Semantic Web 15 (2024) 429–460
DOI 10.3233/SW-233396
IOS Press

ConSolid: A federated ecosystem for heterogeneous multi-stakeholder projects

Jeroen Werbrouck [a,c,d,*], Pieter Pauwels [a,b], Jakob Beetz [c], Ruben Verborgh [d] 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, Department of Electronics and Information Systems, Ghent University – IMEC, Belgium*

Editors: Bahar Aameri, University of Toronto, Canada; María Poveda-Villalón, Universidad Politécnica de Madrid, Spain; Emilio M. Sanfilippo, ISTC-CNR Laboratory for Applied Ontology, Italy; Walter Terkaj, STIIMA-CNR, Italy
Solicited review: three anonymous reviewers

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 many subdomains, 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 decentralized 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 decentralization, 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-inspired interfaces to a Solid Pod and introduce the concept of metadata-generated 'virtual views', to be generated using an access-controlled SPARQL interface to a Pod. A recursive, scalable way to discover multi-vault aggregations is proposed, along with data patterns for connecting and aligning heterogeneous (RDF and non-RDF) resources across vaults 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, Construction and Operations (AECO) industry.

Keywords: Solid, DCAT, interdisciplinary collaboration, Common Data Environment, semantic enrichment

1. Introduction

1.1. A double patchwork of stakeholders and projects

Interdisciplinary collaborations often require 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

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

for its entire life cycle. 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 – a core aspect of Industry 4.0 [36]. The example industry used in this paper is the Architecture, Engineering, Construction and Operations (AECO) industry, worldwide one of the most decentralized industries [6]. In the AECO 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.

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*, as many purposes require different combinations of interdisciplinary and contextual datasets. A modular approach allows gradually building the project model whenever input from a new discipline is required. In this collaborative context, Semantic Web and Linked Data technologies are often identified as apt technologies to achieve interdisciplinary data mapping, logical inferencing and query-based discovery of federated data [2,32].

At the same time, expressing knowledge using these technologies often increases its complexity and processing time, and sometimes adds little value to the overall data. This applies for example to *non-structured* or *semi-structured* datasets, such as imagery, point clouds, geometry and, sensor data streams [48], where the increase in complexity does not need to a significant increase in value. This need for heterogeneous solutions is acknowledged by both academia and industry, but efficient solutions for dynamic management of such resources are currently lacking.

The fact that the number of disciplines involved is dynamic and depending on the use case also implies that it is very impractical to try to centralize all this information in a single *Common Data Environment (CDE)* [21]. Present-day CDEs are mostly not created with data federation at their core, nor do they use domain-agnostic standards to express knowledge about the asset. This impedes a more mature data integration and reuse practice for the AECO industry and adjacent disciplines aiming to incorporate building data in their workflow – which does not only result in information losses during data handover phases, but also in the creation of parallel, unlinked and unsynchronized duplicates of information. For example, when contextual data is copied into the CDE, or when not all stakeholders during the life cycle of an asset use the same CDE. In turn, this weakens the asset's intended *Single Source of Truth (SSoT)* [50] because the risk for ambiguities and inconsistencies increases. Moreover, uploading everything to a centralized, commercial cloud hub may cause issues regarding intellectual property rights (IPR) and data sovereignty ('how do I remain in control of my data and how can it be used legally') [17]. This holds for both project-specific and external, contextual data (e.g. geospatial, governmental and historical data). When the digital asset model is meant to be used during the operational phase, other information streams will need to be integrated as well, such as user preferences and room reservations. Similar to Open Data [28] practices, added value will rise from combining different datasets (or collections of datasets) in order to answer a particular question. In many scenarios, building data will not even be the core input, but only provide contextual information for other activities. Defining a 'project' as a question-specific collection of datasets supports the above-mentioned claim that it is inherently incomplete, or, in other words, that it can always be further extended. This naturally aligns with the Open World Assumption (OWA) [10] on which the Semantic Web [3] relies. In this light, the aim to centralize 'all' project data on a single platform is simply impossible. Phrasing this differently in context of present-day infrastructures, a 'central CDE' can thus be considered 'federated' from the moment it refers to external contextual information.

To the authors' knowledge, currently there are no federated CDEs based on domain-agnostic open standards. However, the technologies to support such infrastructure are available. In such an environment, Linked Data technologies allow stakeholders to freely choose where their project contributions will be hosted, and control in a fine-grained way who may access certain project information (e.g., other stakeholders, the public, product manufacturers, etc.) [47].

This may be a self-hosted data environment. Upcoming open Web specifications such as WebID-OIDC [37] 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 project [42], where WebIDs are used to control access to resources on a personal data vault. In Solid, such vaults are called ‘Pods’. In the case of multi-stakeholder, multi-resource collaborative projects, this means that a company can authenticate in a standardized way to the vaults of other project participants to access their (filtered) project contributions. Also, online services (i.e., those which have been granted access) 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-vault environments. However, multi-vault environments require a different approach in many cases. In this paper, we propose an architecture for such multi-vault environments. Our proposal will be based on the Solid ecosystem, although several additional boundary conditions will be applied, regarding URL stability, metadata management and sub-document, cross-Pod references of heterogeneous resources.

1.2. Research questions

The above considerations indicate the need for a *federated, multi-purpose, cross-domain ecosystem for heterogeneous digital twin projects*. 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 follows:

RQ1: How to devise a federated, multi-stakeholder ecosystem for interdisciplinary, heterogeneous digital asset models, based on the Solid specifications? Which (meta)data patterns and service architecture will provide a stable and scalable environment that maximizes the potential for discovery of related information, and integration and reuse of existing data?

The ecosystem described in this research question needs to meet the following constraints:

- **Objective 1**: The ecosystem has a federated nature. I.e., resources hosted on different servers on the Web can be part of the same larger project and are discoverable as such.
- **Objective 2**: The ecosystem is domain-agnostic, and therefore allows a multi-disciplinary approach to documenting, aggregating and enriching asset data.
- **Objective 3**: Resource URLs in the ecosystem are not influenced by implicit semantics subject to changes and remain stable over their publication life cycle. As this prohibits container-based discovery of resources, the framework must implement a query-based discovery alternative.
- **Objective 4**: An interdisciplinary ecosystem implies dataset heterogeneity. The ecosystem must not make any assumptions on the data formats of its resources.
- **Objective 5**: 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.
- **Objective 6**: To make the project more than a collection of datasets, the ability to establish sub-document links between non-RDF datasets will be a prerequisite.
- **Objective 7**: 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 which are not originally embedded in the ecosystem.

In devising our approach to answer this research question, we will start from the following hypotheses:

- **Hypothesis 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 knowledge graph [9] instead of as only a collection of documents. This is used to address multi-purpose use of resources and aggregations of datasets living on different Pods.
- **Hypothesis 2**: 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.

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 AECO industry to identify the requirements of the ecosystem, we aim for a solution that is generally applicable. We will describe and address every sub-challenge in a domain-agnostic fashion, although the examples will be related to the AECO industry.

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 [47] to use the Solid infrastructure as a means to create a federated Common Data Environment, and to map the specific 'decentral' nature of the AECO industry to the available technologies offered by the Solid specifications. This research was extended in [51], which describes a methodology for pattern-based access control to AECO 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)[1] standard. An initial proposal for patterns to connect heterogeneous datasets in a federated ecosystem was described in [45], based on data conversion patterns described in [25]. However, the proposed data patterns still featured multiple avoidable complexities. Furthermore, this previous research 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 [46] and will largely contribute in simplifying the early versions of the ecosystem and make it more robust, scalable and domain-agnostic. This paper organizes 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, 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 we need to make to the current Solid infrastructure in order to achieve the goals of the envisaged ecosystem, regarding metadata patterns, interfaces and microservices. Section 4 then builds upon those extensions to propose aggregation structures for single- and multi-vault configurations, and Section 5 devises the patterns for asynchronous sub-document linking. A minimal proof-of-concept from the AECO industry is then described in Section 6. We evaluate the ecosystem against the original objectives in Section 7. Furthermore, the ecosystem will be evaluated against the FAIR principles (Findable, Accessible, Interoperable and Reusable) [52] for scientific data management and stewardship, as a measurement for its general applicability. Although FAIR data management is usually related to *open* datasets, we note significant overlaps with *access-restricted* multi-disciplinary environments as required by the AECO industry, and, more generally, to the built environment, due to the cross-domain usage of project-specific and contextual datasets. The paper concludes with a discussion and overview of future work (Section 8). In the remainder of the paper, the ecosystem will be called ConSolid. Definitions related to the data patterns used in the ecosystem are published as the ConSolid vocabulary.[2] The different software prototypes discussed in this paper are aggregated on a public Github repository.[3]

[1]SHACL: https://www.w3.org/TR/shacl/, accessed 12/01/2023.
[2]ConSolid namespace: https://w3id.org/consolid#, accessed 12/01/2023.
[3]ConSolid proof-of-concept: https://github.com/ConSolidProject/infrastructure, accessed 27/01/2023.

2. Related work

This section will cover key technologies for data aggregation and containerization, as well as the fundamentals of the Solid ecosystem. Finally, we review some related initiatives for Web decentralization.

2.1. Containerization of federated, heterogeneous datasets

In a digital construction project, many resources may actually refer to one and the same object. A window may be semantically described in graphs produced by different stakeholders in the project, and be visualized 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' [13] or, according to ISO 19650, an 'information model', i.e. *a set of structured and unstructured information containers* [21]. An 'information container' is then a '*named persistent set of information, retrievable from within a file, system or application storage hierarchy*' [21]. For the remainder of the text, we will use the term 'multi-model'. The term 'federated multi-model' is related to project data residing on multiple servers, connected in a larger catalog using Linked Data technologies. A multi-model can be organized 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 in context of this paper: the Linked Data Platform (LDP),[4] and the Data Catalog Vocabulary (DCAT)[5] 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* [1]. 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.). 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 AECO project resources have been proposed in [4], such as the Construction Dataset Context (CDC) ontology.[6]

2.2. Web decentralization ecosystems

Over the last years, the concept of 'data sovereignty' of actors in the digital economy has known growing interest [18]. Data sovereignty is closely related to data security and protection. It allows to *self-determine how, when and at what price others may use [your data] across the value chain* [20]. One of the core ideas is that the owner of the data remains in control [43]. Hence, organizations cannot use your data for whichever purpose that suits their interests. The majority of web services still rely on centralized data storage, which means that it is still up to the service providers to comply to existing regulations for data protection (such as the EU's

[4]LDP: https://www.w3.org/TR/ldp/, accessed 27/01/2023.
[5]DCAT: https://w3c.github.io/dxwg/dcat/, accessed 27/01/2023.
[6]CDC: https://mathib.github.io/cdc-ontology/, accessed 27/01/2023.

General Data Protection Regulation (GDPR) [7]). Numerous recent initiatives therefore focus on decentralized 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 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, etc.) from random access, while only allowing authenticated users through, for example, tokens (2-way handshake). Different is the Solid project for Web decentralization [42], which adds a decentralized authentication layer on top of a heterogeneous resource server, with an LDP-inspired protocol with containers and documents as the primary interface. The authentication layer is based upon the concept of a WebID; a 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).

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[7] (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 everyone to set up *a personal identity provider* and authenticate to any Solid-enabled service. Hence, identity verification may happen without third-party companies acting as a middle-man.

Access control rules in Solid 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 (WAC) [38] 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. Alternative data patterns are provided by the Access Control Policy[8] (ACP) specification, which will allow more semantic freedom to express access control rules.

The general use of Solid as an enabler of decentralized digital ecosystems is described in [43], from a techno-economic perspective. Arguments to extend Solid Pods beyond documents and containers, and consider it a 'hybrid knowledge graph' exposed via multiple interfaces are given in [9]. Furthermore, in earlier work (Section 1.3), we have identified the Solid specifications as a suitable basis for the needs of a digital AECO industry. This has multiple reasons:

```
<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 the Appendix

[7]OIDC: https://openid.net/connect/, accessed 27/01/2023.

[8]ACP: https://solidproject.org/TR/acp, accessed 27/01/2023.

```
# 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 container and its resources

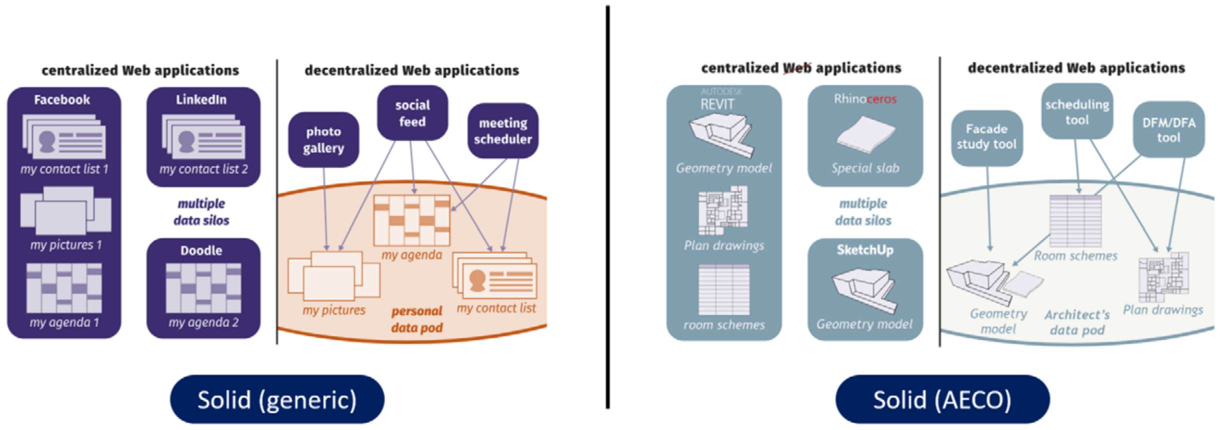


Fig. 1. Web applications in the Solid ecosystem [42] and applied to the AECO industry [33].

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 Information' (SSoI) principle by design, as CDE services do not locally duplicate information, but services can discover the right data for their use case and create cross-server references.

A comparison between the original goals of Solid and an interpretation for the AECO industry is given in Fig. 1. The Solid specifications and their implementation in the Community Solid Server[9] will provide the main infrastructure for the ecosystem in this paper.

In the remainder of the paper, we will use the term 'Solid Pods' when referring to the Solid specific interpretation of data vaults. When speaking about the concept in general, the term 'data vault' will be used.

2.2.2. *International data spaces*

The International Data Spaces (IDS)[10] initiative is a recent, multi-disciplinary initiative oriented towards data sovereignty in the digital economy [30]. 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

[9]Community Solid Server: https://github.com/CommunitySolidServer/CommunitySolidServer, accessed 27/01/2023.
[10]IDS: https://internationaldataspaces.org/, accessed 27/01/2023.

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[11] (2019), which may complement the IDS architecture with technologies for data storage, data compliance and cloud infrastructure [29]. 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 made in [43].

2.2.3. Interplanetary file system

The Interplanetary File System (IPFS)[12] is a peer-to-peer protocol for resilient, distributed storage of hypermedia [8]. 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 [31].

3. Interfaces to a discovery-oriented data vault

3.1. Query-based resource discovery

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

While this container-based interface can essentially be seen as just one of the many possible APIs on top of the Pod [9], its current application hard-wires ‘implicit’ (i.e. non RDF-based) semantics in the URLs of resources and imposes a tree-like folder structure. In this folder structure, 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.

We consider the following example, related to the stages of publication (Work-in-progress, Shared, Published, Archived) as defined in ISO 19650 [21], the authoritative standard in the AECO 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 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 Solid Protocol specification, as it allows to interact with resources via HTTP – there are just no slash semantics beyond the root of the Pod, which remains essentially a (very large) `ldp:Container` (Fig. 2).

[11]GAIA-X: https://gaia-x.eu/, accessed 27/01/2023.
[12]IPFS: https://ipfs.io/, accessed 27/01/2023.

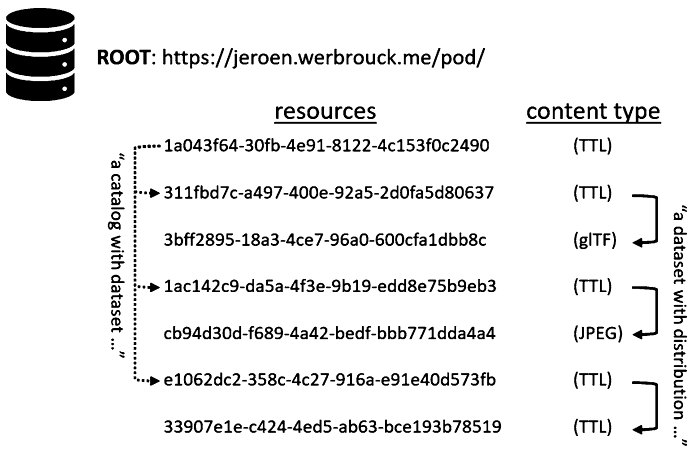

ROOT: https://jeroen.werbrouck.me/pod/
resources
content type
1a043f64-30fb-4e91-8122-4c153f0c2490 (TTL)
311fbd7c-a497-400e-92a5-2d0fa5d80637 (TTL)
3bff2895-18a3-4ce7-96a0-600cfa1dbb8c (glTF)
1ac142c9-da5a-4f3e-9b19-edd8e75b9eb3 (TTL)
cb94d30d-f689-4a42-bedf-bbb771dda4a4 (JPEG)
e1062dc2-358c-4c27-916a-e91e40d573fb (TTL)
33907e1e-c424-4ed5-ab63-bce193b78519 (TTL)
"a catalog with dataset ..."
"a dataset with distribution ..."

Fig. 2. A flat list of resources in a data vault. Containment triples and metadata are mentioned using RDF instead of slash semantics, allowing resources to be grouped flexibly.

The 'meaning' that allows semantic containerization and filtering on higher-level (domain-specific) layers must now come from an explicit metadata record that 'enriches' the heterogeneous resource using RDF ontologies, instead of from the parent container. Note that the form of a metadata record as a specific type of document is still very much under discussion in the Solid ecosystem. However, explicit metadata records can be handled exactly the same as other resources on a vault: it is a regular RDF resource which happens to advertise itself as being metadata of another resource, hence allowing query-based discovery of this other resource.

One advantage of storing semantically rich metadata on a resource in a metadata record, is that this record 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 [46]. 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.

3.2. *A union of metadata records for heterogeneous datasets*

File-based resources on the Web are inherently heterogeneous. In industries such as the AECO industry, a myriad 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 project resource needs at least one RDF-based metadata record to allow query-based discovery of the resource, and inclusion in query-based views on a data vault. 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. We identify the need for a queryable, access-controlled union of all metadata records on a data vault, to be able to construct these virtual views, which are essentially just filters on vault data. Optionally, this union includes RDF-based project resources as well. The infrastructure for such union of resources on a Pod will be described in Section 3.3.

As a vocabulary to structure this metadata network, we will opt for the DCAT vocabulary (Section 2.1), contrasting with Solid's default usage of LDP containers. This is because the DCAT vocabulary allows to:

1. address 'containerization' and expressive metadata descriptions with a single, integrated vocabulary;
2. semantically decouple metadata records and actual resources, which allows RDF-based discovery of heterogeneous datasets and semantic indication of a distribution's versions and content-type;

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

Listing 3. Catalog and metadata record of a resource, using the DCAT vocabulary. Both the catalog and the dataset are accessible as an HTTP resource and as a named graph in the SPARQL satellite. An example vocabulary is used to denote the publication statuses

```
PREFIX pod: <https://pod.b-b.be/>

DELETE {pod:1e19ed7c-2809-408b-a619-f27e601d7423 ex:publicationStatus ex:workInProgress }
INSERT {pod:1e19ed7c-2809-408b-a619-f27e601d7423 ex:publicationStatus ex:shared }
WHERE {pod:1e19ed7c-2809-408b-a619-f27e601d7423 ex:publicationStatus ex:workInProgress }
```

Listing 4. SPARQL INSERT/DELETE query for updating the ISO 19650 publication status of a resource in the ecosystem. The definitions for publication statuses are placeholders

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

  BIND(UUID() as ?virtualContainer)
}

# yields (Turtle):
# <urn:uuid:6f8dd731-4a95-4abd-95fc-23e7d6385ab3> ldp:contains pod:90329a28-6663-44c0-8e40-
    deb50a9e24b1 .
```

Listing 5. SPARQL query to filter project datasets that are 'shared' and return their distributions as virtual containers, to be used by LDP-compatible tooling

3. let metadata records semantically indicate versions of distributions, which is currently not an option in Solid;
4. avoid conflicts with Solid's default usage of containers.

Furthermore, DCAT is the preferred metadata vocabulary for data points compliant to the FAIR principles [27]. Listing 3 gives an example of an RDF-based metadata record in the ecosystem, aggregated in a larger catalog, using DCAT.

The example of changing a resource's publication status (ISO 19650) while maintaining its URL (Section 3.1) now boils down to the insertion of a SPARQL INSERT/DELETE query to this resource. The query mentioned in Listing 4 will allow the resource to be included in a new virtual container for shared resources, and excluded from the one for resources tagged 'work in progress' (Listing 5).

The union of metadata records is to be used as the primary way to query a vault, but it also allows other (higher-level) interfaces to be established. They can be used as a generic way to present a (set of) resource(s) on the vault via industry-specific APIs. In the AECO 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 vault or of the entire federated project. Such APIs create virtual views on the knowledge graph of the project (similar to the one in Listing 5), 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-organized to fit particular industry standards, as is described in [46]. For example, Listing 5 can be part of an API-based, domain-specific interface that organizes the project conforming to the ISO 19650 specification and its defined stages of publication. Other examples, such as a mapping to the Information Container for Linked Document Delivery (ICDD) (ISO 21597) [19] or the BIM Collaboration Format (BCF) API[13] are documented in [49].

3.3. A SPARQL interface to the data vault

We consider a (private) SPARQL endpoint an apt interface to query the union of knowledge graphs on a data vault. Such interface to the Pod Union is already implemented by the Enterprise Solid Server (ESS, Inrupt).[14] However, it is permissioned such that only the Pod owner can access it – external agents are denied access to this functionality.

As indicated in Section 2.2.1, access control on a Solid Pod currently works on the level of (RDF and non-RDF) documents, by means of ACL resources that mention which actors are granted which access rights to which 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 this framework, the hierarchical LDP folder structure will be omitted in favour of a flattened list of resources (Section 3.1). ACL inheritance is no option here, as a resource can be ‘contained’ in multiple (virtual) parent containers, which may impose conflicting ACL rules to their child resources or subcontainers. To maintain compatibility with the Solid specifications it is necessary that either (1) a single ACL resource governs all effective resources on the Pod, or (2) that every effective resource has its own ACL resource. The first option is immediately ruled out, as this would contradict the goal of having a fine-grained access control mechanism. Therefore, the second option is chosen. Although hierarchy-based ACL structures have the advantage of access control inheritance, having an individual ACL graph per document has the advantage of direct mapping. When not only metadata and project resources are mirrored to a SPARQL endpoint, but also their ACL resources, the union of these ACL graphs can be easily checked to construct the set of resources a visitor is allowed to interact with: a combination of explicit authorizations and the authorizations granted to the public (mostly `acl:Read`). A query to retrieve this union from the SPARQL endpoint is given in Listing 6. This query needs to be executed by an agent with full access to the endpoint, i.e. the Pod owner or a delegate (service).

Multiple options are now possible to include only the resulting set of allowed resources – which option to choose will depend on the purpose of the query. The first and most general option is the creation of a permissioned union graph through the injection of the allowed resources via a‘FROM <{resource}>’ statement. A second option is querying the vault through injection of the allowed resources via a ‘FROM NAMED <{resource}>’ – this is similar to the first option, but triple patterns will need to be enclosed by a graph variable: if the enclosed triple patterns are not present in the same named graph, the result set will be empty. An example query modification comparing both options is given in Listing 7. More information on the differences between ‘FROM’ and ‘FROM NAMED’ in SPARQL queries can be found in [23].

The first option is the most flexible one to query the Pod as a whole, but, as we will demonstrate in Section 3.4, it is also the most expensive one regarding query execution time – especially in fragmented data vaults containing large amounts of resources. Execution time for the second option lies much closer to the original query time and may, for example, be used to query metadata patterns which are known to occur within the same graph. This maps well with the purpose of the SPARQL endpoint in the ConSolid ecosystem, where it will be primarily used for query-based discovery of relevant project datasets. Following this discovery, targeted queries can be executed on

[13]BCF-API: https://github.com/BuildingSMART/BCF-API, accessed 27/01/2023.

[14]ESS: https://www.inrupt.com/products/enterprise-solid-server, accessed 23/02/2023.

```
SELECT DISTINCT ?resource
WHERE {
 ?acl a acl:Authorization ;
    # SELECT queries correspond with an acl:Read permission
    acl:mode <http://www.w3.org/ns/auth/acl#Read> .

  # the resources that will be injected in the eventual query
  # the query may also introduce additional requirements for the ?resource variable, in order
     to decrease the amount of resources that need to be included in the selection set,
    resulting in a smaller query execution time.
  {?acl acl:accessTo ?resource } UNION {?acl acl:default ?resource }

  # the actor to check access rights for
  {?acl acl:agent <https://pod.werbrouck.me/b-b/profile/card#me> }
  UNION
  # publicly accessible resources are to be included as well
  {?acl acl:agentClass <http://xmlns.com/foaf/0.1/Agent> }
}
```

Listing 6. SPARQL query to retrieve the resources a given agent is able to query. Public resources should be included as well. This query assumes that every resource on the Pod has its own ACL. In the scenario explained in Section 3.1, in which all resources are present in a flattened LDP folder, this is the case. Richer descriptions of the ?resource variable are possible, to yield a smaller but more detailed result set

smaller union graphs of project-specific (i.e. non-metadata) resources, making the queries based on 'FROM' queries more feasible.

Considering that a typical SPARQL endpoint does not implement such access control and query adaptation functionality, currently all requests to the SPARQL endpoint need to pass via a proxy service which has been granted full read permission, acting on behalf of the Pod owner. This service determines the WebID of the visitor, checks for which resources they have access rights, and injects these in the query, which is then executed. Optional arguments can be passed to instruct which of the query options (FROM or FROM NAMED) should be used. In the remainder of this text, we will consider the proxy service and database service as one, calling it a 'satellite'. The SPARQL satellite should be easily discoverable, which can be done by registering the triple pattern in Listing 8 in the WebID of the owner of the Pod.

The above-described approach can be considered a hybrid solution between the current document-oriented focus of a Solid Pod, set by the Solid specifications, and the Pod as a 'hybrid knowledge graph' as envisioned in [9], which considers documents and their corresponding ACL resources as just one view on the data in a Pod. At the moment of writing, the latter is hypothetical, as there are no implementations yet of this approach.

3.4. *Implementation and considerations regarding the SPARQL satellite*

As a prototypical implementation of the SPARQL satellite, we modified the Community Solid Server codebase to forward any RDF-based information to a SPARQL store, next to offering HTTP access to all resources via a root LDP container in the Pod.[15] As a SPARQL store, Apache Jena Fuseki[16] is chosen. The option `tdb:unionDefaultGraph` is set to true, which allows to query the Pod as the union of its resources. The satellite is implemented as a NodeJS (ExpressJS) server.[17] WebIDs are retrieved from authenticated requests (WebID-OIDC and OAuth 2.0 [15]), after which the query in Listing 6 is executed and its results injected in the original query, before sending it to the SPARQL store.

Performance optimization of this setup is out of the scope of this paper. However, it is important to get at least a notion of the order of magnitude of query execution time with the above-mentioned method for access control verification. As a quick verification of this setup, a Pod was populated with 10 718 851 triples, spread over 1288 graphs, including ACL resources and metadata descriptions (see Section 4). For each of these resources, the owner

[15]Solid Community Server + SPARQL store: https://github.com/LBD-Hackers/SolidCommunity_Fuseki/tree/paper, accessed 27/01/2023.
[16]Apache Jena Fuseki: https://dlcdn.apache.org/jena/source/jena-4.5.0-source-release.zip, accessed 27/01/2023.
[17]SPARQL satellite prototype: https://github.com/ConSolidProject/auth-satellite, accessed 23/02/2023.

```
# The original SPARQL query
SELECT ?md ?durl ?mt
WHERE {
    ?md a dcat:Dataset ;
        dcat:distribution ?dist .
    ?dist dcat:downloadURL ?durl ;
       dcat:mediaType ?mt .
}

# The modified SPARQL query (option 1: FROM)
# This option allows to reconstruct a "permitted union graph" of the Pod, but comes at the
    expense of query execution time
SELECT ?element ?dam ?cause
FROM <https://pod.architect.com/0e12ae3b-744a-4590-aaf1-6daae827dfbd>
...
FROM <https://pod.architect.com/2cb2e8a2-e3b0-4e11-a15e-5626dbbc89a1>
WHERE {
    ?md a dcat:Dataset ;
        dcat:distribution ?dist .
    ?dist dcat:downloadURL ?durl ;
       dcat:mediaType ?mt .
}

# The modified SPARQL query (option 2: FROM NAMED)
# In this option, triple patterns must be enclosed in a named graph to query, which means all
     wrapped triples must occur in this graph. Multiple graph enclosures can occur within one
     query.
SELECT ?element ?dam ?cause
FROM NAMED <https://pod.architect.com/0e12ae3b-744a-4590-aaf1-6daae827dfbd>
...
FROM NAMED <https://pod.architect.com/2cb2e8a2-e3b0-4e11-a15e-5626dbbc89a1>
WHERE {
    GRAPH ?g {
        ?md a dcat:Dataset ;
            dcat:distribution ?dist .
        ?dist dcat:downloadURL ?durl ;
            dcat:mediaType ?mt .
    }
}
```

Listing 7. The SPARQL satellite dynamically updates any query to only include those named graphs to which the visitor has read access

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

Listing 8. Triple pattern to find the SPARQL satellite to a Pod, via the WebID of the Pod's owner

was given full access rights, resulting in 644 permitted resources (as every resource has an ACL attached). Two other accounts were created and were assigned read permissions to a random subset of resources on the main Pod, respectively resulting in 242 and 208 permitted resources. Execution of the original (metadata) query in Listing 7 yields the results listed in Table 1, using a machine for which the specifications are listed in Table 2. For each modification, Table 1 includes the query execution time directly on the SPARQL endpoint, the execution time from the perspective of the client (i.e. including the creation of the set of allowed resources (Listing 6)). It is technically possible to create a bypass for the Pod owner, which would allow to query the full union graph without restrictions (cf. the Enterprise Solid Server by Inrupt). The equivalent is execution of the original query on the default graph (column 'Default' in Table 1).

In this minimal example, we observe that the execution time of 'FROM NAMED </resource/>' queries directly on the SPARQL endpoint lies generally in the same order of magnitude as executing the query without access control

Table 1
Query execution time for the query in Listing 4

	#permitted resources	ACL query (Listing 3) (ms)	FROM		FROM NAMED		Default
			Endpoint[*]	Client[*]	Endpoint[*]	Client[*]	
Owner	644	156.6	8979	9617	65	620	13
Visitor 1	242	143.9	1163	1851	23	517	
Visitor 2	208	143.6	827	1270	21	755	

[*]Query execution time directly on the SPARQL endpoint.
[**]Total query execution time for "client > satellite > SPARQL store (ACL) > satellite > SPARQL store (query) > satellite > client"

Table 2
Specifications of the machine hosting the satellite implementation and the Fuseki SPARQL store

OS	Linux – Ubuntu 20.04.3 LTS
CPU model name	Intel(R) Xeon(R) Platinum 8268 CPU @ 2.90 GHz
CPU count	4
Memory (RAM)	16 GB

on the default graph. In the case of the queries injected with 'FROM </resource/>', query execution time increases significantly, demonstrating its limited applicability, notwithstanding its potential for generating a 'permitted union graph'. As mentioned before, one of the main purposes of the SPARQL endpoint is discovery of relevant datasets on the Pod, i.e. metadata queries where the triple patterns will indeed reside in a single named metadata graph. Therefore, within the boundaries of the ConSolid ecosystem, we can in most cases rely on the first case. The results of this discovery can then be used in a targeted follow-up query on project data. As execution time will decrease with smaller amount of allowed graphs, application of 'FROM </resource/>' will still have its use cases. Note that reduction of the set of queried resources can already be integrated in the access control query (Listing 6), which can be refined with more precise queries on which graphs to include, i.e. by imposing additional metadata queries on the '?resource' variable (e.g., regarding publication status, publication date or even the ontologies that are used). Lastly, from the perspective of the client, performance would increase when all functionality would be included directly in the SPARQL endpoint, allowing to reduce the amount of requests from 6 (client > satellite > SPARQL store (ACL) > satellite > SPARQL store > satellite > client) to 2 (client > SPARQL store > client).

We can generally conclude that the permissioned SPARQL satellite is feasible for query-based discovery of project datasets, using modified queries ('FROM NAMED <{resource}>'). Execution time for querying the entire project Pod as a permitted union graph ('FROM'), however, may only be acceptable for some use cases. However, when the list of targeted resources is limited, it is still a possibility.

4. Discovery and aggregation of project datasets

In Section 3, we set the basic infrastructure for a Solid 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 via a SPARQL satellite. However, these resources were not yet aggregated into a larger whole, a 'multi-model' in AECO terms [13]. 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-vault aggregations. When deemed necessary for the ecosystem, we introduce specific RDF classes and properties, which are published in the ConSolid vocabulary.[18]

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 [46]),

[18]https://w3id.org/consolid#

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

# 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 3
        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

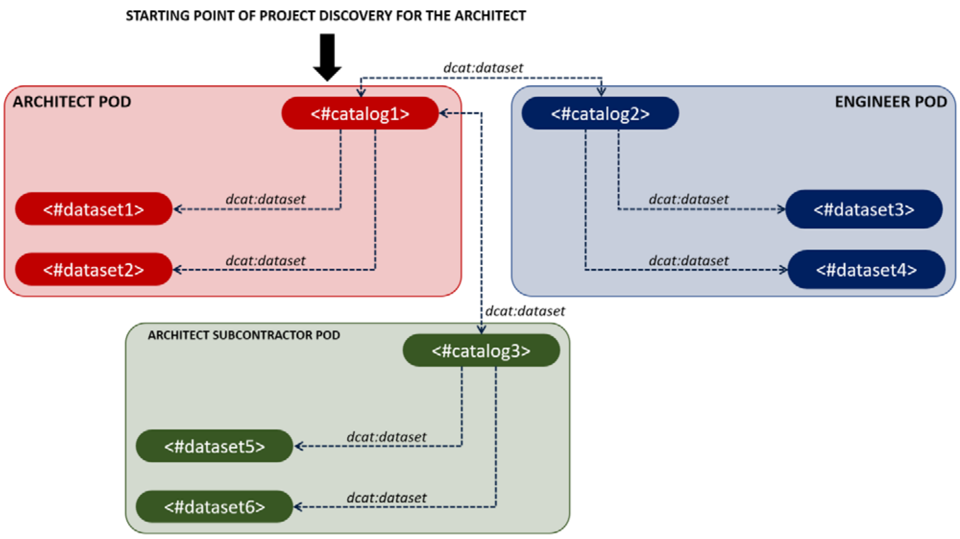


Fig. 3. Starting from one Dataset Aggregator in a specific data vault, it is possible to recursively find all datasets in the federated project. Because bi-directional links may be present, looping should be avoided in the query process.

or manually curated. Dataset Aggregators will form the core to maintain federated multi-models in a scalable way in the ConSolid ecosystem.

The initial level of aggregation occurs when an actor bundles resources on a single data vault, for example, when grouping all resources on the vault 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, vault-specific project definition which follows the DCAT specification exactly.

All stakeholders can thus create their own local aggregator for a collaborative project, independently from the others, forming the main 'access point' on the vault to find project data. Aggregating local datasets, this can perfectly be called a multi-model of its own, but by referencing the catalogs of other stakeholders (on their vaults) (using `dcat:dataset`), it becomes straightforward for a client to discover and query also the contributions of these other stakeholders. This situation is illustrated in Fig. 3. We will use the term 'partial project' to refer to the collection of project data on a single vault. The total set of discoverable project data then consists of the union of all partial projects that are discoverable via the used access point.

```
SELECT ?dataset
WHERE {
    <https://example.org/schools-in-flanders/catalog> dcat:dataset+ ?dataset .
}
```

Listing 10. Recursive query to discover the metadata records of all datasets in an aggregator. Using this query, engines that use link traversal can find all federated datasets in a project. In this example, the catalog URL refers to a Web resource that describes a multi-asset aggregator for ‘schools in Flanders’. Each of those projects will be a set of nested catalogs on its own, grouping contributions from various people

Higher-level aggregations will work in exactly the same way, and allow for a scalable, discovery-based approach of nested DCAT catalogs, which define a project’s boundaries. This means that those boundaries will be very flexible: a ‘project’ can then range from single-building level over neighbourhoods or a geographically scattered group of buildings from the same typology (‘all public schools in Flanders’, ‘all bridges in Germany’). 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, which is 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 manner, 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 [16], such as the Comunica [39] Link Traversal Engine.[19] Higher-level aggregations will make use of this recursive pattern to scale up the level of aggregation without increasing the complexity of the queries. Because bi-directional links will be present between access points of different stakeholders, a mechanism that prevents infinite loops should be part of the querying algorithm.

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 the human employees are the ones who divide the workload and do the eventual work. An office might decide to have a central office data vault, where all employees contribute, but it might also be the case that every employee maintains their own vault, or alternatively that the office creates a dedicated vault 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 vaults (Fig. 3).

Similarly, an office might appoint a subcontractor for specific tasks in the project. These subcontractors will not work on the stakeholder’s main data vault, but will have their own vaults to store their contributions. Often, a project counts a few ‘main contractors’ (e.g. an architect, structural engineer, HVAC engineer, owner). From an organizational 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. *Integrating project-external information*

A resource belongs to a ConSolid project from the moment it is described in a dataset that is, in turn, aggregated in a dataset aggregator. In this paper, we primarily consider resources hosted on the same Pod as the dataset’s distribution. However, a dataset indicates its distributions via their URL – which means these can be located anywhere on the Web. Hence, when project-external resources (e.g. geospatial or governmental) should become part of the overall multi-model, it suffices to create a DCAT metadata record (dataset) that has this external dataset or database as a distribution. Client applications can now discover this dataset and present it to the end user as part of the project.

[19]Comunica Feature Link Traversal: https://github.com/comunica/comunica-feature-link-traversal, accessed 27/01/2023.

4.4. Implementation

A prototypical API to interact with the basic setup described in Section 3.4 using the above mentioned patterns for storage and discovery of data, was created in context of this paper. The API is available on Github[20] and NPM[21] 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. However, other mechanisms are needed to align and reference information from different, potentially federated datasets on a sub-document level. Sub-document identifiers of project resources will in many cases not be natively expressed using RDF. For example, a 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 defined in the image resource. In turn, 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. This principle is illustrated in Fig. 4.

Information creation and information alignment can happen asynchronously amongst stakeholders. For example, indicating that a pixel zone on a picture identifies the same element modeled 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

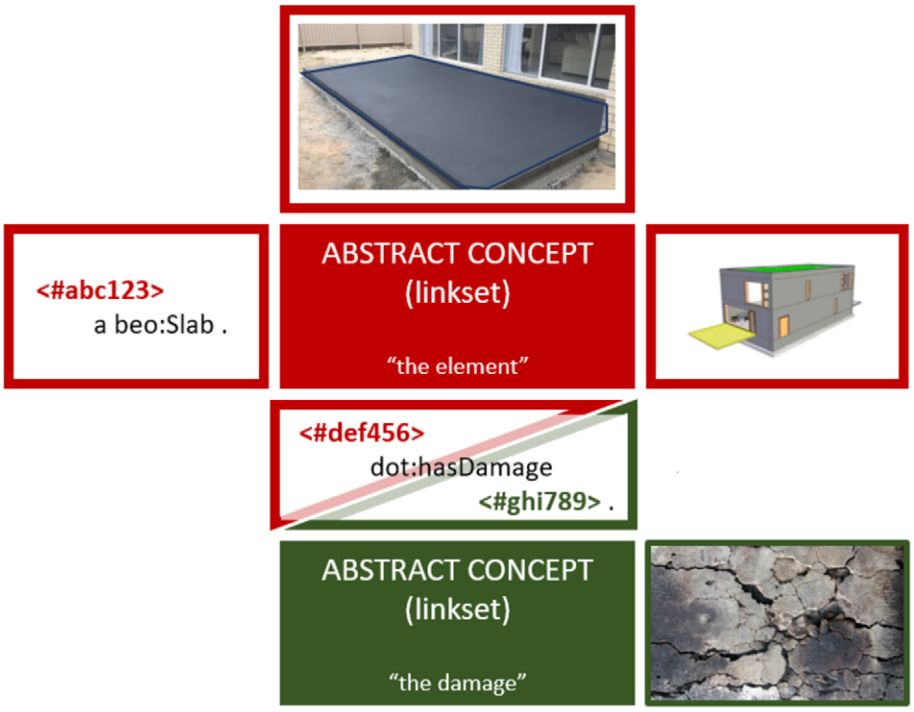


Fig. 4. Document-specific representations of two concepts. Any relationships between those concepts will also be expressed via representations, i.e., the concepts themselves will not be enriched directly with project data.

[20]Daapi (Github): https://github.com/LBD-Hackers/daapi/tree/paper, accessed 27/01/2023.
[21]Daapi (NPM): https://www.npmjs.com/package/consolid-daapi, accessed 27/01/2023.

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 project, when someone (person or digital service) recognizes that the 3D object and the image actually represent the same element. Also, a scalable, federated 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 by other actors.

Lastly, 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, one may want 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 (`consolid:ReferenceAggregator`) 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. 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 Reference Aggregator, 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) [19]:

– A Reference Aggregator (`consolid:ReferenceAggregator`) 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 Reference Aggregator can also recursively aggregate other Reference Aggregators, to allow discovery of alias aggregators of a 'thing' on other data vaults, using a query similar to the one in Listing 10, but using `consolid:aggregates` instead of `dcat:dataset`.
– 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 visualized in Fig. 5. These patterns are demonstrated with a real-world example in Section 6 (Listings 15 and 16).

Different Reference Aggregators can be aggregated themselves in a single named graph on the vault, 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 Registries in a specific project.

Higher-level identifiers (`consolid:Identifier`) in Reference Aggregators even the field between RDF and non-RDF resources, because they indicate a value which is generally only valid in a certain document. In other words, identification of sub-document objects and their allocation is 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

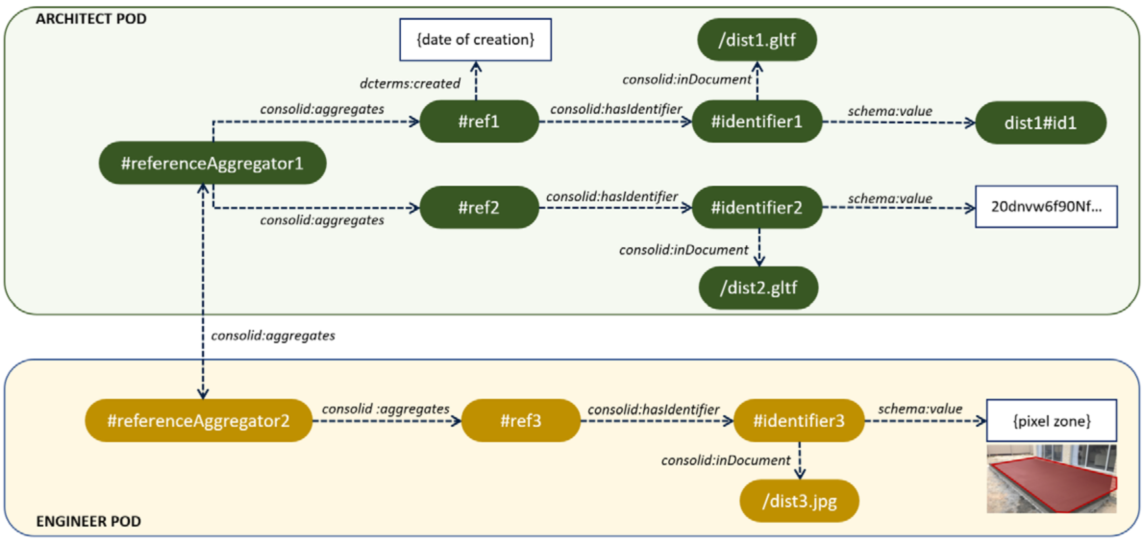


Fig. 5. RDF patterns from Reference Aggregator to sub-document identifier.

Registry. This is a core aspect of the asynchronous enrichment, allowed by the combination of the metadata-level Reference Aggregators and their 'lower level' occurrences in actual resources – since registering references or removing them does not impact the documents themselves, and aliases can be created and 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 Fig. 5 (and Listing 15), the recursive property `consolid:aggregates` allows a Reference Aggregator to aggregate equivalent Reference Aggregators on other stakeholder vaults, i.e. its aliases. In the example, a bi-directional aggregation exists between the Reference Aggregator in the architecture firm's vault and the one on the vault 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 basic queries given in Listings 11, 12 and 13 allow to find the references aggregated by the same Reference Aggregator. The first query (Listing 11) is to be executed on the project Reference Registry in the vault where the *metadata* of the active document is registered – to base upon the location of the metadata instead of on the location of the actual document allows retrieval of references to project-external resources on the Web (e.g., regarding contextual data). This query yields the Reference Aggregator identifier and its aliases, which can now be used to find other representations of this concept in the vaults of the consortium members. This is illustrated in the second query (Listing 12), which is executed on the same vault as the query in Listing 11 – the retrieved *local references* (?local) are further resolved to find other documents (?doc), identifiers (?value) and metadata (?meta) of other local references. Finally, in the third query (Listing 13), the retrieved aliases are used to find remote references (including documents, identifiers and metadata), thus executed on the respective vaults where these aliases reside, which can be easily derived from the alias' URL.

Additional metadata filters may be applied to only retrieve specific references. This can be done by attaching a metadata triple pattern to the query, commented out in Listings 12 and 13.

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

```
SELECT ?concept ?local ?alias
WHERE {
    ?concept consolid:aggregates/consolid:hasIdentifier ?id  .
    ?id consolid:inDocument <{document}> ; # the document is known
        schema:value "{identifier}" . # the identifier is known

    ?concept consolid:aggregates ?local .
    # local references include the URL of the queried data vault
    FILTER CONTAINS(str(?local), '{queried-data-vault}')

    OPTIONAL {
      ?concept consolid:aggregates ?alias .
      # remote references do not include the URL of queried data vault
      FILTER regex(str(?alias), '^((?!{queried-data-vault}).)*$')
    }
}
```

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

```
SELECT ?concept ?reference ?value ?doc ?meta
WHERE {
    BIND(<{concept}> as ?concept) # concept is known and added as variable
    BIND(<{reference}> as ?reference) # reference is known and added as variable

    <{reference}> consolid:hasIdentifier ?id .

    ?id consolid:inDocument ?doc ;
        schema:value ?value .

    ?meta dcat:distribution/dcat:downloadURL ?doc .
    # optional further filters for metadata (e.g. considering media type, publication status,
     version ...)
}}
```

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

```
`SELECT ?concept ?reference ?value ?doc ?meta ?alias
    WHERE {
        BIND({concept} as ?concept) # the starting concept is known and will be used to
    aggregate all references client-side
        BIND({alias} as ?alias) # the alias is known

        <{alias}> consolid:aggregates ?reference .
        ?reference consolid:hasIdentifier ?id .

        ?id consolid:inDocument ?doc ;
            schema:value ?value .

        ?meta dcat:distribution/dcat:downloadURL ?doc .
        # optional further filters for metadata (e.g. considering media type, publication
    status, version ...)
    }
```

Listing 13. Recursive query to find all remote representations, identifiers and datasets for a given selected concept

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 – whilst enriching existing project resources or setting up a project in the ecosystem from its conception. 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 amount of existing concepts. For diagnosing many aliases of the same concept, mapping algorithms such as proposed in [25] 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 4.4, *Daapi*. The API is available on Github[22] and NPM[23] as the Reference Aggregation API ('Raapi').

6. Demonstration

This section demonstrates the usage of the ecosystem with a real-world example. This proof of concept illustrates the ecosystem's ability to combine federated, heterogeneous and multi-disciplinary datasets in a global multi-model. We estimate the Technology Readiness Level (TRL) [26] of the ecosystem at TRL 3, which corresponds with 'Analytical and experimental critical function and/or characteristic proof of concept' [40]. Because the ecosystem is highly experimental, the stakeholders mentioned in this proof of concept did not use the ecosystem in a real-world project, which would require a much more mature graphical user interface (GUI). However, the two stakeholders involved in the eventual enrichment case, namely the Facility Management office of UGent and Bureau Bouwtechniek (BE), were contacted to confirm the validity of the scenario, which involves enriching the federated project with damage record data.

The case of damage enrichment is considered 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 [41]. However, it is a cumbersome activity which is carried out in a largely manual fashion [14], and, if digitized at all, the resulting reports are most often spreadsheets or form results [41], detached from other digital information about the asset. Indeed, the use of BIM (Building Information Management) models in this regard is scarce [24,35], also because BIM models are often not updated or documented to reflect the as-built state [44]. Semantic Web-based documentation of damages, which allows a more interdisciplinary approach, is the idea behind the Damage Topology Ontology (DOT)[24] [14]. For

[22]Raapi (Github): https://github.com/LBD-Hackers/raapi/tree/paper, accessed 27/01/2023.
[23]Raapi (NPM): https://www.npmjs.com/package/consolid-raapi, accessed 27/01/2023.
[24]DOT: https://w3id.org/dot#, accessed 27/01/2023.

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 present the actual building case and stakeholder constellation; then we explain which datasets and distributions have been created. We end the section with a brief validation of the case, from the perspective of a GUI oriented towards damage enrichment.

6.1. *Case description and setup*

Every stakeholder maintains their own Solid 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 Intellectual Property Rights (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. This setup is visualized in Fig. 6.

As a preparation step, the original Autodesk Revit partial BIM (Building Information Modelling) [11] models[25] for Architecture (modeled by Bureau Bouwtechniek (BE)), Engineering, Electricity, and HVAC (Heating, Ventilation and Air-Conditioning) (all three modeled by Arcadis (BE)) were converted to IFC (Industry Foundation Classes, ISO 16739)[26] [22] and then to a semantic model for RDF (Turtle) and a geometric model (glTF).[27]

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 automatically mirrored as named graphs on the SPARQL satellite, as are the RDF-based semantics, derived from the partial models and structured using the Linked Building Data (LBD) ontologies that are implemented in the IFCtoLBD converter [5]: the Building Topology Ontology (BOT)[28] [34] and the BuildingElement Ontology (BEO).[29]

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 Pods, even when referring to data on other servers.

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 [25] to automatically link RDF concepts with their GUID-based counterpart in the 3D model. However, in current industry practice, partial BIM models are

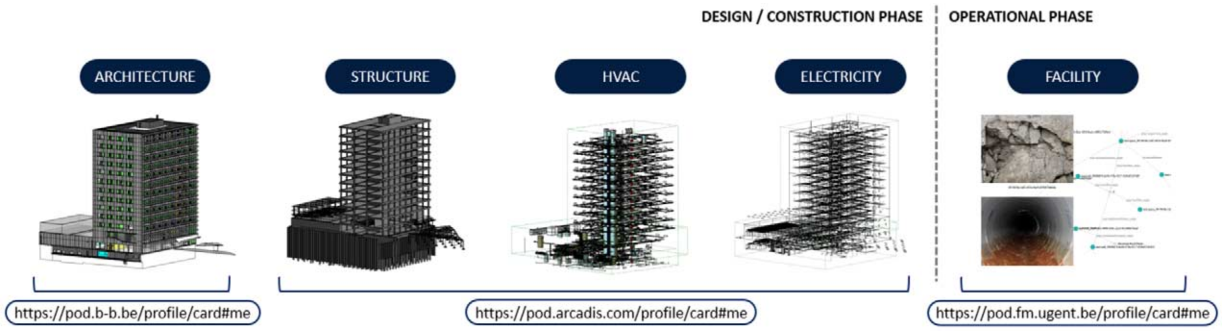

DESIGN / CONSTRUCTION PHASE
OPERATIONAL PHASE
ARCHITECTURE
STRUCTURE
HVAC
ELECTRICITY
FACILITY
https://pod.b-b.be/profile/card#me
https://pod.arcadis.com/profile/card#me
https://pod.fm.ugent.be/profile/card#me

Fig. 6. Use case setup and model IPR.

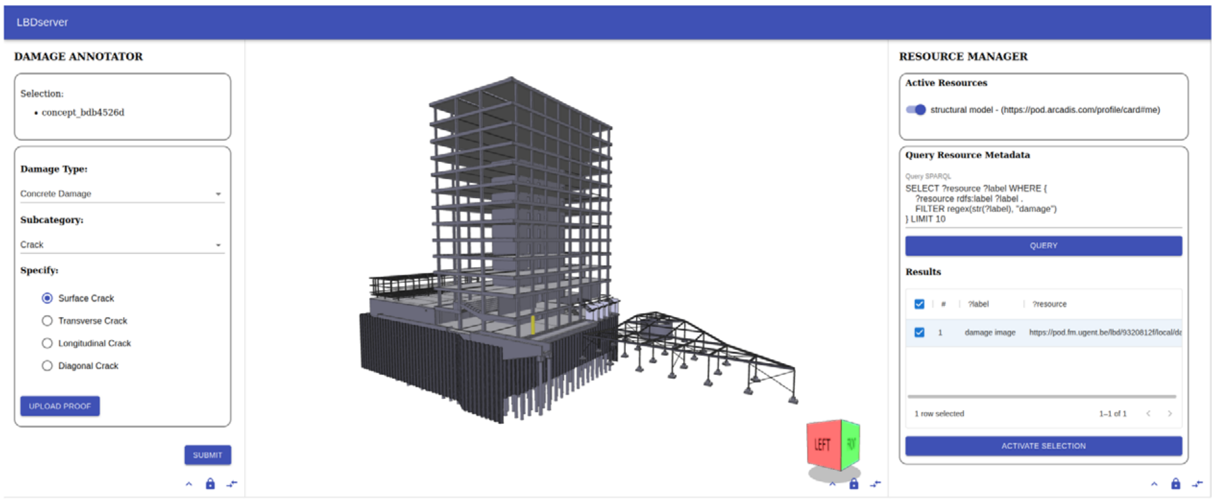
[25]Partial models are sub-models for an AECO project related to a particular discipline and created by specialist stakeholders.

[26]The Industry Foundation Classes are the international open standard for digital description of built assets, developed by buildingSMART International.

[27]glTF: https://registry.khronos.org/glTF/specs/2.0/glTF-2.0.html, accessed 27/01/2023.

[28]BOT, https://w3id.org/bot#, accessed 27/01/2023.

[29]BEO: http://pi.pauwel.be/voc/buildingelement#, accessed 27/01/2023.



Selection:
- concept_bdb4526d

Damage Type: Concrete Damage

Subcategory: Crack

Specify: [✓ Surface Crack] Transverse Crack Longitudinal Crack Diagonal Crack

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

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

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

1 row selected 1–1 of 1

Fig. 7. Protoype 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, specialized UI modules can be created and combined depending on the use case.

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.

We will discuss a scenario where the Facility Manager localizes damages on a regular building element and documents it in their Pod, thereby referencing the external federated models of other stakeholders (in this case Bureau Bouwtechniek as creator of the architectural BIM model) and semantically enriching the overall project. The following heterogeneous datasets will be linked: federated semantics (a building graph at and a damage graph) with a 3D element in a geometric model and an image pixel zone. The involved resources are managed by different stakeholders. The demonstration is reproducible with the software prototypes mentioned in earlier sections, executing the scripts at https://github.com/LBD-Hackers/consolid-demo/tree/paper consecutively and following the guidelines described in the repository's README document. However, the model used for reproducing the experiment is based on the benchmark dataset Duplex,[30] an IP-free model dataset.

Below, we describe the raw data structures created by these steps. However, it should be clear that an agent will never initiate these interactions directly but use a dedicated GUI for every purpose. A prototypical GUI for the case of damage enrichment is illustrated in Fig. 7.

6.2. *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 metadata record on their Pod (see respectively the prefixes `damageMetaFM` and `damageDistFM` in Listing 14), referencing an RDF-based distribution, which is created simultaneously. 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'. 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) [14].

At first, two graph-specific identifiers are created, namely one for the damaged element (`damageDistFM:instance_a003d5d4`) and one for the damage area (`damageDistFM:instance_8aa5bc1b`) (Listing 14). 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 15). These

[30]Duplex dataset: https://github.com/LBD-Hackers/consolid-demo/tree/paper/src/resources/duplex, accessed 27/01/2023.

```
# The RDF resource on the Facility Manager's Pod that documents the damage
@prefix damageDistFM: <https://fm.ugent.be/cac6d088#> .
@prefix damageMetaFM: <https://fm.ugent.be/3c1c9a18#> .

damageDistFM:instance_a003d5d4
    dot:hasDamageArea damageDistFM:instance_8aa5bc1b ;
    a dot:ClassifiedDamage, cdo:SurfaceCrack ;
    cdo:crackWidth "35" .

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

Listing 14. Triples in the distribution of the Damage dataset, maintained by the FM. The metadata graph will be similar to the one listed in Listing 3

```
# The Reference Registries of the Facility Manager and Bureau Bouwtechniek
@prefix refFM: <https://fm.ugent.be/b677ce97#> .
@prefix refArch: <https://pod.b-b.be/f0c8cb37#> .

# The RDF resource on the Facility Manager's Pod that documents the damage
@prefix damageDistFM: <https://fm.ugent.be/cac6d088#> .

# 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#instance_a003d5d4"^^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#instance_8aa5bc1b"^^xsd:anyURI .
```

Listing 15. 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

(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.

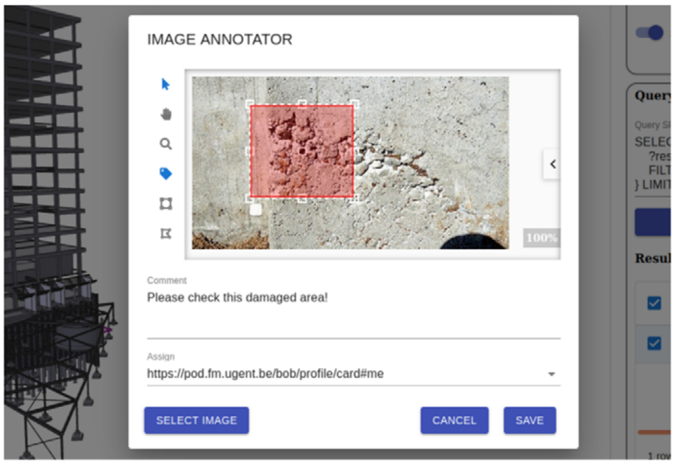
6.3. *Lifting local identifiers to project level*

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. Because of this, 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 15). A notification must now be sent to Bureau Bouwtechniek to automatically or manually aggregate this new concept as well, so bi-directional discovery is made possible.

6.4. 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 shows a possible interface which can be used by the handling employee to interact with the project data in a non-technical way. 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 16, which uses the International Image Interoperability Framework (IIIF) specification's image API[31] 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.

This concludes the enrichment scenario. All stakeholders in the project can now discover and query this information starting from their own project access point.



Comment
Please check this damaged area!

Assign
https://pod.fm.ugent.be/bob/profile/card#me

Fig. 8. An image region annotator helps in interacting with the ecosystem in a non-technical way, implementing domain-specific views on top of it.

```
# The Reference Registries of the Facility Manager and the Architecture Office
@prefix refFM: <https://fm.ugent.be/b677ce97#> .
@prefix refArch: <https://pod.b-b.be/f0c8cb37#> .

# The image resource on the Facility Manager's Pod that shows the damage
@prefix pictureDistFM: <https://fm.ugent.be/9f9b1794> .

# 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/pct:20,10,25,55/max/0/default"^^xsd:string .
```

Listing 16. Defining an image's pixel zone as a specific representation of an abstract concept, identified by a Reference Aggregator

[31] IIIF: https://iiif.io/api/image/3.0/, accessed 27/01/2023.

6.5. Proof-of-concept: Querying the project graph

A client can now 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 visualize the results and access knowledge about an object documented in other resources via dedicated GUIs. In the above described case, a GUI that offers a perspective of damage management would allow an agent to perform the following steps:

– Authenticate with a Solid WebID.
– Select the project (access point) of interest and discover the SPARQL satellites (Section 3.4) of its aggregated catalogs in order to query the permissioned union graphs containing these catalogs.
– Query the project (via the satllite) for damage assessments (Listing 17) and find the damaged element. In Listing 17, this corresponds with the variables '?el' and '?doc'. Results are included in Table 3.
– 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, along with its local and remote representations. With the query results from Table 3 as input, this yields the data indicated in Table 4.
– Inject the query results given in Table 4 for '?local' into the query pattern of Listing 12. The result of this query is given in Table 5.
– Inject the query results given in Table 4 for '?alias' into the query pattern of Listing 13. The result of this query is given in Table 6.
– The concept and its file-specific representations are the union of the results in Tables 5 and 6. This information can now be visualized in a GUI.

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

Listing 17. 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

Table 3
Results of the query in Listing 17. In this case, the prefix damageDistFM corresponds with the result of the '?doc' variable

?el	<https://fm.ugent.be/cac6d088#instance_a003d5d4>
?doc	<https://fm.ugent.be/cac6d088>

Table 4
Concept identifier and representation identifiers for the results of the query in Listing 17

?concept	<https://fm.ugent.be/b677ce97#ra_bdb4526d>
?local	<https://fm.ugent.be/b677ce97#ref_a80854ae>
?alias	<https://pod.b-b.be/f0c8cb37#ra_3de17fbe>

Table 5
Results of the query in Listing 17. In this case, the prefix damageDistFM corresponds with the result of the '?doc' variable

?concept	<https://fm.ugent.be/b677ce97#**ra**_bdb4526d> *(abstract concept (alias in fm-ugent pod))*
?reference	<https://fm.ugent.be/b677ce97#**ref**_a80854ae>
?value	<https://fm.ugent.be/cac6d088#**instance**_a003d5d4>
?doc	<https://fm.ugent.be/cac6d088>
?meta	<https://fm.ugent.be/3c1c9a18>

Table 6

Resulting concept alias and its two external references data on the scale of the federated project

?concept	<https://fm.ugent.be/b677ce97#**ra**_bdb4526d> *(abstract concept (alias in fm-ugent pod))*
?alias	<https://pod.b-b.be/f0c8cb37#**ra**_3de17fbe> *(abstract concept (alias in b-b Pod))*
?reference	<https://pod.b-b.be/f0c8cb37#**ref**_8894fd07> *
?value	“1uxwRB00H01B1i8VRQsVcL” *(the geometric identifier in context of ?doc)**
?doc	<https://pod.b-b.be/12008088> *(the URL of the glTF model)**
?meta	<https://pod.b-b.be/e7d45e01> *(the metadata URL of the glTF model)**
?concept	<https://fm.ugent.be/b677ce97#**ra**_bdb4526d> *(abstract concept (alias in fm-ugent pod)*
?alias	<https://pod.b-b.be/f0c8cb37#**ra**_3de17fbe> *abstract concept (alias in b-b Pod))*
?reference	<https://pod.b-b.be/f0c8cb37#**ref**_2ced93d3> *
?value	<https://pod.b-b.be/12008088#**instance**_6f8474e0< *(the semantic identifier in context of ?doc)**
?doc	<https://pod.b-b.be/12008088> *(the URL of the semantic model)**
?meta	<https://pod.b-b.be/6a4303c2> *(the metadata URL of the semantic model)**

* = identifiers not indicated in earlier listings, hosted in resources on the Pod of Bureau Bouwtechniek

These steps are generally applicable for retrieving heterogeneous sub-document identifiers in aggregated projects, based on a combination of domain-specific queries (Listing 17) and ConSolid data patterns (Listings 11 and 12 and 13).

With the setup described in Section 3.4 and the demo scripts available on Github,[32] we found that a concept retrieval action generally takes about 1 s. A lot of execution time is due to the many requests back and forth, and the fact that all requests are being rerouted via the ExpressJS middleware between the client and the SPARQL store (Section 3.4). The queries used in this paper generally retrieve only one concept at the time, although optimizations can be made to the algorithm when retrieving multiple concepts at once. For example, when a document is opened, a single query can already determine all concept URLs for its identifiers, avoiding the need to execute the query in Listing 11 for every concept. Furthermore, a dedicated endpoint can be set up in the SPARQL satellite to bundle the access control checking step for multiple queries.

7. Evaluation

In this section, we evaluate the ecosystem against the original research question and the objectives mentioned in Section 1.2. We also indicated in Section 1.4 that there are significant parallels between interdisciplinary digital twins and open data: although the project-specific part of the former will often consist of access-restricted datasets, the general multi-disciplinary context in which the industry operates, along with its federated nature, has as a result that it heavily benefits from data being findable, accessible, interoperable and reusable. Therefore, evaluation of the ecosystem against these principles makes sense. Hence, the framework will also be evaluated against the *FAIRification* principles indicated in [12].

7.1. Objective 1: Federation

One of the core goals of the ecosystem is to address the double patchwork of stakeholders and projects in a way that naturally mirrors knowledge production in the AECO industry. Based upon the Solid ecosystem and the concept of WebIDs, access to the information generated by individual stakeholders remains under full control by those offices, as well as its legal ownership and intellectual property rights over those documents. Semantic enrichment and alignment of digital twin concepts can take place in an asynchronous way, using concept aliases.

[32]ConSolid demo; https://github.com/LBD-Hackers/consolid-demo/tree/paper, accessed 24/02/2023.

7.2. Objective 2 and 3: Discipline-agnostic data patterns and URL stability

The ecosystem fully works on metadata level, both for describing project resources and discovering them via an expressive, query-based procedure. The application of query-based views eliminates the need for implicit semantics in the URL and allows a more flexible aggregation of project resources, depending on the task at hand. Furthermore, changes in document statuses or other metadata will not break the resource's URL, guaranteeing stability during the resource's publication life cycle.

7.3. Objective 4, 5 and 6: Sub-document linking of heterogeneous datasets

Resources in the ecosystem can use of any (domain-specific) file schema or RDF ontology. Membership of one or more ConSolid projects does not affect the content of a resource, e.g., a BIM model can still be opened in the authoring tool that was used to create it. The heterogeneity challenge is also related to the ability to connect identifiers on a sub-document level. When resources use open standards, sub-document linking is rather straightforward, as identifiers are often embedded in a publicly documented JSON (e.g., glTF) or XML (e.g. COLLADA[33]) structure or use a standardized media type. In the case of resources encoded in proprietary, encrypted data schemas, sub-document linking may not be possible out-of-the-box, unless the company developing the data format offers an API that is able to parse the resource. For example, the Autodesk Platform Services[34] (APS) (formerly Autodesk Forge) would allow to read an (encrypted) Revit BIM model and hence to link its sub-document identifiers with the abstract concepts of the ConSolid ecosystem, in a Reference Registry. In the case of unstructured data formats, sub-document linking needs to happen entirely decoupled from the original resource. This was demonstrated with the usage of the IIIF specification for determining pixel zones on imagery (Section 6.4). A similar approach may be identified for other unstructured datasets such as point clouds.

7.4. Objective 7: Embedding external datasets

A federated approach for managing the built environment allows many actors to make statements about a built asset or its context. Adoption of the proposed ecosystem by third parties who provide contextual datasets is beyond the control of a project consortium. However, the metadata-based discovery described in Section 4 allows project members to include external datasets (RDF and non-RDF) in the federated project catalog, hence making them discoverable within the project, and clarify their purpose in the project. Sub-document references can be made to such external datasets as well (Section 4.3).

7.5. FAIR principles

In [12], the FAIR principles are subdivided into sub-requirements for each principle. This section lists these sub-requirements and explain them from the perspective of the ConSolid ecosystem.

- **F1. (Meta)data are assigned a globally unique and persistent identifier**: every resource has a URL, which is made as stable as possible by eliminating implicit semantics.
- **F2. Data are described with rich metadata (defined by R1 below)**: all project resources are described in metadata records. Because these records are RDF-based, they can be used for fine-grained filtering and discovery of project resources.
- **F3. Metadata clearly and explicitly include the identifier of the data they describe**: this is done using DCAT distributions.
- **F4. (Meta)data are registered or indexed in a searchable resource.**: The ecosystem bases on recursive catalogs to allow clients to index relevant (meta)data on-the-go and cache this index for future usage. The SPARQL interface to the data vault offers a union of its RDF resources, allowing quick querying of project (meta)data.

[33]COLLADA: https://www.khronos.org/collada/, accessed 27/01/2023.

[34]Autodesk Platform Services: https://aps.autodesk.com/, accessed 27/01/2023.

- **A1. (Meta)data are retrievable by their identifier using a standardized communications protocol**: All resources on a data vault get a HTTP URL.
- **A2. Metadata are accessible, even when the data are no longer available**: at this moment this is not ensured by the ecosystem. As data can be self-hosted, the archival of (meta)data and retention policies can not yet be controlled in a technical way. In multi-stakeholder consortia, however, legal ways exist to enforce this.
- **I1. (Meta)data use a formal, accessible, shared, and broadly applicable language for knowledge representation**: Metadata are recorded using RDF, which the main W3C framework for knowledge representation.
- **I2. (Meta)data use vocabularies that follow FAIR principles**: in the FAIR data points specification [27], the DCAT vocabulary is indicated as the basis for metadata content.
- **I3. (Meta)data include qualified references to other (meta)data**: in [12], qualified references mean that (1) it is specified if one dataset builds on other datasets, (2) if additional datasets are needed to complete the data or (3) if complementary information is stored in different datasets. In the ConSolid ecosystem, metadata is formed by a union of the data catalogs on a vault and the reference registry. This union can be queried via the SPARQL interface, to retrieve dataset aggregations and complementary information for project resources.
- **R1. (Meta)data are richly described with a plurality of accurate and relevant attributes**: according to [12], this relates to metadata describing the context under which the metadata was generated to determine the usefulness of a particular dataset. This usefulness will, of course, depend on the purpose of the service requesting the data. In the context of construction data, we mentioned the CDC ontology. As DCAT metadata is graph-based, it can be combined with domain-specific vocabularies in a straightforward way, in order to allow interdisciplinary purposes.

8. Discussion and conclusion

In this paper, we proposed an architecture for organising federated datasets for the AECO 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 containers). For the microservices, we discussed the benefits of an access-controlled SPARQL interface to a Pod, mainly aiding in resource discovery, and provided a viable approach to establish such interface. Based on this SPARQL endpoint, virtual views can be constructed to group similar resources in 'virtual containers' or present 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 nested project data which is hosted by different stakeholders is easily discoverable with a single triple pattern. This introduces a flexibility to fine-grainedly include relevant datasets at the level of individual project partners (e.g., subcontractor datasets, visitor datasets, etc.). In the same move, a scalable approach for interacting with 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 organization 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. The evaluation section showed that the ecosystem is in line with existing efforts towards FAIR data creation and management. To make the ecosystem more mature and industry-ready, its performance needs to improve. We indicated several possibilities to achieve such improvements, for querying (meta)data in general and for concept retrieval specifically.

Eliminating the hard-coded hierarchy of folders on a Solid Pod 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 needs to focus on alternative solutions, e.g., establishing inheritance of ACL rules via metadata-based reasoning.

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 AECO 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, and the layering of (modular) interfaces to the project(s) should be further investigated. 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 needs to focus on how this can be achieved.

When flexible combinations of access-controlled, interdisciplinary datasets are to be made, 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 data vaults, and contextual information, leveraging the enormous amount of knowledge on the Web.

Acknowledgements

This research is funded by the Research Foundation Flanders (FWO), as a Strategic Basic Research grant (grant no. 1S99020N). The authors acknowledge the support of the DigiChecks project, which has received funding from the European Union's Horizon Research and Innovation Program under grant agreement no. 101058541, and of SolidLab SolidLab Vlaanderen (Flemish Government, EWI and RRF project V023/10). We thank Bureau Bouwtechniek and Arcadis for granting permission to use their models in this paper, and the Facility Management office of UGent for giving insights into their current workflows.

Appendix. Prefixes used in this paper

Listing 18 contains a list of prefixes used throughout this paper, which refer to ontologies. Prefixes related to a data vault or a specific resource on a data vault 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: <https://pi.pauwel.be/voc/buildingelement#> .
@prefix bot: <https://w3id.org/bot#> .
@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/> . #used for placeholder definitions
@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 18. Prefixes used in this paper

References

[1] R. Albertoni, D. Browning, S. Cox, A. Gonzales Beltran, A. Perego and P. Winstanley, Data Catalog Vocabulary (DCAT) – Version 3, W3C Recommendation, W3C, 2023, https://www.w3.org/TR/vocab-dcat-3.

[2] J. Beetz, Facilitating distributed collaboration in the AEC/FM sector using Semantic Web Technologies, PhD thesis, Technische Universiteit Eindhoven, 2009.

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

[4] M. Bonduel, A Framework for a Linked Data-based Heritage BIM, PhD thesis, KU Leuven, 2021.

[5] 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.

[6] Box, Inc., The Information Economy: A Study of Five Industries, Technical Report, Box, Inc., 2014, https://img.forconstructionpros.com/files/base/acbm/fcp/document/2014/06/box-cloud-study_11535206.pdf.

[7] E. Commission, reform of EU data protection rules, *European Commission* **2018** (2018), https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679.

[8] 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, https://ieeexplore.ieee.org/abstract/document/9684521. doi:10.1109/COMST.2022.3143147.

[9] R. Dedecker, W. Slabbinck, J. Wright, P. Hochstenbach, P. Colpaert and R. Verborgh, What's in a Pod? – A knowledge graph interpretation for the solid ecosystem, in: *Proceedings of the 6th Workshop on Storing, Querying and Benchmarking Knowledge Graphs*, 2022, https://solidlabresearch.github.io/WhatsInAPod/.

[10] N. Drummond and R. Shearer, The open world assumption, in: *eSI Workshop: The Closed World of Databases Meets the Open World of the Semantic Web*, Vol. 15, 2006, p. 1, https://www.cs.man.ac.uk/~drummond/presentations/OWA.pdf.

[11] 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, Wiley, New York, United States, 2011, p. 650, 978-0-470-54137-1. ISBN 978-0-470-54137-1.

[12] GO FAIR, FAIR principles, 2016, https://www.go-fair.org/fair-principles/.

[13] 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. doi:10.1007/978-3-319-24141-8_21.

[14] 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.

[15] D. Hardt, The OAuth 2.0 authorization framework, Technical Report, 2012.

[16] O. Hartig, An overview on execution strategies for Linked Data queries, *Datenbank-Spektrum* **13**(2) (2013), 89–99, https://link.springer.com/article/10.1007/s13222-013-0122-1. doi:10.1007/s13222-013-0122-1.

[17] HH Judge Eyre QC, Mott Macdonald Ltd v Trant Engineering Ltd [2021] EWHC 754 (TCC), https://www.bailii.org/ew/cases/EWHC/TCC/2021/754.html.

[18] P. Hummel, M. Braun, M. Tretter and P. Dabrock, Data sovereignty: A review, *Big Data & Society* **8**(1) (2021), https://journals.sagepub.com/doi/10.1177/2053951720982012.

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

[20] International Data Spaces, Data Sovereignty – you decide how your data gets used! International Data Spaces, 2020, https://internationaldataspaces.org/why/data-sovereignty.

[21] ISO 19650-1:2018 Organization and digitization of information about buildings and civil engineering works, including building information modelling (BIM) – Information management using building information modelling – Part 1: Concepts and principles, Standard, International Organization for Standardization, 2018.

[22] ISO 16739-1:2018 Industry Foundation Classes (IFC) for data sharing in the construction and facility management industries, Standard, International Organization for Standardization, 2018.

[23] P. Klinov, FROM vs FROM NAMED in SPARQL, 2021, https://www.stardog.com/labs/blog/from-vs-from-named-in-sparql/.

[24] 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, https://link.springer.com/chapter/10.1007/978-3-030-73043-7_21.

[25] 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, https://pure.tue.nl/ws/files/163377646/5FMA_paper_34_final.pdf.

[26] J.C. Mankins et al., *Technology Readiness Levels, White Paper, April*, Vol. 6, 1995, https://aiaa.kavi.com/apps/group_public/download.php/2212/TRLs_MankinsPaper_1995.pdf.

[27] L. Olavo Bonino, K. Burger and R. Kaliyaperumal, 2022, FAIR Data Point – working draft, GO FAIR, https://specs.fairdatapoint.org/.

[28] Open Knowledge Foundation, The Open Definition, Accessed: 11/01/2023, https://opendefinition.org/.

[29] B. Otto, GAIA-X and IDS, Zenodo, 2021. doi:10.5281/zenodo.5675897.

[30] B. Otto, M.T. Hompel and S. Wrobel, International data spaces, in: *Digital Transformation*, Springer, 2019, pp. 109–128, https://link.springer.com/chapter/10.1007/978-3-662-58134-6_8. doi:10.1007/978-3-662-58134-6_8.

[31] F. Parrillo and C. Tschudin, *Solid over the Interplanetary File System, in: 2021 IFIP Networking Conference (IFIP Networking)*, IEEE, 2021, pp. 1–6, https://ieeexplore.ieee.org/abstract/document/9472772.

[32] 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.

[33] M.H. Rasmussen, The vision of a decentralized, distributed AEC information infrastructure using LBD technologies, 2018, https://drive.google.com/file/d/1MWbwh3ihzhMtF_pj0G0t3XYP0glJ7v5o/view.

[34] 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.

[35] M.K. Seeaed and A. Hamdan, BIMification of stone walls for maintenance management by utilizing Linked Data, in: *31st Forum Bauinformatik*, 2019, https://depositonce.tu-berlin.de/bitstreams/de1e386f-e0e1-4bb2-a6f7-e821cfb20021/download#page=179.

[36] 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, https://www.mdpi.com/2571-5577/4/2/36/pdf. doi:10.3390/asi4020036.

[37] Solid, WebID-OIDC Authentication Spec, Specification, Solid, 2019, https://github.com/solid/webid-oidc-spec.

[38] W. Solid, Access Control Spec, Specification, Solid, 2022, https://solidproject.org/TR/wac.

[39] 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, https://link.springer.com/chapter/10.1007/978-3-030-00668-6_15.

[40] I. Tzinis and T.R. Level, *National Aeronautics and Space Administration (NASA)* (2012), https://www.nasa.gov/directorates/heo/scan/engineering/technology/technology_readiness_level.

[41] 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*, E. Verstrynge, B. van Bommel, N. Vernimme and R. van Hees, eds, Vol. 35, 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.

[42] R. Verborgh, Solid: Innovation through personal data control, 2021, https://rubenverborgh.github.io/ECA-2021/#title.

[43] M. Verstraete, S. Verbrugge and D. Colle, Solid: Enabler of decentralized, digital platforms ecosystems, in: *31st ITS European Conference*, 2022, pp. 1–19, https://biblio.ugent.be/publication/8760525/file/8760526.

[44] 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.

[45] 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, https://biblio.ugent.be/publication/8724183/file/8750812.pdf.

[46] 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, https://ceur-ws.org/Vol-3213/paper06.pdf.

[47] 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.

[48] 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. doi:10.1016/j.autcon.2020.103286.

[49] J. Werbrouck, O. Schulz, J. Oraskari, E. Mannens, P. Pauwels and J. Beetz, A Generic Framework for Federated CDEs applied to Issue Management, Under review.

[50] J. Werbrouck, M. Senthilvel and M.H. Rasmussen, Federated data storage for the AEC industry, in: *Buildings and Semantics*, CRC Press, 2022, pp. 139–164. doi:10.1201/9781003204381-8.

[51] 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, https://ceur-ws.org/Vol-2636/09paper.pdf.

[52] M.D. Wilkinson, M. Dumontier, I.J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J.-W. Boiten, L.B. da Silva Santos, P.E. Bourne et al., The FAIR guiding principles for scientific data management and stewardship, *Scientific data* **3**(1) (2016), 1–9. doi:10.1038/sdata.2016.18.