to express access control rules.

```
# 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 <https://jeroen.werbrouck.me/pod/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
```

Listing 2: Example ACL document, regulating access to a specific LDP container and its resources.

The general use of Solid as an enabler of decentralised digital ecosystems is described in Verstraete et al. [15], from a techno-economic perspective. Furthermore, in earlier work (Section 1.3), we have identified the Solid specifications as a suitable basis for the needs of a digital AEC industry. This has multiple reasons:

1. It allows a stakeholder-centric approach to manage collaborative projects in a federated way.
2. It supports storage and linking of heterogeneous datasets.
3. It supports the 'Single source of truth' (SSoT) principle by design, as services do not locally duplicate information but all interact with the same data.

A comparison between the original goals of Solid and an interpretation for the AEC industry is given in Figure 3. The Solid specifications and their implementation in the Community Solid Server[11] will provide the main infrastructure for the ecosystem in this paper.

### *2.2.2. International Data Spaces*

The International Data Spaces (IDS)[12] initiative is a recent, multi-disciplinary initiative oriented towards data sovereignty in the digital economy [18]. The IDS ecosystem is based on the so-called IDS-RAM (Reference Architecture Model), which defines ways to let agents such as Data Providers and Consumers, Brokers, Identity Providers, App Store and Service Providers and Vocabulary Providers interact with one another to facilitate trust between actors that the data they exchange is going to be used only for specific use cases and intentions ('Usage policy enforcement'). IDS is a core component of the ongoing European GAIA-X project[13] (2019), which may complement the

[9] https://github.com/solid/web-access-control-spec
[10] https://solidproject.org/TR/acp
[11] https://github.com/CommunitySolidServer/CommunitySolidServer
[12] https://internationaldataspaces.org/
[13] https://gaia-x.eu/

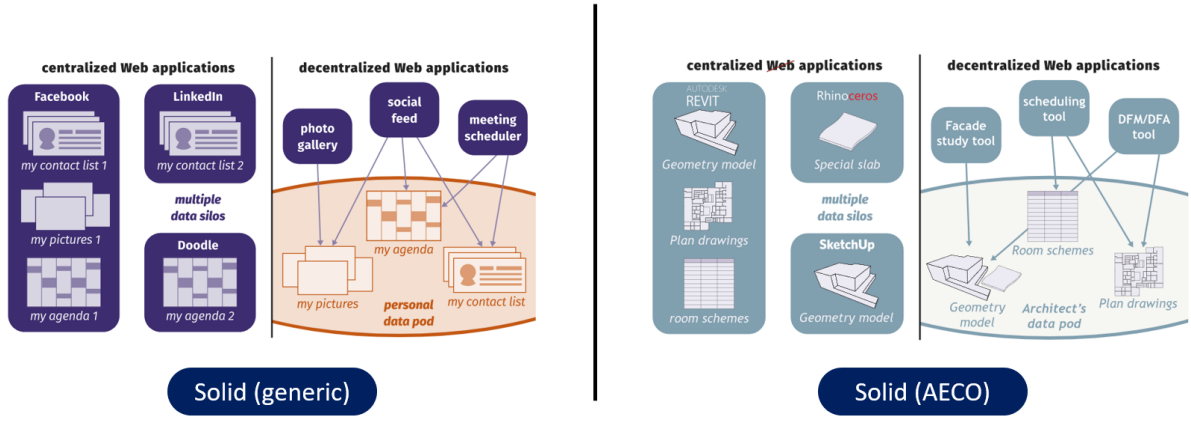


Fig. 3. Web applications in the solid ecosystem [16] and applied to the AEC industry [17]

IDS architecture with technologies for data storage, data compliance and cloud infrastructure [19]. GAIA-X will be applicable as an additional layer on top of existing cloud platforms. A comparative analysis between IDSA, Gaia-X and Solid is given in [15].

#### *2.2.3. Interplanetary File System*

The Interplanetary File System (IPFS)[14] is a peer-to-peer protocol for resilient, distributed storage of hypermedia [20]. Instead of HTTP URIs, IPFS 'content' is identified by its cryptographic hash (its 'Content Identifier' or CI). Many nodes in the IPFS network can serve (snippets of) this content upon request, making it much more resilient than resource storage on a single server. When content changes, the CI changes as well, so it has build-in protection against data tampering. Version management for a given resource is possible, however, using the 'Interplanetary Naming System' (IPNS), which provides a dereferenceable key to find the most recent version of the resource. Ongoing (early) research investigates the use of IPFS as a backend for Solid Pods [21].

## 3. Interfaces to a Discovery-oriented Solid Pod

### *3.1. Stripping implicit semantics from URLs*

As mentioned in Section 2.2.1, a Solid Pod can be seen as a resource server with a decentralised authentication layer on top. The authentication layer communicates with the IDP of the visiting agent to see if they are allowed to interact with resources on the Pod. The spine of a Pod is currently formed by the LDP specification, which defines patterns for reading and writing RDF data, based upon RESTful HTTP, and allows a containerisation of resources. The result resembles a classic folder-based file system. Every resource is retrievable via a URL by concatenating the origin with the respective containment branches separated by slashes - much like a Web-based file system indeed.

While LDP can essentially be seen as just one of the many possible APIs on top of the Pod, its current application hard-wires 'implicit' (i.e. non RDF-based) semantics in the URLs of resources and imposes a tree-like folder structure on what essentially is a (hybrid) knowledge graph (KG). I.e., every URL contains the URLs of their parent directories up until the 'root' of the Pod. This is a design choice embedded in the Solid specifications, which enables quick inference of a resource's parent containers (a feature called 'slash semantics') and its governing access control rules. However, at the same time it also imposes a quite rigid structuring of resources, because it implies that there is only one possible (direct) parent folder, and resource URLs inherit the (often arbitrary) tags of all their (recursive) parent folders. This while these parent directories may change over time, thereby invalidating the resource URL.

[14] https://ipfs.io/

We consider the following example, related to the stages of publication (Work-in-progress, Shared, Published, Archved) as defined in ISO 19650 [22], the authoritative standard in the AEC industry which defines the basic concepts for information management in a Common Data Environment. Moving a resource from the folder '/work-in-progress' to '/shared' will change its URL on the Pod from 'https://jeroen.werbrouck.me/pod/work-in-progress/file1' to 'https://jeroen.werbrouck.me/pod/shared/file1', thereby breaking any reference pointing to the original URL. Moreover, in a multi-purpose collaboration platform, the containment of a resource in this specific parent LDP container might only be relevant in a specific situation but totally illogical in others: maybe someone would like to aggregate their resources in a different container structure when addressing a different use case. Hence, it makes sense to strip these implicit containment semantics from the URL as much as possible, so the URL can be reduced to a string of the form 'root + GUID'. Note that this 'form' still follows the LDP specification, as it allows to interact with LDP resources via HTTP - there are just no slash semantics beyond the root of the Pod, which remains essentially a (very large) `ldp:Container` (Figure 4).

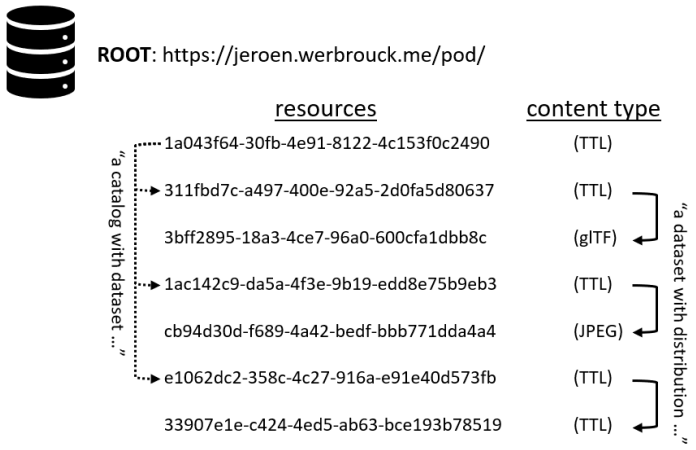


Fig. 4. A flat list of resources in a Pod. Containment triples and metadata are mentioned using RDF instead of slash semantics, allowing resources to be grouped flexibly.

The 'meaning' that allows semantic containerisation and filtering on higher-level (domain-specific) layers must now come from a metadata record that 'enriches' the heterogeneous resource using RDF ontologies, instead of from the parent container. Metadata records, which are still very much under discussion in the Solid ecosystem, can be updated without changing the resource or its URL itself: for example, the URL of the resource remains the same, but changing a metadata tag from 'work-in-progress' to 'shared' allows the original URL of the resource to remain intact, while a use-case specific 'virtual' folder structure can be created using a query-based discovery pattern [11]. In other words, query-based resource discovery allows to get rid of implicit semantics in a URL, as resource discovery does not depend anymore on URL destructuring. Multiple 'views' can be constructed on the same resource, and different containment configurations can be created depending on the use case, on a higher interface level in the ecosystem. However, to be able to perform these virtual views, which are essentially just filters on Pod data, the Pod must be accessible as an (access-controlled) knowledge graph which is the union of its resources, parallel to the resource-oriented interface of the Linked Data Platform that is currently implemented. We consider a SPARQL endpoint an apt interface to query this knowledge graph. Obviously, the same URLs that are used to serve resources via HTTP (LDP) can also be used as URIs for named graphs in a SPARQL endpoint, in order to facilitate the access control protocols as specified in Solid. Considering that a typical SPARQL endpoint does not implement these protocols, all requests to the SPARQL endpoint then need to pass via a proxy service, which may perform access control on each triple pattern in the query by querying for the named graph they reside in. An example query and its internal modification are given in Listing 3. In the remainder of this text, we will consider the proxy service and database service as one, calling it a 'satellite' to the Pod.

```
# The original SPARQL query
SELECT ?element ?dam ?cause WHERE {
    ?element a beo:Wall ;
        dot:hasDamage ?dam .
    ?dam dot:hasCausation ?cause .
}

# The modified SPARQL query.
# Query parameters g1, g2 and g3 are used to check access to the results.
# Only if someone has access to all graphs, the result can be included in the final set.
SELECT ?element ?dam ?cause WHERE {
    graph ?g1 { ?element a beo:Wall .}
    graph ?g2 { ?element dot:hasDamage ?dam .}
    graph ?g3 { ?dam dot:hasCausation ?cause .}
}
```

Listing 3: The SPARQL satellite service needs to check the resource a triple pattern occurs in, in order to check access control before passing the query result to the client. An example query is used to query for registered damages of an element.

The SPARQL satellite should be easily discoverable, which can be done by registering the triple pattern in Listing 4 in the WebID of the owner of the Pod[15].

```
<https://jeroen.werbrouck.me/pod/profile/card#me>
        consolid:hasSparqlSatellite <https://satellite.example.org/bob/sparql> .
```

Listing 4: Triple pattern to find the SPARQL satellite to a Pod, via the WebID of the Pod's owner.

### *3.2. Discovery and Storage of Heterogeneous Datasets*

File-based resources on the Web are inherently heterogeneous. In industries such as the AEC industry, a plethora of file formats is used for different use cases: although RDF-based knowledge description is becoming more popular, non-RDF datasets (e.g. imagery, point clouds and geometry) will continue to play a big role in many situations. However, using RDF on a *metadata* level can offer a lot of benefits without changing the nature of the resources themselves - RDF then acts as the 'semantic glue', flexibly connecting heterogeneous, federated resources on the Web in a 'hybrid' knowledge graph. Because the approach discussed in Section 3.1 eliminated implicit URL-based semantics, every resource needs at least one RDF-based metadata record to allow discovery of resources based on filtering queries. This is particularly of value to non-RDF resources, since e.g. binary resources cannot be directly queried with SPARQL - but a metadata record allows registration of queryable semantic descriptions. The SPARQL interface mentioned in Section 3.1 must expose a union of all metadata records, allowing rapid discovery of specific resources - given the requesting agent has the necessary reading rights (which we consider to be the case for the example requests given in this paper). As a vocubulary to structure this metadata network, we will opt for the DCAT vocabulary (Section 2.1), contrasting with Solid's default usage of LDP for containerisation of resources. This has the following reasons:

1. Addressing 'containerisation' and expressive metadata descriptions with a single, integrated vocabulary .
2. Semantic decoupling of metadata records and actual resource, which allows RDF-based discovery of heterogeneous datasets and semantic indication of a distribution's versions and content-type.
3. Avoiding conflicts with the default Solid's usage of LDP.
4. The flexibility to let metadata records semantically indicate versions of distributions, which is currently not an option in Solid.

[15] We consider the WebID to be located on the Pod, although this is not strictly enforced by the Solid specifications

Listing 5 gives an example of an RDF-based metadata record in the ecosystem, aggregated in a larger catalog, using DCAT.

```
@prefix pod: <https://jeroen.werbrouck.me/pod/> .

# The catalog resource, with URL pod:0d1ffe69-6d69-4b76-ae79-87aacbfc2ca3
pod:0d1ffe69-6d69-4b76-ae79-87aacbfc2ca3 a dcat:Catalog ;
    dcat:dataset pod:1e19ed7c-2809-408b-a619-f27e601d7423 .

# The dataset resource, with URL pod:1e19ed7c-2809-408b-a619-f27e601d7423
pod:1e19ed7c-2809-408b-a619-f27e601d7423 a dcat:Dataset ;
    ex:publicationStatus ex:workInProgress ;
    dcterms:created "2022-07-29T14:13:28.167000"^^xsd:dateTime ;
    dcat:distribution pod:90329a28-6663-44c0-8e40-deb50a9e24b1 .

pod:90329a28-6663-44c0-8e40-deb50a9e24b1 a dcat:Distribution ;
    dcat:downloadURL pod:90329a28-6663-44c0-8e40-deb50a9e24b1 . # Where to find the file.
```

Listing 5: Catalog and metadata record of a resource, using the DCAT vocabulary. Both the catalog and the dataset are accessible as an HTTP resource, or as a named graph in the SPARQL satellite. An example vocabulary is used to denote the publication statuses.

The example of changing a resource's publication status (ISO 19650) while maintaining its URL (Section 3.1) now boils down to the following steps, considering we only know the URL of the actual resource. First, we need to discover the metadata graph of this resource via the SPARQL satellite (Listing 6). Now a SPARQL INSERT query to this resource may change the content of this metadata graph, allowing it to be included in a new virtual container for shared resources, and excluded from the one for resources tagged 'work in progress' (Listin 7).

```
SELECT ?mdGraph WHERE {
GRAPH ?mdGraph {
    ?md dcat:distribution/dcat:downloadURL <{the-url-of-the resource}> .
}}
```

Listing 6: Discovery of a distribution's metadata resource via the SPARQL interface to the Solid Pod.

```
CONSTRUCT {?virtualContainer ldp:contains ?downloadURL }
WHERE {
  ?ds ex:publicationStatus "shared" ;
    dcat:distribution/dcat:downloadURL ?downloadURL.

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

Listing 7: SPARQL query to filter project datasets that are 'shared' and return their distributions as LDP containers

Moreover, when a resource's containment structure is no longer reflected in its URL, it can be embedded in multiple semantic containment structures at the same time. I.e., multiple resources can indicate containment of a certain resource, and multiple discovery paths become possible. Some of these discovery paths may be made explicit, while others could be generated dynamically by executing queries over the explicit knowledge. This means that a one-on-one mapping of a Pod's DCAT catalogs is not the only option: as documented in Werbrouck et al. [11], 'virtual' LDP containers (or DCAT catalogs) may be created based on (CONSTRUCT) queries, e.g. to group datasets according parameters such as media type, publication status or creation date.

### 3.3. Access control in a SPARQL-interfaced Pod

As indicated in Section 2.2.1, resources on a Pod are protected by ACL resources. In an exclusively LDP-based environment, the applying ACL document is found by searching for the 'closest' ACL resource: a feature allowed by the slash semantics feature. This means that either the ACL is linked directly to the resource to be found (as '{URL-of the-resource}.acl') or a stepwise approach is taken to find a general ACL document in one of the parent containers (the closest one is the one that counts). In a Pod environment that is primarily SPARQL-based, this is no option, as a resource can be 'contained' in multiple parent containers, which may impose conflicting ACL rules to their child resources or subcontainers. One option is to enforce that every resource has its dedicated ACL file. This option is fully allowed by the current Solid specifications, but might quickly result in an overload of ACL resources. Alternative possibilities, such as to allow a resource (via its metadata record) to semantically indicate its governing ACL resource, enable re-use of ACL resources, but most certainly break overall compatibility with the Solid specifications. This topic is not further explored in this paper and is considered in future research.

### 3.4. Domain-specific Interfaces

SPARQL interfaces to project Pods might be used to find containment information, but also allow other query-based views to be established. They can be used as a generic way to present a (set of) resource(s) on the Pod via industry-specific APIs, on a higher level. In the AEC industry, many subdisciplines are involved, all having specific internal information representation standards and agreements, but eventually they need to access the same data. Reading and discovering data could then happen via dedicated APIs on top of the Pod or of the entire federated project. Such APIs create virtual views on the knowledge graph of the project, filtering relevant information for a particular domain or use case. Since many industry standards are not RDF-based, the APIs also serve a mapping purpose: graph-based knowledge can be re-organised to fit particular industry standards, as is illustrated in Werbrouck et al.[11]. For example, Listing 7 can be part of an API-based, domain-specific interface that organises the project conforming to the ISO 19650 specification and its defined stages of publication (see Section 3.1). Other examples, such as a mapping to the Information Container for Linked Document Delivery (ICDD) (ISO 21597) [23] or the BIM Collaboration Format (BCF) API[16] are documented in Werbrouck et al. [11].

Domain-specific interfaces also include domain-specific access control regulators. For example, to implement a dynamic, pattern-based approach to check if the visitor has the necessary properties to do a specific action or allow them to see only data snippets that adhere to a specific shape (e.g. based on the use of SHACL). Retaking the example of the publication statuses, we can imagine a service that grants a given visitor access depending on a resource's publication status. I.e. when a resource still has the status 'work-in-progress', only in-office employees can access and modify it. From the moment it changes to shared, other stakeholders can view it as well, and when the document is 'published' or 'archived' even in-office employees cannot modify it anymore.

As soon as the SPARQL satellite is known, it can be used to discover other, higher-level interfaces of the Pod. An interface can be documented on the Pod just like any other resource, i.e., with a metadata record. In this framework, we choose to document satellites as `dcat:Services`. By indicating to which standard the service conforms (`dct:conformsTo`), a client which expects a certain API or form can quickly check whether there is a satellite that offers this particular view on the Pod. An example RDF description is given in Listing 8, indicating an interface that offers a view which is compliant to the BCF API, an industry specification for issue management in AEC projects.

```
<> a dcat:DataService ;
    # The BCF API is a standardised API to communicate issues related with BIM models
    dcterms:conformsTo <https://github.com/BuildingSMART/BCF-API> ;
    dcat:endpointURL <https://example.org/bcf-satellite> ;
```

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

```
    dcterms:description "Service for mapping ConSolid projects to the BCF (BIM Collaboration
    Format) API" .
```

Listing 8: RDF description of a BCF-compliant interface to the Pod, registered as a `dcat:Service`.

### *3.5. Implementation*

As a prototypical implementation of the setup described in this Section, we modified a Community Solid Server instance to forward any RDF-based information to a SPARQL store, next to offering HTTP access to all resources via LDP. The prototype setup contains the following components:

1. A SPARQL store (Apache Jena Fuseki) with the option `tdb:unionDefaultGraph` set to true. This allows to query the Pod as the union of its resources.[17]
2. A Solid server based on the Community Solid Server which forwards all RDF-related events (resource creation/update) to the SPARQL store (Apache Jena Fuseki 4.5.0).[18]
3. A NodeJS (ExpressJS) proxy server to the SPARQL store, implementing WebID-based authentication and authorization. Any external agent should only be able to query the SPARQL endpoint through this proxy. Unauthorized SPARQL results are filtered out by this proxy service.[19]

Higher-level interfaces, such as a satellite for sensor data or domain-specific interfaces, can be easily added using `dcat:Service`-s.

## 4. Discovery and Aggregation of Project Datasets

In Section 3, we set the basic infrastructure for a Pod to allow Pod-wide queries and the application of virtual views. Using the DCAT standard, metadata records and the resources they represent were linked semantically as `dcat:Dataset`-s, `dcat:Distribution`-s and their `dcat:downloadURL`-s, all available in a union graph exposed via a SPARQL endpoint. However, these resources were not yet aggregated into a larger whole, a 'multi-model' in AEC terms [12]. In this section, we will again use the DCAT specification to provide a scalable approach for describing and aggregating information in both single- and multi-Pod aggregations.

### *4.1. Dataset Aggregators*

We define a 'Dataset Aggregator' as a collection of pointers to relevant datasets on the Web. This collection may be automatically generated, based on specific parameters or queries (i.e. it may take the form of a virtual view [11]), or manually curated. Dataset Aggregators will form the core to maintain federated multi-models in a scalable way in the ConSolid project.

The initial level of aggregation occurs when an actor bundles resources on a single Pod, for example, when grouping all resources on the Pod that belong to the same project. As shown in Listing 9, this boils down to creating a `dcat:Catalog` instance and aggregating its constituent datasets via `dcat:dataset`, alongside indicating metadata for the project to be easily queryable. This is a local, Pod-specific project definition which follows the DCAT specification exactly.

```
@prefix pod: <https://architects.example.org/pod/> .
@prefix otherPod1: <https://engineers.example.org/pod/> .
@prefix otherPod2: <https://hvac.example.org/pod/> .

```

[17] https://dlcdn.apache.org/jena/source/jena-4.5.0-source-release.zip
[18] https://github.com/LBD-Hackers/SolidCommunity_Fuseki
[19] https://github.com/LBD-Hackers/lbdserver-sparql-satellite

```
# the Dataset Aggregator resource, at URL pod:d07af06a-4a94-48fd-99d1-f164f7e3a04c
<> a dcat:Catalog, consolid:DatasetAggregator ;
    rdfs:label "myFirstProject" ;

    # first level aggregation, locally on the Pod
    dcat:dataset pod:1e19ed7c-2809-408b-a619-f27e601d7423 ,  # see Listing 5
        pod:32ad4402-51ed-4c74-9ffb-bde2e5fa1540 ;

    # second level aggregation, including access points from two other stakeholders
    dcat:dataset otherPod1:aa3c09de-7e86-4bf8-bbe0-08eb01dbbbfe ,
        otherPod2:bd503663-3583-4d3e-a350-b0478fbe09f4 .
```

Listing 9: An aggregation which is only one level away from aggregating `dcat:Dataset` instances on a Pod.

All stakeholders can thus create their own local aggregator for a collaborative project, independently from the others, forming the main 'access point' on the Pod to find project data. This project can perfectly be called a multi-model of its own, but by referencing the catalogs of other stakeholders (on their Pods), it becomes straightforward for a client to discover and query also the contributions of these stakeholders. This effectively makes it an aggregator of second level. This situation is illustrated in Figure 5.

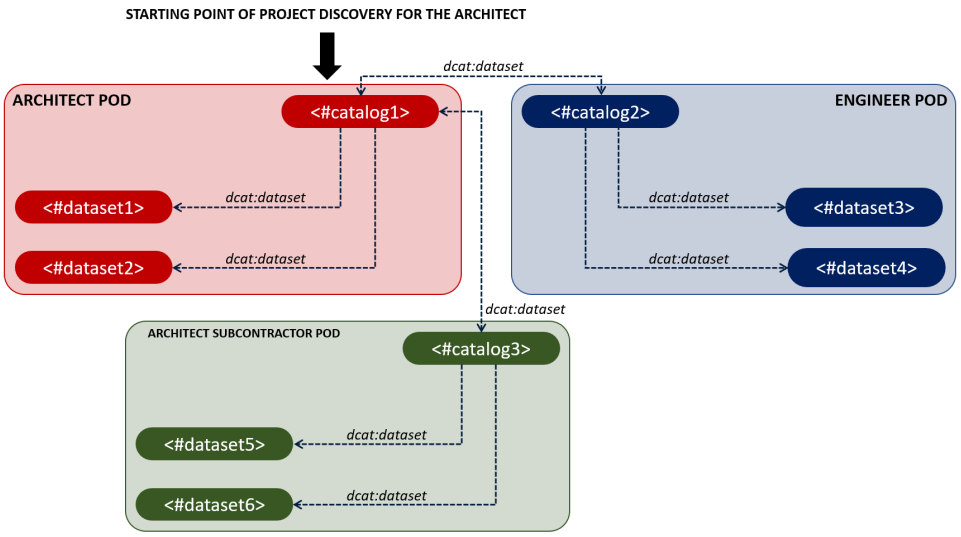


Fig. 5. Starting from one Dataset Aggregator in a Pod, it is possible to recursively find all datasets in the federated project.

Higher-level aggregations will work in exactly the same way. For example, an external actor (e.g. a governmental instance) may wish to create indexing aggregators for buildings of the same typology ('all public schools in Flanders', 'all bridges in Germany'). Potentially, these aggregators make use of sub-aggregators which group them according to geospatial location or creation year. No matter the 'depth level' of a catalog, it should eventually lead to discovery of all resources in the aggregator. As all aggregations are indicated with the `dcat:dataset` relationship, the depth of aggregation has no impact on discovery. This 'matryoshka' principle, possible because `dcat:Catalog` is a subclass of `dcat:Dataset`, gives us a simple yet powerful way to discover collections of information in a scalable and recursive way, using the query listed in Listing 10 and starting from one single Dataset Aggregator. One way of implementing such recursive discovery is to make use of 'link traversal'-enabled query engines [24], such as the Comunica [25] Link Traversal Engine[20]. Higher-level aggregations will make use of this recursive pattern to scale up the level of aggregation without increasing the complexity of the queries.

[20]https://github.com/comunica/comunica-feature-link-traversal

```
SELECT ?dataset
WHERE {
    <{url-of-the-catalog}> dcat:dataset+ ?dataset .
}
```

Listing 10: Recursive query to discover the metadata records of all datasets in an aggregator. Query engines that use link traversal can in this way quickly find all federated datasets in a project.

### *4.2. Integrating information created by subcontractors or individual employees*

A stakeholder office might be the legal entity responsible for some tasks in the project, but it is the human employees who divide the workload and do the eventual work. An office might decide to have a central office Pod, where all employees contribute, but it might also be the case that every employee maintains their own Pod, or alternatively that the office creates a dedicated Pod for its specific roles in the project. The above described catalog breakdown structure allows both situations to be easily integrated without changing the basic infrastructure. In that case, the office's project access point just aggregates the catalogs created in the employee Pods (Figure 5).

Similarly, an office might appoint a subcontractor for specific tasks in the project. These subcontractors will not work on the stakeholder's main Pod, but will have their own Pods to store their contributions. Often, a project counts a few 'main contractors' (e.g. an architect, structural engineer, HVAC engineer, owner). From an organisational perspective, not everyone should include all subcontractors of all other stakeholders. The breakdown structure simply allows any of the stakeholders to aggregate their subcontractor access points, making them discoverable to the other stakeholders as well, provided they have the correct access rights.

### *4.3. Implementation*

A prototypical API to interact with the basic setup described in Section 3.5 using the above mentioned patterns for storage and discovery of data, was created in context of this paper. The API is available on Github[21] and npm[22] as the Dataset Aggregation API ('daapi').

## 5. Aligning and enriching heterogeneous, federated data

The recursive catalogs described in Section 4 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, potentially federated datasets on a sub-document level. Sub-document identifiers of project resources will, in most cases, not be natively expressed using RDF. For example, the 3D element will probably have a sub-document GUID, and a particular pixel zone of the image will need to be described externally, as it is not an intrinsic part of the image. However, a semantic indication that an element corresponds with a wall can be expressed using domain-specific RDF statements. To relate all these sub-document identifiers as manifestations of the same 'thing X' is not only relevant from a data management perspective, but also to facilitate user-friendly interaction with project data: any identifier in any project resource can then be used to access all available information about 'thing X' in the project, independent of the mediatype of the resource that describes this information.

Apart from the federated access control and the heterogeneity of resources, information creation and information alignment can happen asynchronously amongst stakeholders. For example, indicating that a pixel zone on a picture identifies the same element modelled in the 3D model, and that, moreover, this element identifies a wall instance (stored in an RDF resource), can happen either (semi)instantly when uploading the picture (i.e. when the 3D object is used as a proxy to create and immediately align a local concept, used to 'enrich' the actual element) or later in the

[21] https://github.com/LBD-Hackers/daapi
[22] https://www.npmjs.com/package/consolid-daapi

project, when someone (person or digital service) recognises that the 3D object and the image actually represent the same element. Also, a scalable infrastructure as proposed in this paper should allow a local project (see Section 4) to function independently from other potential partial projects - as an office cannot oversee all external aggregations of their datasets. One should therefore acknowledge that there will always exist different, federated aliases of the same concept, rather than having one unique identifier for a concept that is potentially managed other actors.

Lastly, for a digital twin platform to function properly, it should be possible to differentiate between digital knowledge about the 'real world asset' and knowledge about the digital representations themselves. 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 an interface to further enrich the damaged area ("caused by erosion"), but also to comment on the pixel zone itself ("The damage is actually larger than the zone you identified - please extend"). The data patterns that address these challenges will be introduced in the following section, using the concept of 'Reference Aggregators', based upon the same aggregation principles as the Dataset Aggregators described in Section 4.

### *5.1. Reference Aggregators and Sub-document Linking*

We can generally define a Reference Aggregator as a way to group references according to a specific parameter. In the most high-level interpretation, a Reference Aggregator can group elements that represent the same concept (`consolid:ReferenceAggregator`). Because a recursive pattern will be used similar to the Dataset Aggregators, lower-level differentiation is possible when dictated by the project requirements. For example, to group representations which were valid during a specific time frame or for representations that situate the real-world element within a well-defined spatial boundary. However, to avoid overly complicated examples, we will continue in this paper with one-step aggregations that directly relate a reference to a concept aggregation, thus independent from temporal and spatial subfilters.

We can use a minimal pattern to 'lift' identifiers from a heterogeneous dataset and make it available for aggregation within a Reference Aggregator. This pattern breaks down references into three parts, and is inspired by the destructuring mechanisms in the Information Container for Linked Document Delivery (ICDD) (ISO 21597) [23]:

- A Concept Aggregation at the highest level, used for identifying the concept but not for directly enriching it. It aggregates references (`consolid:Reference`) to the same concept (`consolid:aggregates`). A Concept Aggregation can also recursively aggregate other Concept Aggregations, to allow discovery of alias aggregators of a 'thing' on other Pods.
- A Reference level, used to determine that a given identifier in a given file is referenced and relate metadata (e.g. creation date) to the reference (`consolid:hasIdentifier`).
- An identifier level; a higher-level URI used to 'even the field' between RDF and non RDF resources. This higher-level URI relates the value (`schema:value`) of the identifier (a Literal, to allow divergent identifier syntaxes beyond URIs) to the source that mentions it (`consolid:inDocument`). The standard to which the identifier and its parent document conform should also be indicated (`dct:conformsTo`).

The three-level pattern is visualised in Figure 6. These patterns are demonstrated with a real-world example in Section 6, Listings 14 and 15 (Turtle syntax).

Different Concept Aggregations can be aggregated themselves in a single named graph on the Pod, called a Reference Registry, which helps in discovery of the references present in a local project. Using the storage patterns devised in Section 3, this means that besides the RDF resource that semantically defines the aggregators, a metadata record is created describing a `dcat:Dataset` that is also an instance of `consolid:ReferenceRegistry`. This way, it is possible to quickly find and query the Reference Registry in a specific project.

We indicate that higher-level identifiers (`consolid:Identifier`) in Reference Aggregators evens the field between RDF and non-RDF resources, because they indicate a value which is by assumption only valid in a certain document. In other words, identification of sub-document objects and their allocation are expressed in separate triples. An identifier, whether RDF or not, is assumed valid only within the document that mentions it, as registered in the Reference Registry. This is a core aspect of the asynchronous enrichment, allowed by the combination of the higher-level (metadata-level) Reference Aggregations and their 'lower level' occurrences in actual resources - as registering references or removing them does not impact the documents themselves, and aliases can be created and

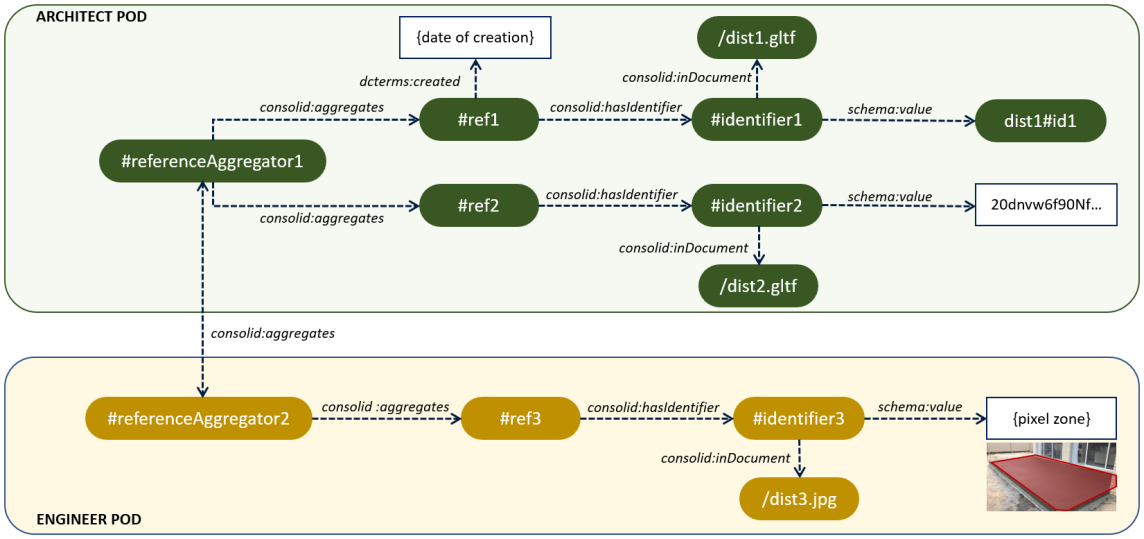


Fig. 6. RDF patterns from Reference Aggregator to sub-document identifier.

registered freely. Consequently, changing the URL of a document (e.g. at a data handover phase) does not impact its sub-document identifiers, and it is possible to asynchronously register the new location of the resource in the network. A detailed discussion on data handover is considered out of scope for this paper.

As illustrated in Figure 6 (and Listing 14), the recursive property `consolid:aggregates` allows a Reference Aggregator to aggregate equivalent Reference Aggregators on other stakeholder Pods, i.e. its aliases. In the example, a bi-directional aggregation exists between the Reference Aggregator in the architecture firm's Pod and the one on the Pod of the engineering company. Using this pattern, a client can select a representation of interest (e.g. using a visual interface showing a 3D representation of a wall in the building) and then find the distributions, datasets and sub-document identifiers of its other (local and remote) representations with a federated query. As these representations are meant to be generally applicable, they may lead to references in heterogeneous resources, from geometry and imagery over spreadsheets to textual descriptions. The queries given in Listing 11 and Listing 12 allow to find the references aggregated by the same Reference Aggregator. The first query should be executed over the SPARQL endpoints of the project consortium, to retrieve the aliases of interest and local references of the intended Reference Aggregator. Although most of them will return an empty result set, querying only the owner of the main document would be insufficient, as people can theoretically make sub-document references to resources that are not on their Pod. In the second query, the found aliases are used to find remote references. As these aliases are based on the URL of the Reference Registry that contains them, targeted queries can now be send to the corresponding SPARQL satellites. Additional metadata filters may be applied to only find specific references, e.g. only yielding geometric resources as referenced documents. This can be done by attaching a metadata triple pattern to the query, commented out in Listings 11 and 12.

```
SELECT ?value ?doc ?alias
WHERE {
    ?concept consolid:aggregates ?selectedReference, ?reference .
    ?selectedReference consolid:hasIdentifier ?selectedId .
    # Example: inject ?doc query result from validation in Section 6.3, Listing 15.
    ?selectedId consolid:inDocument <{the document containing the identifier}> ;
        # Example: inject ?el query result from validation in Section 6.3, Listing 15.
        schema:value "{the selected identifier}" .

    # find other references of this concept in the same reference registry
    ?reference consolid:hasIdentifier ?id .
    ?id consolid:inDocument ?doc ;
```

```
        schema:value ?value .

    # e.g. additional filter for the document media type
    # ?meta dcat:distribution ?dist .
    # ?dist dcat:mediaType <{media type of interest}> ;
    #    dcat:downloadURL ?doc .


    # find remote aliases of the concept
    OPTIONAL {
        ?concept consolid:aggregates ?alias .
        FILTER(?selectedReference != ?alias && ?selectedReference != ?reference)
    }
}
```

Listing 11: Query pattern to find the local references of a specific concept aggregator given a known reference, and the potential remote aliases of this concept.

```
SELECT ?value ?doc
WHERE {
    # Inject the ?alias query result from the query in Listing 10.
    <{alias}> consolid:aggregates ?reference .
    ?reference consolid:hasIdentifier ?id .
    ?id consolid:hasDocument ?doc ;
        schema:value ?value .

    # e.g. additional filter for the document media type
    # ?meta dcat:distribution ?dist .
    # ?dist dcat:mediaType <{media type of interest}> ;
    #    dcat:downloadURL ?doc .
}
```

Listing 12: Recursive query to find all representations, identifiers and datasets for a given selected identifier.

### *5.2. The Action of Linking*

The activity of linking references to an alias of the same concept will largely depend on the project, and is not directly handled by the ecosystem, which only provides the patterns to do so. Depending on the goal and size of the project, alignment can either be done manually or (semi-)automatically. A manual alignment makes sense if a dedicated GUI is available, and documentation happens on-the-go - while enriching existing project resources or while setting up a project in the ecosystem from its start. For example, a case of damage enrichment as documented in Section 6 can happen step-by-step by linking pictures to geometry, during the operational phase. When large-scale projects are imported, however, a manual concept alignment is not possible, given the volume of existing concepts. For diagnosing many aliases of the same concept, mapping algorithms such as proposed in Malcolm et al. [10] can be used in certain situations. With the advances made in the fields of Machine Learning and image recognition, third-party services are expected to provide opportunities here as well. However, as semantic relationships between elements do not happen at the level of references, but at the level of documents and resources, this is considered within the realm of domain-specific applications, and therefore outside the scope of this paper.

### *5.3. Enriching sub-document Identifiers*

References can not only be used to for aggregation of heterogeneous information related to the physical counterpart of the digital twin. They should also be usable to reference (comment, enrich, ...) 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. Because the references themselves are URLs, a new ('higher level') reference can be made elsewhere that has the first reference's URL as a `schema:value`. The comment is then registered via another reference, pointing to the document where the comment is made.

### 5.4. Implementation

A prototypical API to create and interact with Reference Aggregators and Reference Registries was created in context of this paper. As it makes use of the data patterns described in Section 4, this API depends on the lower-level implementation discussed in Section 4.3, *daapi*. The API is available on Github[23] and NPM[24] as the Reference Aggregation API ('raapi').

## 6. Demonstration

While the above sections explain the proposed ecosystem, 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 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. The demo concerns the documentation of a damage case, involving linking of the following heterogeneous datasets: federated semantics (a building graph at and a damage graph) with a 3D element and an image zone. The involved resources are managed by different stakeholders. The demonstration is reproduceable with the software prototypes mentioned in earlier sections, executing the scripts at https://github.com/LBD-Hackers/consolid-demo consecutively. However, the model used for reproducing the experiment bases on conversions of the (hypothetical) benchmark dataset Duplex[25], an IP-free model dataset.

### 6.1. Case description and Setup

To illustrate the data patterns discussed in this paper, we consider the following example setup, 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 BIM (Building Information Modelling) [26] models[26] for Architecture (modelled by Bureau Bouwtechniek (BE)), Engineering, Electricity, and HVAC (Heating, Ventilation and Air-Conditioning) (all three modelled by Arcadis (BE)) were converted to IFC (Industry Foundation Classes, ISO 16739)[27] [27] and then to a semantic model for RDF (Turtle) and a geometric model (glTF)[28].

In this simulation, we assume that every stakeholder maintains their own Pod and associated WebID and a SPARQL satellite. We consider three stakeholders: Bureau Bouwtechniek, Arcadis, and the Facility Management office of Ghent University, since it concerns a University Building. Each of them maintains their own contributions to the project in their Pod. Corresponding with their IPR, Bureau Bouwtechniek hosts the architectural model, Arcadis the three others. At this starting point of the operational phase, the FM office does not host anything yet.

Each converted semantic RDF model and geometric glTF model is stored as a distribution with attached metadata (dataset) on the creator's Pod. Metadata records for all project resources are mirrored as named graphs on the SPARQL satellite, as are the RDF-based semantics, derived from the partial models and structured using the Linked

[23] https://github.com/LBD-Hackers/raapi
[24] https://www.npmjs.com/package/consolid-raapi
[25] https://github.com/LBD-Hackers/consolid-demo/tree/4e1aaa982b5eb3bc63980fe9e5e6d041d3d54a6d/src/resources
[26] Partial models are sub-models for an AEC project related to a particular discipline and created by specialist stakeholders.
[27] The Industry Foundation Classes are the international open standard for digital description of built assets, developed by buildingSMART International.
[28] https://registry.khronos.org/glTF/specs/2.0/glTF-2.0.html

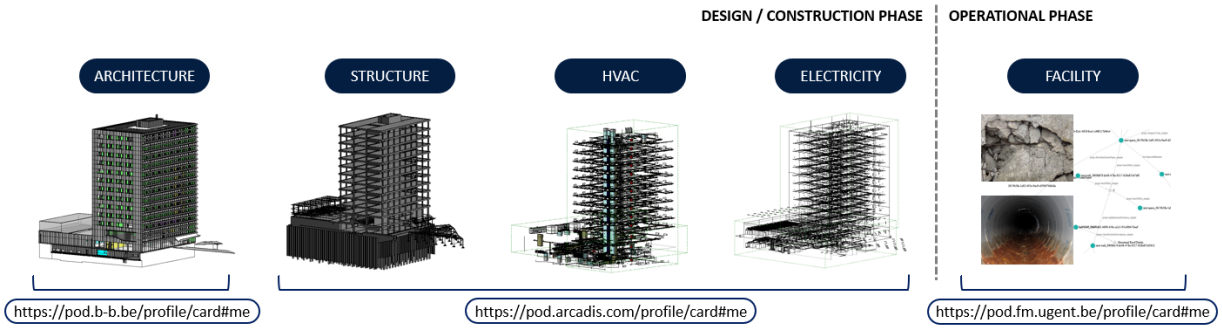


Fig. 7. Use case setup and model IPR

Building Data (LBD) ontologies implemented in the existing IFCtoLBD converter [28]: the Building Topology Ontology (BOT)[29] [29] and the BuildingElement Ontology (BEO)[30].

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`). 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 servers, even when referring to data on other servers in the project.

Upon project participation, a Reference Registry can be created for each of the four stakeholders. When data is uploaded, this registry can be populated with abstract concepts that link corresponding entities in the semantic and geometric models. We use the conversion methodology described in Malcolm et al. [10] to automatically link RDF concepts with their GUID-based counterpart in the 3D model. 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, using the 3D models as interfaces to heterogeneous project data.

### *6.2. Semantic Enrichment for a Damage Case*

This compact demonstration includes the discovery and documentation of damage on an element in the federated model. We consider this an exemplary scenario, as it covers both the use of heterogeneous data sources (RDF, imagery, geometry) and sources from multiple stakeholders are involved in the process. Damage assessment is a frequently occurring activity in the operational phase of a building or heritage object. As an illustration, Monumentenwacht Vlaanderen, an initiative to monitor the state of heritage in Flanders, performs more than 1000 inspections per year, resulting in a multitude of reported damages [30]. However, it is a cumbersome activity which is carried out in a largely manual fashion [31], and, if digitised at all, the resulting reports are most often spreadsheets or form results [30], detached from other digital information about the asset. Indeed, the use of BIM models in this regard is scarce [32, 33], also because BIM models are often not updated or documented to reflect the as-built state [34]. Semantic Web-based documentation of damages, which allows a more interdisciplinary approach, is the idea behind the Damage Topology Ontology (DOT)[31] [31]. For this demonstration, we will use the DOT ontology for semantic documentation, in combination with the ConSolid ecosystem for documentation and linking of heterogeneous sources, such as imagery and 3D geometry.

We will discuss a situation where the Facility Manager localises damages on a regular building element 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

[29] https://w3id.org/bot#
[30] http://pi.pauwel.be/voc/buildingelement#
[31] https://w3id.org/dot#

data structures created by these steps. However, it should be clear that an agent will never initiate these interactions directly but use a dedicated UI instead (e.g., Figure 8).

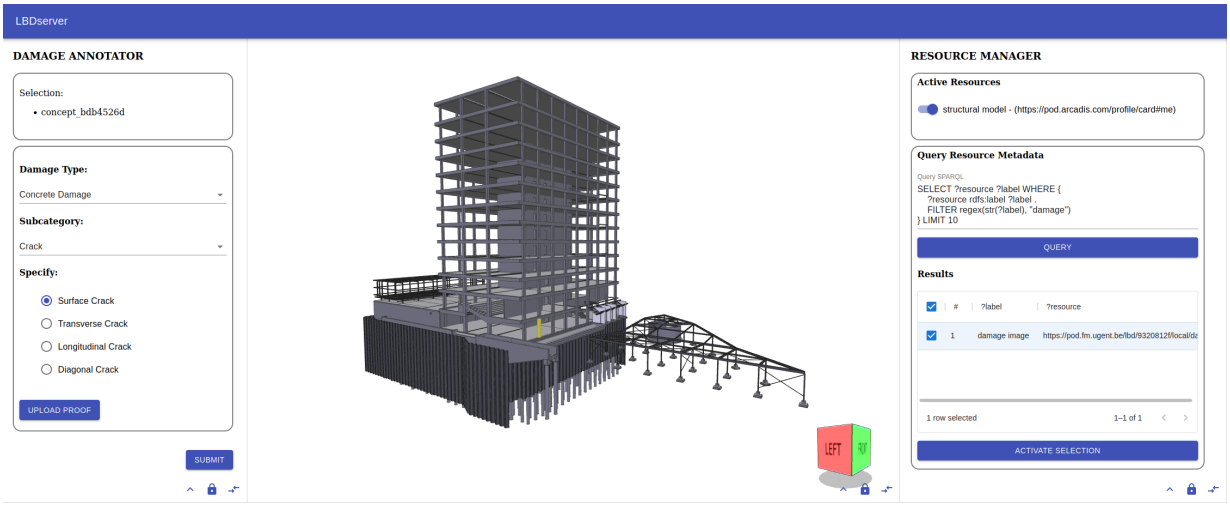


Fig. 8. Example UI to navigate and enrich resources in the ConSolid ecosystem. Included modules in the depicted setup are a 3D Viewer (based on the Xeokit SDK), a Query Module for finding and activating federated project datasets and a Damage Enrichment Module. Since data and applications are fundamentally separated, specialised UI modules can be created and combined depending on the use case.

### *6.2.1. 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, which is to be created as well. During the assessment activity, the publication status (see Section 3.1) is set to 'work-in-progress'. When the assessment is ready, this can be changed to 'shared' using the pattern described in Section 3.4. In our example, we will create the semantics in the distribution using the DOT ontology and one of its extensions, the Concrete Damage Ontology (CDO) [31].

At first, two graph-specific identifiers are created, namely one for the damaged element (`damageDistFM:instance_a003d5d4-c410-42fe-a132-83ff5b69435c`) and one for the damage area (`damageDistFM:instance_8aa5bc1b-7bc5-4555-90d7-81a2bb9e57bc`) (Listing 13). They only exist in this specific document and have not yet been linked to any other existing project information. To be able to reference them in other resources later on, both the damage and the damaged element need to be aggregated by a local 'Reference Aggregator' in the project's Reference Registry on the Pod of the Facility Manager. In this example, this is done with respectively `<#ra_bdb4526d>` and `<#ra_c2427dd9>` (Listing 14). 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 5, these abstract concepts are mapped to the local identifiers created for documenting the damage, as mentioned above.

```
# The RDF resource on the Facility Manager's Pod that documents the damage
@prefix damageDistFM: <https://fm.ugent.be/cac6d088-00b3-4433-8d68-e3d22ade072a#> .

damageDistFM:instance_a003d5d4-c410-42fe-a132-83ff5b69435c
    dot:hasDamageArea damageDistFM:instance_8aa5bc1b-7bc5-4555-90d7-81a2bb9e57bc ;
    a dot:ClassifiedDamage, cdo:SurfaceCrack ;
    cdo:crackWidth "35" .

[...] # further domain-specific damage enrichment triples
```

Listing 13: Triples in the distribution of the Damage dataset, maintained by the FM. The metadata graph will be similar to the one listed in Listing 5.

```
# The Reference Registries of the Facility Manager and the Architecture Office
@prefix refFM: <https://fm.ugent.be/b677ce97-3191-4583-96cb-93c7f895b102#> .
@prefix refArch: <https://pod.b-b.be/f0c8cb37-5a91-43fc-846f-15cb3c28c4d5#> .

# The RDF resource on the Facility Manager's Pod that documents the damage
@prefix damageDistFM: <https://fm.ugent.be/cac6d088-00b3-4433-8d68-e3d22ade072a#> .

# a reference aggregator is created for all references to the (damaged) element
refFM:ra_bdb4526d consolid:aggregates refFM:ref_a80854ae , # local aggregation
        refArch:ra_3de17fbe . # 2nd level aggregation of remote concept, related to the 3D element

# the damaged element as identified in the damage semantics graph
refFM:ref_a80854ae consolid:hasIdentifier refFM:id_c69531c5 .
refFM:id_c69531c5 consolid:inDocument damageDistFM: ;
    dct:conformsTo <http://www.w3.org/1999/02/22-rdf-syntax-ns#> ;
    schema:value "https://fm.ugent.be/cac6d088-00b3-4433-8d68-e3d22ade072a#instance_a003d5d4-c410-42fe-a132-83ff5b69435c"^^xsd:anyURI .

# a concept is created for all references to the damage area
# the newly created reference is the only reference to the damage area so far
refFM:ra_c2427dd9 consolid:aggregates refFM:ref_f1f2704e .
refFM:ref_f1f2704e consolid:hasIdentifier refFM:id_27801276 .

# the damage area as identified in the damage semantics graph
refFM:id_27801276 consolid:inDocument damageDistFM: ;
    dct:conformsTo <http://www.w3.org/1999/02/22-rdf-syntax-ns#> ;
    schema:value "https://fm.ugent.be/cac6d088-00b3-4433-8d68-e3d22ade072a#instance_8aa5bc1b-7bc5-4555-90d7-81a2bb9e57bc"^^xsd:anyURI .
```

Listing 14: Linking the local identifiers in the damage documentation graph to a 'Reference Aggregator', i.e. a neutral concept in the Reference Registry of the stakeholder.

### *6.2.2. Lifting local identifiers to project level*

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. A local concept (`refFM:ra_bdb4526d`) is created for the damaged element, which was selected in a 3D viewer. Therefore, the concept located at Bureau Bouwtechniek's Pod (the owners of the 3D architectural model) is also known. Hence, it can be immediately aggregated as an alias of the new concept on the FM's Reference Registry (Listing 14). A notification must now be sent to Bureau Bouwtechniek to automatically (e.g. via a satellite) or manually aggregate this new concept as well, so a bi-directional, recursive discovery is made possible.

### *6.2.3. Enrichment of sub-document identifiers: pixel region*

The FM office will now further document the damage with an image, and does so by creating a new metadata description for this photograph (`dcat:Dataset`), with the image (image/jpeg) as a distribution (Figure 8). 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 15, which uses the International Image Interoperability Framework (IIIF) specification's image API[32] for pixel identification. 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.

```
# The Reference Registry of the Facility Manager
# The Reference Registries of the Facility Manager and the Architecture Office
@prefix refFM: <https://fm.ugent.be/b677ce97-3191-4583-96cb-93c7f895b102#> .
```

[32] https://iiif.io/api/image/3.0/

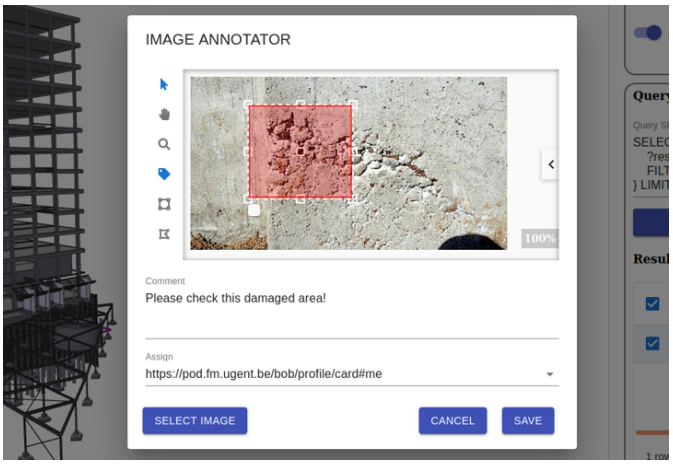


Fig. 9. An image region annotator helps in interacting with the ecosystem, implementing domain-specific views on top of it.

```
@prefix refArch: <https://pod.b-b.be/f0c8cb37-5a91-43fc-846f-15cb3c28c4d5#> .

# The image resource on the Facility Manager's Pod that shows the damage
@prefix pictureDistFM: <https://fm.ugent.be/9f9b1794-3ef7-4d16-87ff-2bb5e2602639> .

# this abstract concept is the same as the one mapped to the damage area,
# effectively linking the image zone with the damage area.
refFM:ra_c2427dd9 consolid:aggregates refFM:ref_ba0fe231 .

refFM:ref_ba0fe231 consolid:hasIdentifier refFM:id_1d2eee0e .
refFM:id_1d2eee0e consolid:inDocument pictureDistFM: ;
    # The identifier notation conforms to the IIIF specification
    dct:conformsTo <https://iiif.io/api/image/3.0/> ;
    schema:value "https://fm.ugent.be/9f9b1794-3ef7-4d16-87ff-2bb5e2602639/pct:20,10,25,55/
    max/0/default"^^xsd:string .
```

Listing 15: Defining an image's pixel zone as a specific representation of an abstract concept, identified by a Reference Aggregator.

Again, this can be easily done with a dedicated user interface: if a 3D element is selected in the viewer, the pixel zone can be immediately linked to the concept related to the 3D element (i.e. the 3D element acts as an interface to select the abstract concept), even if it is originally registered elsewhere.

### *6.3. Validation: Querying the project graph*

Once this mapping is done, a client can query the set of federated project resources 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, and the non-semantic resources (e.g., 3D models, imagery) can be used to visualise the results and access knowledge about an object documented in other resources. In the above described case, a GUI that looks to the project from a perspective of damage management would allow an agent to perform the following steps:

- Authenticate with a Solid WebID.
- Select the project of interest and discover the SPARQL satellites of its aggregated catalogs.
- Query the SPARQL satellites in the project for damage assessments (Listing 16) and find the damaged element. In Listing 16, this corresponds with the variables '?el' ad '?doc'.

– Inject the query results for '?el' and '?doc' in the query templates given in Listing 11. This will identify the concept that aggregates the identifier of the damaged element in this specific resource.
– Inject the query results for '?alias' (query in Listing 11) in the query in Listing 12. The alias is a URL which includes the URL of its Reference Registry and the root of the Pod it resides in, so targeted queries can be executed on the corresponding SPARQL satellites. In the example developed in this section, this will yield all references of this concept and the references aggregated by its remote aliases, including the pixel zone in the image and the identifier in the 3D model.
– Display the found sources and the sub-document references they contain in an appropriate GUI. E.g., if references in a geometric resource are found, display in a 3D viewer. If the damages are only documented through pictures or textually, use an image annotation GUI or a text editor component.

```
SELECT ?el ?doc WHERE {
    GRAPH ?doc {
        {?el dot:hasDamage ?dam .}
        UNION
        {?el dot:hasDamageArea ?dam .}
    }
}
```

Listing 16: SPARQL Query to find the elements that have been damaged. Other references and aliases of this element can then be found by querying the Reference Registries of the project.

These steps are generally applicable for retrieving heterogeneous sub-document identifiers in aggregated projects, based on a combination of domain-specific queries (Listing 16) and ConSolid data patterns (Listings 11 and 12). It is clear that the templates for domain-specific queries must be provided by third-party tools that offer a domain-specific interface to the federated project data (see Section 3.4 for APIs and Section 7 for GUIs).

## 7. Evaluation and Conclusion

In this paper, we proposed an architecture for organising federated datasets for the AEC sector. The Solid infrastructure forms the core of the project, but we extended this infrastructure both in terms of microservice architecture and (meta)data patterns (using DCAT instead of LDP). For the microservices, we discussed the benefits of a SPARQL interface to the Pod, mainly aiding in resource discovery. Based on this SPARQL endpoint, virtual views can be constructed to group similar resources in 'virtual containers' or present Pod data according to a specific industry standard, using domain-specific interfaces. Concerning the (metadata) patterns, we proposed to avoid implicit semantics in resource URLs and only indicate relationships between resources using semantic links in metadata records. Using recursive DCAT catalogs, which we called 'Dataset Aggregators', we have shown that project data hosted by different stakeholders is discoverable with a single triple pattern. This introduces a flexibility to finegrainedly include relevant datasets at the level of individual project partners (e.g., subcontractor datasets, visitor datasets, etc.). In the same move, a scalable approach towards the data integration of multiple projects becomes possible: 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, sub-document identifiers should be kept decentrally as well. We argued that every stakeholder should be able to construct links between identifiers in heterogeneous datasets, independent from other participants in the project. Using recursive Reference Aggregators, clients that have access to a project access point can easily combine those aliases into a set of links, all referring to the same abstract 'thing'. Since both Dataset Aggregators and Reference Aggregators function on metadata level, the content of project resources remains untouched, so compatibility with their original authoring tools remains. 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, using the scenario of damage enrichment.

Considering a Solid Pod as a hybrid knowledge graph instead of as a set of nested LDP folders brings the benefits of flexible aggregation, but also introduces new challenges that can only be partly addressed by the existing Solid specifications. For example, when a resource can be aggregated in multiple catalogs, access control requirements will change, as inheritance would become very chaotic and prone to errors. In this paper, this was solved by allowing each resource to have its own ACL file, although this is not an ideal solution. Future research includes testing the validity and security of referencing ACL rules semantically in metadata files, thus allowing to apply one ACL rule to multiple resources.

Over the course of the text, we described concepts such as virtual views and aggregation patterns as domain-agnostic as possible, using small examples from the AEC industry as illustrations. Future research should make the concepts put forward in this paper more tangible, showing use cases for different tasks during the life cycle of a project. Furthermore, the interaction of domain-specific third party services with the ecosystem remained largely untouched. The layering of domain-specific interfaces to the project(s) should be further investigated, so specialised services can be set up, and GUIs can be configured. As different standards exist for any type of heterogeneous resource, a flexible way for configuring GUIs needs to be set up to match the many combinations that may exist for source documents representing a certain concept. Future research will need to focus on how this can be achieved, e.g. using technologies such as Webpack Module Federation[33] to aggregate UI modules that can interact with specific resource types, in a container UI that is able to interact with the ecosystem.

When flexible aggregations are to be made, e.g. to combine information at larger scale, taking a decentral approach from the beginning offers benefits not only regarding IP and data sovereignty but also regarding scalability and data integration on the Web. With the concepts outlined in this paper, we hope to show how an environment that has data federation as its core principle can facilitate this, both regarding project-specific information, managed on Solid Pods, and contextual information, leveraging the enormous amount of knowledge on the Web.

## 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] M. Singh, E. Fuenmayor, E.P. Hinchy, Y. Qiao, N. Murray and D. Devine, Digital twin: Origin to future, *Applied System Innovation* **4**(2) (2021), 36.

[2] I. Box, The Information Economy: A Study of Five Industries, Technical Report, 2014, Box, Inc..

[3] 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.

[4] J. Werbrouck, P. Pauwels, M. Bonduel, J. Beetz and W. Bekers, Scan-to-graph: Semantic enrichment of existing building geometry, *Automation in Construction* **119** (2020), 103286.

[5] P. Hummel, M. Braun, M. Tretter and P. Dabrock, Data sovereignty: A review, *Big Data & Society* **8**(1) (2021), 2053951720982012.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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] https://webpack.js.org/concepts/module-federation/

[11] J. Werbrouck, P. Pauwels, J. Beetz and E. Mannens, Mapping Federated AEC projects to Industry Standards using dynamic Views, in: *10th Linked Data in Architecture and Construction Workshop*, CEUR-WS. org, 2022.
[12] 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.
[13] 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/.
[14] M. Bonduel, A Framework for a Linked Data-based Heritage BIM, 2021. https://lirias.kuleuven.be/retrieve/618662.
[15] M. Verstraete, S. Verbrugge and D. Colle, Solid: Enabler of decentralized, digital platforms ecosystems, in: *31st ITS European Conference*, 2022, pp. 1–19.
[16] R. Verborgh, Solid: innovation through personal data control, 2021. https://rubenverborgh.github.io/ECA-2021/#title.
[17] M.H. Rasmussen, Solid: innovation through personal data control, 2021. https://rubenverborgh.github.io/ECA-2021/#title.
[18] B. Otto, M.t. Hompel and S. Wrobel, International data spaces, in: *Digital Transformation*, Springer, 2019, pp. 109–128.
[19] P.D.B. Otto, GAIA-X and IDS, Zenodo, 2021. doi:10.5281/zenodo.5675897.
[20] E. Daniel and F. Tschorsch, IPFS and friends: A qualitative comparison of next generation peer-to-peer data networks, *IEEE Communications Surveys & Tutorials* **24**(1) (2022), 31–52.
[21] F. Parrillo and C. Tschudin, Solid over the Interplanetary File System, in: *2021 IFIP Networking Conference (IFIP Networking)*, IEEE, 2021, pp. 1–6.
[22] Organization and digitization of information about buildings and civil engineering works, including building information modelling (BIM) — Information management using building information modelling — Part 1: Concepts and principles, Standard, International Organization for Standardization, 2018.
[23] Information container for linked document delivery , Standard, International Organization for Standardization, 2020.
[24] O. Hartig, An overview on execution strategies for Linked Data queries, *Datenbank-Spektrum* **13**(2) (2013), 89–99.
[25] 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.
[26] C. Eastman, P. Teicholz, R. Sacks and K. Liston, *BIM handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers and Contractors*, 2nd edn, John Wiley & Sons, Hoboken, New York, United States, 2011, p. 650, ISBN 978-0-470-54137-1. ISBN 978-0-470-54137-1.
[27] Industry Foundation Classes (IFC) for data sharing in the construction and facility management industries, Standard, International Organization for Standardization, 2018.
[28] M. Bonduel, J. Oraskari, P. Pauwels, M. Vergauwen and R. Klein, The IFC to Linked Building Data Converter - Current Status, in: *6th Linked Data in Architecture and Construction Workshop (LDAC), CEUR Workshop Proceedings*, Vol. 2159, London, United Kingdom, 2018, pp. 34–43. http://ceur-ws.org/Vol-2159/04paper.pdf.
[29] 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.
[30] B. van Laar, Vernieuwde inspectieverslagen van Monumentenwacht helpen eigenaars om hun gebouwd erfgoed beter te onderhouden en te beheren, in: *Preventieve Conservatie van Klimaat- en Schademonitoring naar een Geïntegreerde Systeembenadering*, Vol. 35, E. Verstrynge, B. van Bommel, N. Vernimme and R. van Hees, eds, WTA-NL-VL, Leuven, Belgium, 2019, pp. 1–9. https://www.wta-international.org/fileadmin/user_upload/Nederland-Vlaanderen/syllabi/2019-04-05_Preventieve_Conservatie.pdf.
[31] 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.
[32] M.K. Seeaed and A. Hamdan, BIMification of stone walls for maintenance management by utilizing Linked Data, in: *31st Forum Bauinformatik*, 2019.
[33] K. Luig, D. Mustedanagic, D. Jansen, S. Fuchs, R. Schülbe, P. Katranuschkov, A.-H. Hamdan, C. Franzen, K. Hiemann and R. Scherer, Towards a Building Information Modeling System for Identification and Retrofit Planning of Stone Damages, in: *Euro-Mediterranean Conference*, Springer, 2020, pp. 254–261.
[34] R. Volk, J. Stengel and F. Schultmann, Building Information Modeling (BIM) for Existing Buildings - Literature Review and Future Needs, *Automation in Construction* **38** (2014), 109—127. doi:10.1016/j.autcon.2013.10.023.

## Appendix A. Prefixes used in this paper

This appendix contains a list of prefixes used throughout this paper, which refer to ontologies. Prefixes related to a Solid Pod or a specific resource on a Solid Pod are included in the individual listings that mention them.

```
@prefix consolid: <https://w3id.org/consolid#>.

@prefix acl: <http://www.w3.org/ns/auth/acl#> .
@prefix beo: <http://pi.pauwel.be/voc/buildingelement#> .
@prefix cdo: <https://w3id.org/cdo#> .
@prefix dcat: <http://www.w3.org/ns/dcat#> .
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix dot: <https://w3id.org/dot#> .
@prefix ex: <http://example.org/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix ldp: <http://www.w3.org/ns/ldp#> .
@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#> .
```

Listing 17: Prefixes used in this paper.