The Web as a Common Data Environment: Management of Federated Multi-Models

Jeroen Werbrouck

Doctoral dissertation submitted to obtain the academic degrees of

Doctor of Architectural Sciences and Engineering (UGent) and
Doktor der Ingenieurwissenschaften (Dr.-Ing.) (RWTH)

Supervisors

Prof. Pieter Pauwels, PhD* - Prof. Jakob Beetz, PhD** - Prof. Erik Mannens, PhD***

* Department of Architecture and Urban Planning
Faculty of Engineering and Architecture, Ghent University

** Design Computation
Faculty of Architecture, RWTH Aachen University, Germany

*** Department of Electronics and Information Systems
Faculty of Engineering and Architecture, Ghent University

April 2024

GHENT UNIVERSITY

RWTH AACHEN UNIVERSITY

ISBN 978-94-6355-826-6
NUR 986, 957
Wettelijk depot: D/2024/10.500/31

Members of the Examination Board

Chairs

Prof. Patrick De Baets, PhD, Ghent University
Prof. Marcel Schweiker, PhD, RWTH Aachen University, Germany

Other members entitled to vote

Prof. Sigrid Brell-Cokcan, PhD, RWTH Aachen University, Germany
Ben De Meester, PhD, Ghent University
Prof. Linda Hildebrandt, PhD, RWTH Aachen University, Germany
Prof. Jelle Laverge, PhD, Ghent University
Maria Poveda-Villalon, PhD, Universidad Politécnica de Madrid, Spain

Supervisors

Prof. Pieter Pauwels, PhD, Ghent University
Prof. Jakob Beetz, PhD, RWTH Aachen University, Germany
Prof. Erik Mannens, PhD, Ghent University

Acknowledgements

This dissertation is the result of a five-year journey, which would not have been possible without the support of many people. Firstly, I wish to thank my supervisors Pieter Pauwels, Jakob Beetz and Erik Mannens, who have guided me throughout the years and offered me plenty of opportunities for disseminating my work in an international context. Thanks also to Ruben Verborgh and Pieter Colpaert, who have provided invaluable feedback on many occasions. I would like to express my gratitude to Research Foundation Flanders (FWO) to believe in the project and provide 4 years of funding, and to the H2020 project BIM4Ren to kickstart the first year of my research.

I have always found a warm welcome among my colleagues in the 'Landscape/Rommelaere Office' of the Department of Architecture and Urban Planning, the people from Digital Design Techniques and the team from 'Knowledge on Web Scale' (KNoWS). The same holds for the chair of Design Computation at RWTH Aachen, where I could stay multiple times in an amazing and fun research context. Over the years, I had the pleasure to work with extremely talented minds on various international occasions. Mads, Olli, Madhu, Alex, Jyrki, Ruben, Mathias and Anna, thank you for your enthusiasm, for the inspiring semantic ping-pong, the engaging discussions, the hackathons.

Bedankt aan iedereen bij wie ik de voorbije jaren terechtkon voor ontspanning en een goeie babbel. Team Dolfijn – Seppe, Sam, Eliah, Eline, Isabeau, Velika, Freya, Andrew – voor alle spelletjes, het gezang, de mopjes, de weekendjes en de café's. Anton, Jolien, Seppe en Zoë voor de gezellige avonden die inmiddels ook al vele jaren teruggaan. Louis, Haroun, Joke, Marie, Sofie, Annelies, Paco, Veronica, Felix, Rik, Quinten, Lars, Soetkin en alle leden van de Dagen van de Huismuziek, om twee weken per jaar een magische muziekbubbel te blazen. Tim, Flor en Thomas; en Philippe, Thijs, Miró, Louis en Nathalie om dit traject te doorspekken met zoveel muzikale energie.

In het bijzonder aan mama en papa, dank jullie wel voor het warme nest en om me te steunen bij alles wat ik onderneem. Hetzelfde geldt voor oma Betty en opa Maurits, oma Annie en opa Wilfried: bedankt – altijd fijn om langs te komen. En, natuurlijk, Andreas, Jan en Marijn, voor de zever en de sfeer, en het samen mogen opgroeien.

Gent, 9 april 2024
Jeroen Werbrouck

Contents

Summary (English)

Information related to the built environment is produced by a multitude of actors, many of whom only occasionally interact with one another. In addition to a core team that consists of a client, an architect and project engineers, a construction project often includes contributions from a more extensive network of contractors, subcontractors, consulting firms, etc. In later stages, facility managers, owners and occupants are added to this network, as well as external agencies. Examples of such agencies are governmental institutions, research institutions, surveyors and infrastructure specialists. Each project thus has multiple partners, and each partner may be working on multiple projects at the same time. These overlapping networks of actors and projects makes it possible nor desirable to centralise 'all' available information on a digital platform. Nonetheless, a centralised 'Common Data Environment' (CDE) is often seen as the only solution for information management related to the built environment: a single Web platform for integration of project information. Often, the only way to access information on a CDE is by using a proprietary interface, usually provided by a software company that is not involved in the project.

A CDE is deployed per project, meaning that an office may have to maintain several CDEs simultaneously. In addition, project data itself (often a BIM model) is often described in a proprietary format, so it is not easy to link it with other project data (pictures, point clouds, planning), and to find reusable information from adjacent domains (GIS, damage surveys, user data, historical datasets) or other life cycle phases. Even with open, standardised formats such as IFC (Industry Foundation Classes), linking with other data is not always evident, due to the existing hierarchy between the 3D BIM model on the one hand and the linked documents on the other.

This dissertation explores an alternative path, which starts from the decentralised, multidisciplinary and heterogeneous nature of the built environment. The basis for a 'federated CDE' is an infrastructure that allows using the worldwide Web as a secure and scalable framework for storing sensitive, interconnected data. A 'federated project' is then an aggregation of project-specific and contextual datasets, which may include and exclude specific datasets depending on the task at hand. These aggregations are likely to be heterogeneous in nature, e.g., consisting of BIM models, imagery, point clouds, spreadsheets, regulatory datasets etc. Nevertheless, to form an over-arching information

catalogue, they must be connected to one another in a way that is independent from the mediatypes used. The prerequisites for such an infrastructure will be identified in this dissertation, as well as current technologies that can fulfil these prerequisites. In addition, a technological implementation is devised that illustrates the overall feasibility of the approach, while demonstrating how compatibility with existing BIM standards can be largely maintained – by strictly separating project datasets and metadata. Where project datasets contain the actual information related to the built environment, metadata records offer the necessary context to identify suitable tools for interacting with a dataset. This allows the developed data patterns to be used for purposes other than documenting the built environment: the format or schema of project data does not matter. Consequently, the foundations of the ecosystem will be domain-independent. This also means that connections can be made between disparate documents. For example, one document may contain a 3D representation of a specific object, another a photograph and yet another one a semantic description. The result is an interdisciplinary catalogue of building data that is scalable across the Web – a federated 'multi-model'.

In this work, the developed ecosystem for decentralised data management on the Web is referred to as ConSolid. ConSolid is based on the Semantic Web, the Solid protocol for decentral data vaults ('Pods'), and the FAIR (Findable, Accessible, Interoperable, Reusable) principles for Web-based data management. Where deemed necessary, extensions will be proposed to deploy the Solid ecosystem as the basis for a federated CDE, such as a SPARQL endpoint over an entire data vault and an extendible metadata structure based on the Data Catalog (DCAT) ontology.

This extended infrastructure allows various higher-level data management processes to take place. Firstly, data patterns are proposed to create, assign and validate project-specific requirements to metadata records. Secondly, an approach for an extended access control environment is devised, where semantic properties of both visitors and resources can be validated to determine whether access to this resource can be granted. For example: 'everyone who can prove they are employed by the architecture office involved in this project, gains read access to the datasets with a 'shared' label.'

However, in order to effectively use ConSolid as a CDE, a service layer is necessary on top of this generic storage layer. At its core, this service layer allows to interpret the proposed data patterns, aggregate the necessary data fragments and adapt them to an output that conforms to existing, standardised,

domain-specific formats. With this service layer, a federated infrastructure can nevertheless be used in familiar industry environments that expect the data to be available through centralised endpoints (e.g. BIM authoring tools) .

However, such desktop environments rarely give access to the expressivity offered by (Web-wide) multi-models. As indicated before, it is – by definition – uncertain what data formats such a catalogue will contain. Consequently, human interaction with so much heterogeneity is not an easy task: user interfaces (GUIs) are essential, especially in visually oriented industries such as architecture, construction and facility management. To complement existing applications for *creating* domain-specific datasets, this dissertation also formulates the foundations of an ecosystem for modular GUIs, to enable these heterogeneous datasets to be gradually *linked* together in a project-specific sequence. Not unlike the ConSolid ecosystem, this ecosystem for GUIs is based on federated modules, each with its own specialisation. These modules can be brought together in specific configurations to form a well-defined GUI, which is tailor-made for the activity at hand and the available datasets. Since this ecosystem for interfaces is based on the relatively novel concept of micro-frontends, and the result is a federated catalogue (or 'store'), it bears the name Mifesto (Micro-Frontend Store).

Although ConSolid and Mifesto can function independently of each other, the full potential of either will only be achieved in their combination. In ConSolid, any data type can be stored and connected into a larger project. Mifesto allows a human user to contribute in a user-friendly way to the contents of a project with such vast extents.

Samenvatting (Nederlands)

Informatie gerelateerd aan de gebouwde omgeving wordt geproduceerd door een veelheid aan actoren, waarvan velen slechts af en toe met elkaar interageren. Naast een kernteam bestaande uit een opdrachtgever, een architect en projectingenieurs, omvat een bouwproject vaak bijdragen van een uitgebreider netwerk van aannemers, onderaannemers, adviesbureaus etc. In latere fases worden facilitair managers, eigenaren en bewoners toegevoegd aan dit netwerk, evenals externe instanties zoals overheidsinstellingen, onderzoeksinstituten, landmeters en infrastructuurspecialisten. Elk project heeft dus meerdere partners, en elke partner kan tegelijkertijd aan meerdere projecten werken. Deze overlappende netwerken van actoren en projecten maken het mogelijk noch wenselijk om 'alle' beschikbare informatie op een digitaal platform te centraliseren. Desalniettemin wordt een gecentraliseerd 'Common Data Environment' (CDE) vaak gezien als de enige oplossing voor het beheer van informatie met betrekking tot de gebouwde omgeving: een CDE is een webplatform voor de integratie van projectinformatie. Vaak is de enige manier om toegang te krijgen tot informatie op een CDE via een propriëtaire interface, meestal verstrekt door een commercieel softwarebedrijf dat niet bij het project betrokken is.

Een CDE wordt per project ingezet, wat betekent dat een kantoor mogelijk meerdere CDE's tegelijk moet onderhouden. Daarnaast wordt projectdata zelf (vaak een BIM-model) vaak beschreven in beschermde formaten, waardoor het niet eenvoudig is om de inhoud te koppelen met andere projectdata (bv. foto's, puntenwolken en planning), of herbruikbare informatie te vinden uit aangrenzende domeinen (bv. GIS, schadeonderzoek, gebruikersgegevens, historische datasets) of andere fases uit de gebouwlevenscyclus. Zelfs met open, gestandaardiseerde formaten zoals IFC (Industry Foundation Classes), is het koppelen met andere gegevens niet altijd evident, vanwege de bestaande hiërarchie tussen het 3D BIM-model enerzijds en de gekoppelde documenten anderzijds.

Dit proefschrift verkent een alternatieve piste, die begint bij het gedecentraliseerde, multidisciplinaire en heterogene karakter van de gebouwde omgeving. De basis voor een 'gefedereerde CDE' is een infrastructuur die gebruikmaakt van het wereldwijde web als een veilig en schaalbaar raamwerk voor het opslaan van privacygevoelige, onderling verbonden gegevens. Een 'gefedereerd project' is in deze context een aggregatie van projectspecifieke en

contextuele datasets, die – afhankelijk van de interactie – specifieke datasets kunnen omvatten of uitsluiten. In veel projecten zullen deze aggregaties heterogeen van aard zijn, en bijvoorbeeld bestaan uit BIM-modellen, afbeeldingen, puntenwolken, spreadsheets, regelgevingsdatasets, etc. Bijgevolg is het nodig om deze datasets met elkaar te verbinden op een manier die onafhankelijk is van de gebruikte data- en bestandstypes, teneinde een overkoepelende informatiecatalogus te vormen. De voorwaarden voor een dergelijke infrastructuur worden in dit proefschrift geïdentificeerd, evenals de huidige technologieën die aan deze voorwaarden kunnen voldoen. Daarnaast wordt een technologische implementatie uitgewerkt die de algehele haalbaarheid van de aanpak illustreert, terwijl tegelijkertijd aangetoond wordt hoe compatibiliteit met bestaande BIM-normen grotendeels behouden kan blijven. Dit gebeurt o.a. door projectinformatie strikt te scheiden van metadata. Waar projectdatasets de daadwerkelijke informatie over de gebouwde omgeving bevatten, geeft de metadata de nodige context voor identificatie van geschikte digitale hulpmiddelen die interactie met bepaalde projectdatasets kunnen faciliteren. Dit laat toe om de ontwikkelde datapatronen te gebruiken voor andere doeleinden dan het documenteren van de gebouwde omgeving. Aangezien het formaat of schema van projectgegevens niet uitmaakt, kunnen verbindingen tussen zeer uiteenlopende documenten gemaakt worden. Het ene document kan bijvoorbeeld een 3D-representatie van een specifiek object bevatten, een ander een foto en nog een ander een semantische beschrijving. Het resultaat is een interdisciplinaire catalogus van bouwgegevens die schaalbaar is over het web – een gefedereerd 'multimodel'. Dit ecosysteem voor gedecentraliseerd databeheer op het web wordt verder aangeduid als ConSolid. ConSolid is gebaseerd op het Semantisch Web, het Solid-protocol voor gedecentraliseerde datakluizen ('Pods') en de FAIR-principes (Findable, Accessible, Interoperable, Reusable) voor webgebaseerd databeheer. Waar dit nodig of wenselijk geacht wordt, worden extensies voorgesteld om het Solid-ecosysteem te kunnen inzetten als basis voor een gefedereerde CDE, zoals een SPARQL endpoint over een gehele datakluis en een uitbreidbare metadatastructuur gebaseerd op de Data Catalog (DCAT) ontologie.

Deze uitgebreide infrastructuur maakt verschillende bijkomende niveaus van gegevensbeheerprocessen mogelijk. Een eerste voorbeeld in dit proefschrift is de ontwikkeling van datapatronen om metadata te toetsen aan projectspecifieke vereisten. Ten tweede wordt een benadering voor een uitgebreide toegangscontroleomgeving bedacht, waarbij de semantische eigenschappen van zowel bezoekers als datasets kunnen worden gevalideerd, om te bepalen

of toegang tot deze dataset verleend kan worden. Bijvoorbeeld: 'iedereen die kan bewijzen dat ze werken voor het architectuurbureau van dit project, krijgt toegang tot de datasets waaraan een 'gedeeld' label toegekend werd'.

Om ConSolid echter effectief als een CDE te gebruiken, is het noodzakelijk om de generieke opslaginfrastructuur, die bestaat uit Solid-datakluizen, uit te breiden met een service-infrastructuur. Primair laat deze service-infrastructuur toe om de datapatronen in ConSolid te interpreteren, de benodigde gegevensfragmenten te aggregeren, en deze om te vormen tot een output die voldoet aan bestaande, gestandaardiseerde en domeinspecifieke formaten. Met deze service-infrastructuur kan een gefedereerd ecosysteem toch geïntegreerd worden in de huidige digitale omgevingen die ervan uitgaan dat de gegevens beschikbaar zijn via gecentraliseerde eindpunten (bv. BIM-authoring tools).

Echter, dergelijke 'desktopomgevingen' bieden zelden toegang tot de expressiviteit die eigen is aan (Webgebaseerde) multimodellen. Zoals eerder aangegeven, is het – per definitie – onzeker welke dataformaten een dergelijk multimodel zal bevatten. Bijgevolg is menselijke interactie met zoveel heterogeniteit geen gemakkelijke taak: gebruikersinterfaces (GUI's) zijn essentieel, vooral in visueel georiënteerde industrieën zoals architectuur, constructie en facilitair beheer. Als aanvulling op bestaande applicaties voor het *creëren* van domeinspecifieke datasets, formuleert dit proefschrift dan ook de fundamenten van een ecosysteem voor modulaire GUI's, om deze heterogene datasets aan elkaar te kunnen *linken* op een manier die steek houdt voor elk project afzonderlijk. Net zoals ConSolid is dit ecosysteem voor GUI's eveneens gebaseerd op gefedereerde modules, elk met een eigen specialisatie. Deze modules worden in specifieke configuraties samengebracht om een goed gedefinieerde GUI te vormen, afhankelijk van de ophanden zijnde activiteit en de datasets waarmee de interactie zal plaatsvinden. Aangezien dit ecosysteem voor interfaces gebaseerd is op het relatief nieuwe concept van micro-frontends, en het resultaat een gefedereerde catalogus (of 'store') is, draagt het de naam Mifesto (Micro-Frontend Store).

Hoewel ConSolid en Mifesto onafhankelijk van elkaar kunnen functioneren, wordt het volle potentieel van beide pas bereikt in hun combinatie. ConSolid laat toe om nagenoeg elk datatype op te slaan en te verbinden in een groter project; met Mifesto kan een menselijke gebruiker op een grafische manier bijdragen aan de inhoud van zo'n project.

Zusammenfassung (Deutsch)

Informationen über die gebaute Umgebung werden von einer Vielzahl von Akteuren produziert, von denen viele nur gelegentlich miteinander kommunizieren. Neben einem Kernteam aus Bauherr, Architekt und Projektingenieuren besteht ein Bauprojekt oft aus einem größeren Netzwerk von Auftragnehmern, Subauftragnehmern, Beratungsfirmen und weiteren. In einer späteren Phase kommen Gebäudeverwalter, Eigentümer und Bewohner hinzu, sowie externe Parteien, wie beispielsweise verschiedene Behörden, Forschungsinstitute, Vermessungsingenieure und Infrastrukturspezialisten. Jedes Projekt hat also mehrere Partner, und jeder Partner kann auch an mehreren Projekten gleichzeitig beteiligt sein. Dieses Netzwerk von Akteuren und Projekten macht es weder möglich noch wünschenswert, alle verfügbaren Informationen auf einer digitalen Plattform zu zentralisieren. Dennoch wird eine zentralisierte 'Common Data Environment' (CDE) oft als die einzige Lösung für das Informationsmanagement im Zusammenhang mit der gebauten Umgebung angesehen: eine einzige Webplattform zur Integration von Projektinformationen. Der Zugang zu Informationen in einer CDE erfolgt häufig über eine proprietäre Schnittstelle, die in der Regel von einem nicht am Projekt beteiligten Softwareunternehmen bereitgestellt wird.

Eine CDE wird pro Projekt eingesetzt, was bedeutet, dass ein Büro möglicherweise mehrere CDEs gleichzeitig unterhalten muss. Außerdem sind die Projektdaten selbst (oft ein BIM-Modell) häufig in einem proprietären Format beschrieben, so dass es nicht einfach ist, sie mit anderen Projektdaten (Fotos, Punktwolken, Planung) zu verknüpfen und wiederverwendbare Informationen aus anderen Phasen zu finden. Selbst bei offenen, standardisierten Formaten wie IFC (Industry Foundation Classes) ist die Verknüpfung mit anderen Daten aufgrund der bestehenden Hierarchie zwischen dem 3D-BIM-Modell auf der einen Seite und den verknüpften Dokumenten auf der anderen Seite nicht immer offensichtlich.

Diese Arbeit erforscht einen alternativen Weg, ausgehend von der dezentralen, multidisziplinären und heterogenen Natur der gebauten Umgebung. Die Grundlage für eine 'föderierte CDE' ist eine Infrastruktur, die das World Wide Web als sicheren und skalierbaren Rahmen für die Speicherung sensibler, vernetzter Daten nutzt. Ein 'föderiertes Projekt' ist dann eine Aggregation von

projektspezifischen und kontextbezogenen Datensätzen, die je nach Aufgabe spezifische Datensätze einbeziehen oder ausschließen können. Diese Aggregationen werden häufig heterogen sein, z. B. bestehend aus BIM-Modellen, Bildmaterial, Punktwolken, Tabellen, regulatorischen Datensätzen usw. Dennoch müssen sie miteinander verbunden sein, um einen übergeordneten Informationskatalog zu bilden, unabhängig von den verwendeten Medientypen. Die Voraussetzungen für eine solche Infrastruktur werden in dieser Arbeit identifiziert, ebenso wie aktuelle Technologien, die diese Voraussetzungen erfüllen können. Darüber hinaus wird eine technologische Implementierung erarbeitet, die die Gesamtdurchführbarkeit des Ansatzes veranschaulicht und zeigt, wie die Kompatibilität mit bestehenden BIM-Standards weitgehend durch die strikte Trennung von Projektdaten und Metadaten erhalten werden kann. Während Projektdatensätze die eigentlichen Informationen über die gebaute Umgebung enthalten, bieten Metadatensätze den notwendigen Kontext, um geeignete Werkzeuge für die Interaktion mit einem Datensatz zu identifizieren. Die Grundlagen des Ökosystems werden domänenunabhängig sein. Das Format oder Schema der Projektdaten spielt also keine Rolle, und die entwickelten Datenmuster können für andere Zwecke als die Dokumentation der bebauten Umgebung verwendet werden. Das bedeutet, dass Verbindungen zwischen verschiedenen Dokumenten hergestellt werden können. Beispielsweise kann ein Dokument eine 3D-Darstellung eines spezifischen Objekts enthalten, ein anderes ein Foto und wieder ein anderes eine semantische Beschreibung. Das Ergebnis ist ein interdisziplinärer Katalog von Gebäudedaten, der über das Web skalierbar ist – ein föderiertes 'Multi-Modell'.

In dieser Arbeit wird das entwickelte Ökosystem für dezentrale Datenverwaltung im Web als ConSolid bezeichnet. ConSolid basiert auf dem semantischen Web, dem Solid-Protokoll für dezentrale Datentresore ('Pods') und den FAIR-Prinzipien (Findable, Accessible, Interoperable, Reusable) für die webbasierte Datenverwaltung. Wo es als notwendig erachtet wird, werden Erweiterungen vorgeschlagen, um das Solid-Ökosystem als Grundlage für ein föderiertes CDE einzusetzen, wie z. B. ein SPARQL-Endpunkt über einen gesamten Datentresor und eine erweiterbare Metadatenstruktur, die auf der Data Catalog (DCAT) Ontologie basiert.

Diese erweiterte Infrastruktur ermöglicht verschiedene übergeordnete Datenverwaltungsprozesse. Erstens werden Datenmuster vorgeschlagen, um projektspezifische Anforderungen an Metadatensätze zu erstellen, zuzuweisen und zu validieren. Zweitens wird ein Ansatz für eine erweiterte Zugriffskontrollumgebung entwickelt, bei der semantische Eigenschaften sowohl von Besuchern

als auch von Ressourcen validiert werden können, um zu bestimmen, ob der Zugriff auf diese Ressource gewährt werden kann. Zum Beispiel: "Jeder, der nachweisen kann, dass er für das Architekturbüro dieses Projekts arbeitet, erhält Zugang zu den Datensätzen, denen ein 'Gemeinsam'-Etikett zugewiesen wurde".

Um ConSolid jedoch effektiv als CDE nutzen zu können, ist zusätzlich zu dieser allgemeinen Storagelayer eine Servicelayer erforderlich. Im Kern ermöglicht diese Servicelayer, die vorgeschlagenen Datenmuster zu interpretieren, die notwendigen Datenfragmente zu aggregieren und sie an eine Ausgabe anzupassen, die mit bestehenden, standardisierten, domänenspezifischen Formaten konform ist. Mit dieser Servicelayer kann eine föderierte Infrastruktur dennoch in vertrauten Branchenumgebungen verwendet werden, die erwarten, dass die Daten über zentralisierte Endpunkte verfügbar sind (z. B. BIM-Autorentools).

Jedoch bieten solche Desktop-Umgebungen selten Zugang zur Ausdrucksfähigkeit von (Web-basierten) Multi-Modellen. Wie bereits erwähnt, ist es – per Definition – ungewiss, welche Dateiformate ein solcher Katalog enthalten wird. Folglich ist die menschliche Interaktion bei so viel Heterogenität keine leichte Aufgabe: Benutzeroberflächen (GUIs) sind unerlässlich, besonders in visuell orientierten Branchen wie Architektur, Bauwesen und Gebäudeverwaltung. Um bestehende Anwendungen für *erstellende* domänenspezifische Datensätze zu ergänzen, formuliert diese Arbeit auch die Grundlagen eines Ökosystems für modulare GUIs, um diese heterogenen Datensätze allmählich in einer projektspezifischen Abfolge *miteinander zu verknüpfen*. Wie das ConSolid-Ökosystem basiert auch dieses Ökosystem für GUIs auf föderierten Modulen, von denen jedes seine eigene Spezialisierung hat. Diese Module werden in spezifischen Konfigurationen zusammengeführt, um eine gut definierte GUI zu bilden, abhängig von der anstehenden Aktivität und den Datensätzen, mit denen interagiert werden soll. Da dieses Schnittstellen-Ökosystem auf dem relativ neuen Konzept der Micro-Frontends basiert und das Ergebnis ein föderierter Katalog (oder 'Store') ist, trägt es den Namen Mifesto (Micro-Frontend Store).

Obwohl ConSolid und Mifesto unabhängig voneinander funktionieren können, wird das volle Potenzial beider erst in ihrer Kombination erreicht: In ConSolid können beliebige Datentypen gespeichert und in ein größeres Projekt integriert werden - in Mifesto kann ein menschlicher Benutzer auf grafische Weise zum Inhalt eines solchen Projekts beitragen.

List of Acronyms

AAA Anyone can say Anything about Anything.

ABAC Attribute-Based Access Control.

ACL Access Control List.

ACP Access Control Policy.

AEC Architecture, Engineering and Construction.

AECO Architecture, Engineering, Construction and Operations.

API Application Programming Interface.

BCF BIM Collaboration Format.

BEO BuildingElement Ontology.

BIM Building Information Modelling.

BLC building life cycle.

BOT Building Topology Ontology.

CAD Computer Aided Design.

CAM Computer Aided Manufacturing.

CDE Common Data Environment.

CSS Community Solid Server.

DCAT Data CATalog vocabulary.

DEO DistributionElement Ontology.

DGFB Directie Gebouwen en Facilitair Beheer.

DID Decentralised Identifier.

DT Digital Twin.

FAIR Findable, Accessible, Interoperable and Reusable.

FM Facility Management.

FOG File Ontology for Geometry Formats.

GIS Geographic Information Systems.

GPU Graphics Processing Unit.

GUI Graphical User Interface.

HBIM Heritage Building Information Modelling.

HTTP HyperText Transfer Protocol.

IAI Industry Alliance for Interoperability.

ICDD Information Container for Linked Document Delivery.

IDP Identity Provider.

IFC Industry Foundation Classes.

IP Intellectual Property.

JSON JavaScript Object Notation.

JWT JSON Web Token.

KG Knowledge Graph.

LBD Linked Building Data.

LBD CG Linked Building Data Community Group.

LDN Linked Data Notification.

LDP Linked Data Platform.

LOD Level of Detail.

LTQP Link Traversal Query Processing.

NLP Natural Language Processing.

OIDC OpenID Connect.

OMG Ontology for Managing Geometry.

OWL Web Ontology Language.

PBAC Pattern-based Access Control.

RBAC Role-Based Access Control.

RDF Resource Description Framework.

RDFS RDF Schema.

SHACL SHApes Constraint Language.

SPA Single-Page Web Application.

SPARQL SPARQL Protocol And RDF Query Language.

SSoI Single Source of Information.

SW Semantic Web.

TRL Technology Readiness Level.

Turtle Terse RDF Triple Language.

URI Uniform Resource Identifier.

URL Uniform Resource Locator.

VC Verifiable Credentials.

VDR Verifiable Data Registry.

W3C World Wide Web Consortium.

WAC Web Access Control.

WADM Web Annotation Data Model.

WAV Web Annotation Vocabulary.

Chapter 1
Introduction

1.1 Research Context

The built environment plays a fundamental role in shaping society. It is the backdrop for all daily activities, and there are only few disciplines that are not affected by their surroundings. Architects, urbanists and policy makers have the task to carefully plan this physical ecosystem in which we spend our lives; an ecosystem that acts on small- and medium-scale infrastructure (rooms, buildings) as well as on large (cities) to very large scale (countries and bigger). This influence is bidirectional: every design envisions a future for the built environment to make a particular place more fit for one or more specific purposes. In order to accomplish this, a design needs to integrate a multitude of widely divergent visions and information streams.

In the current era of digitisation, many of these information streams will have digital roots, and consequently they must be structured in a specific way, according to the conventions used in their respective domains. For example, there are other needs and agreements for geospatial data than for historical or heritage documentation, governmental regulations, or building physics. As in a modern society, these streams will be published by different actors and organisations, the result can be seen as a federated, heterogeneous 'web' of context, indeed facilitated by the *World Wide Web*. In this information structure, specific knowledge can be either contextual or principal 'project' information, depending on the task at hand.

So far the theory. In reality, if there are any connections between these disparate information streams, they are all too often only present in a local, semi-structured fashion – if not purely in the head of the designer. Hence, these links are not reusable and do not support incremental data flow and creation. Rather, information remains siloed both in terms of data structures (e.g., because of proprietary formats) and in terms of data environments, which will often impede information exchange with agents outside the ecosystem ('walled gardens'). However, there are few industries where information needs to be exchanged so intensively with so many actors [25]: not only during the design and construction phases (architects, commissioner, engineers, contractors) but also during the eventual operation phase, which (in most cases –

luckily) constitutes the bulk of the building life cycle. This misalignment often leads to information duplication and ambiguities, which can be costly – not only because of the time spent in recreating information that already exists somewhere, in some form, but also as they may lead to on-site issues that must be resolved prior to proceeding the work. Therefore, an approach must be devised to discover and connect these silos while leaving their publishers the freedom to choose the dataset's desired content, schema, location and access rules. Such approach will not only increase economic gains by reducing these ambiguities, but will also open up several opportunities for cross-disciplinary knowledge exchange. Let us take the example of the renovation of a specific heritage monument. During the design phase, the design can be checked against regulations published by the local authorities. Naturally, a digital model aids in planning and executing the renovation. Afterwards, the created model and its geometry can be used as a basis for an 'as-built' model, serving as an input for a Digital Twin (DT) or a virtual museum. More exotic cases can be linked as well, such as combining the topology and program of the building with very specific access control rules, or optimising rooms for the individual preferences of a building's inhabitants.

In this dissertation, a theoretical basis is laid out for a Web-based ecosystem to discover, link, semantically enrich, validate and interact with such heterogeneous, interdisciplinary and federated knowledge. Although the design patterns for this ecosystem will be general-purpose, their primary application will be to digitally document the built environment. Therefore, the above can also be phrased as follows: *to use the Web as a federated Common Data Environment (CDE)*. Taking well-established concepts such as CDEs and Building Information Modelling (BIM) [52, 22] as starting points, the need to further embed the Architecture, Engineering, Construction and Operations (AECO) industries in the broader 'digital economy' [35] is discussed. This dissertation will stepwise explore design principles and data patterns to provide a comprehensive answer to the challenges that lie ahead, both theoretically and practically, applying existing technologies. The resulting framework will support the federation federation of data vaults, open data repositories, services and GUI modules (Figure 1.1).

To achieve this, the framework will strongly base upon general Web standards and technologies, i.e., concepts for federated data management, aggregation, domain-agnostic data modelling and interaction with a heterogeneous set of documents. Three independent layers will be devised to structure the framework. In the following paragraphs, these layers will be briefly discussed.

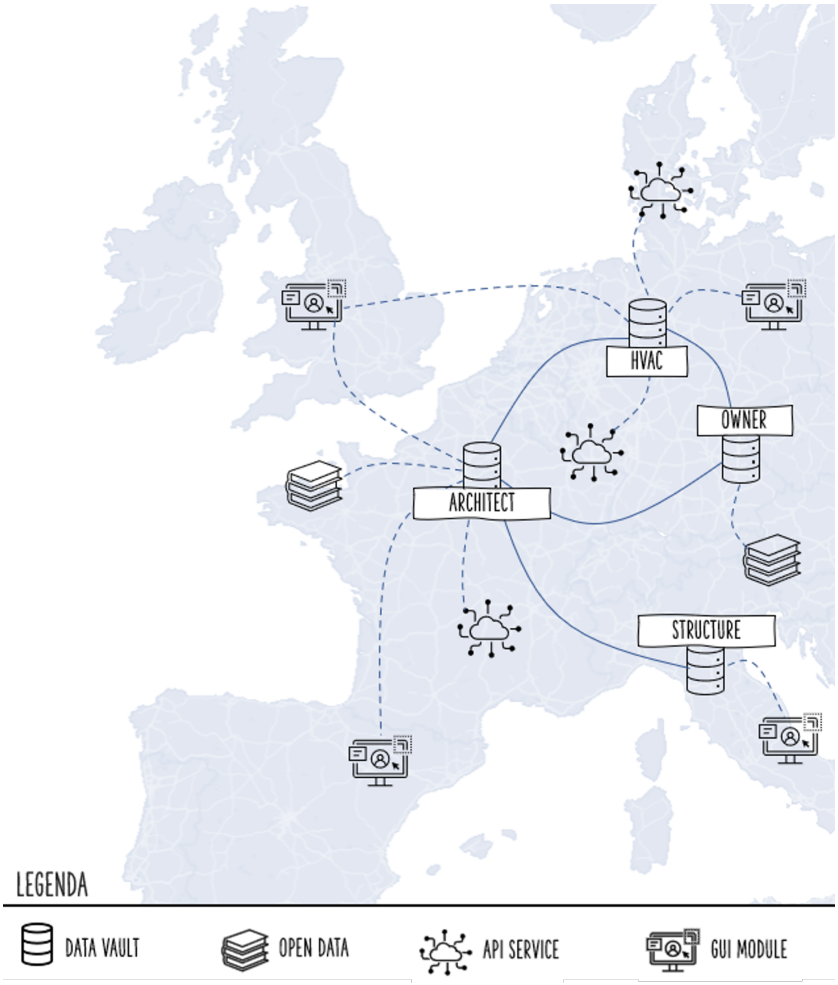

Figure 1.1: The modular ecosystem supports federation of data vaults, GUI modules, functional services and open data repositories.

Before any existing information can be used, it must be clear where and how to retrieve it. Access control policies must be in place to ensure protection of sensitive data. The first layer therefore considers federated data storage, authentication and authorisation, including the notion of secure data vaults for project stakeholders. The meaning of these vaults for the AECO industries and their potential to support a *federated* CDE are clarified in the dissertation. Then, when a dataset can be discovered, it must be assessed whether the dataset has any potential to support decision-making for said scenario.

When rich metadata structures are present, this assessment might be done prior to loading the actual dataset(s), in a filtering phase. Guidelines for such rich metadata structures will be inspired by the FAIR (Findable, Accessible, Interoperable and Reusable) principles [200]. As a technological foundation for the data ecosystem, this dissertation bases upon Semantic Web (SW) technologies and the specifications and standards that together form the Solid ecosystem [111]. However, whenever deemed necessary, custom extensions to the Solid ecosystem will be proposed, in order to achieve the desired functionality. Related to Figure 1.1, this layer deals with 'data vaults' and 'open data (repositories)'.

Simultaneously, it must be checked if there are any available services capable of interpreting the dataset's schema. Naturally, these services must be oriented towards executing (part of) the intended scenario – to know if any interaction between the data and the end user is possible at all. The second layer thus facilitates (1) high-level functional interaction with the data vaults, (2) knowledge aggregation and (3) knowledge adaptation. This layer will also facilitate the ecosystem's compatibility with existing industry standards and the organisational structures of present-day CDEs. Related to Figure 1.1, this layer deals with 'services'.

Finally, the third layer deals with the *cross-linking* of heterogeneous, federated information sets by domain specialists, in addition to existing practices of data *creation*. In frequently occurring, discipline-specific situations, common data formats will be used, so a large part of the potential interactions can be covered by existing practices for developing Web applications. For example, in a phase where project data is restricted to BIM models using the open IFC-format, a Single-Page Web Application (SPA) with a 3D viewer and some panels for visualising properties will be well-suited for the task. However, a 'project' can also be considered as a heterogeneous collection of interconnected datasets which gradually develops throughout time – without any disciplinary boundaries applied. As a consequence, there are no restrictions to the mediatypes of a project's datasets. This will be the approach taken in this dissertation. Although this is a powerful and versatile approach from the perspective of data modelling, it is less clear how users can interact with such heterogeneity. After all, the order of adding knowledge to such catalogue will vary depending on the project. Maybe the starting point is formed by just a few images, a damage survey or a geospatial location, from which gradually the whole catalogue will be built – only adding what is needed for a specific task, but always contributing to the overall Web of data. The third layer therefore considers the build-up of

such a heterogeneous multi-model in a visual way, from any starting point and project planning, using any initial sources or semantic knowledge. Its focus lies on the interaction between a human end-user and the digital knowledge base. To broaden the scope of available tools and interactions, the concept of SPAs is extended with the dynamic aggregation of decentrally published, modular end-user applications. The relatively novel concept of micro-frontends and, again, Web federation technologies, will form the basis to achieve this. Using modules as building blocks to interact with multi-models, domain experts will be able to configure an interface based on the intended activity and the already available data, without extensive programming knowledge. Related to Figure 1.1, this layer deals with 'GUI modules'.

1.2 Assumptions and Research Questions

This dissertation contains an overview of current-day challenges and opportunities for a digital, federated built environment, as an interconnected subset of the global Web of data. Paradoxically, the sheer amount of related subdisciplines related to the built environment, combined with the decentral nature of contributions made by companies and institutions of varying size and trade, implies the need for a federated, modular, *domain-agnostic* approach for organising interrelated data on the Web, and a flexible, user-friendly way of interacting with such heterogeneous data. These topics will be further elaborated in Chapter 2, which will provide the rationale for the following assumptions:

- **A 1:** *To determine a 'complete' data model for the built environment is impossible, as the disciplinary boundaries are not clear. By gradually combining different data models in an over-arching data catalogue, however, the relevant domains can be nevertheless covered in a case-specific way.*
- **A 2:** *Considering the entire life cycle of a digital built asset with wide usage scope, a federated approach provides a more comprehensive 'Single Source of Information' [84] than common centralised solutions – as it is not restricted to a single data provider. However, a higher effort is needed to keep the knowledge base consistent.*
- **A 3:** *A federated CDE better guarantees the stakeholder's ownership of data than centralised solutions, since the location of the data can be freely chosen and a separation of data and services is possible. Additionally, the use of open data formats allows to bypass the need for proprietary APIs.*

Based on these assumptions, the research questions for this dissertation can now be defined. All research questions contribute to the larger goal of demonstrating the feasibility, advantages and challenges of a federated CDE.

- **RQ 1:** *What are the technology-agnostic characteristics for a scalable CDE with high potential for discovery of related information, and integration and reuse of existing data sources?*
- **RQ 2:** *How can these characteristics be addressed using current-day standards, Web engineering concepts and technology specifications?*
- **RQ 3:** *Using the technologies mentioned in RQ2, what are data patterns for structuring, discovering and querying information in a federated environment?*
- **RQ 4:** *What are data patterns for mediatype-agnostic, cross-resource linking and annotation in a federated environment?*
- **RQ 5:** *Is such environment compatible with current-day information management practice in the AECO sector?*
- **RQ 6:** *How can a domain-specialist without extensive IT knowledge link new datasets and their content to an existing federated project catalogue, independent from the media type or present topics of both the new datasets and the existing multi-model?*

The *scalability* mentioned in RQ1 relates to the fact that the ecosystem's boundaries will be per definition unknown upon initialisation. A project must be dynamically extensible both regarding the location of its constituent data sources (federation) and their content-types (heterogeneity). This contrasts with the classic use of a single BIM model or a centralised CDE solution with fixed media types. In such federated context, maximising *discovery potential* of the ecosystem should work in two ways. The first one is a top-down approach, where a single access point for the project can be used to dynamically retrieve the location and metadata of all (or a predefined subset) of its constituent resources. The second one is 'bottom-up', i.e., to start from the content of a particular data source (for example, through the selection of a specific 3D element) and then traverse the project to find related project sources that also contain information about the selected element.

It should be noted that, to the author's knowledge, there is no agreed-upon definition of a 'federated CDE'. The argumentation that will lead to the identification of the *characteristics* an eventual technological solution must adhere

to (RQ1), therefore also explains how this concept is interpreted in this dissertation.

The term 'data patterns' (RQ3) will then be used to indicate a suggested mapping of specific (Linked Data) properties and classes to address a particular characteristic of the ecosystem. Data patterns may then be chains of properties ('property paths') between two entities or more complex graph-like structures that connect various information snippets. For example, in this thesis' interpretation and implementation of the characteristics (ConSolid), the above-mentioned top-down and bottom-up approaches for data discovery within a project will both make use of specific data patterns. Note that data patterns are never exclusive solutions – there will always be alternative approaches to achieve the same goal of connecting multiple information snippets; the patterns provided in this dissertation are thus to be considered exemplary, rather than exclusive.

1.3 Dissertation Outline

Addressing the research questions will largely coincide with the structure of this dissertation. Before handling them, however, a more in-depth introduction and rationale is required, which is the topic of Chapter 2 (*Background and Rationale*). This chapter will maintain a technology-agnostic narrative and identify the built environment as a domain that incorporates many challenges that are also present in other industries. Amongst these cross-industry challenges are the fact that the built environment is the result of contributions from federated consortia of small- to medium-sized enterprises, the existence of unique 'end products', the need for cross-discipline data exchange and continuous monitoring of the asset long after its completion.

Subsequent chapters will each cover a particular part of the ecosystem, from technology-agnostic characteristics over identification of suitable technologies and proof-of-concept implementation to an AECO-related case study. The reasoning behind the main requirements (RQ1) related to structuring, discovery and filtering will be addressed in Chapter 3 (*Storage and Discovery of Federated Projects*), as well as the necessary technologies (RQ2) and the data patterns (RQ3) that base upon these technologies. The topics related to connecting and annotating heterogeneous information will be discussed in Chapter 4 (*Resource Linking and Annotation*), addressing RQ1, RQ3 and RQ4. Chapter 5 (*Data Validation*) and Chapter 6 (*Middleware Services*) will respectively develop validation methods and higher-level interfaces that allow the usage of the

ecosystem as a Common Data Environment, hence also relating to RQ1. Finally, the layer of the ecosystem that facilitates domain-specific end user interaction (with a focus on cross-dataset linking activities) with heterogeneous, federated data catalogues (RQ6) is developed in Chapter 7 (*Interfaces for Linking Federated Multi-Models*). The dissertation concludes with Chapter 8 (*Evaluation*) and Chapter 9 (*Conclusion*).

Figure 1.2 shows how every chapter describes a different part of the ecosystem, and their relationship to one another. When an introduction to a particular concept is deemed convenient, the reader will be referred to a dedicated appendix.

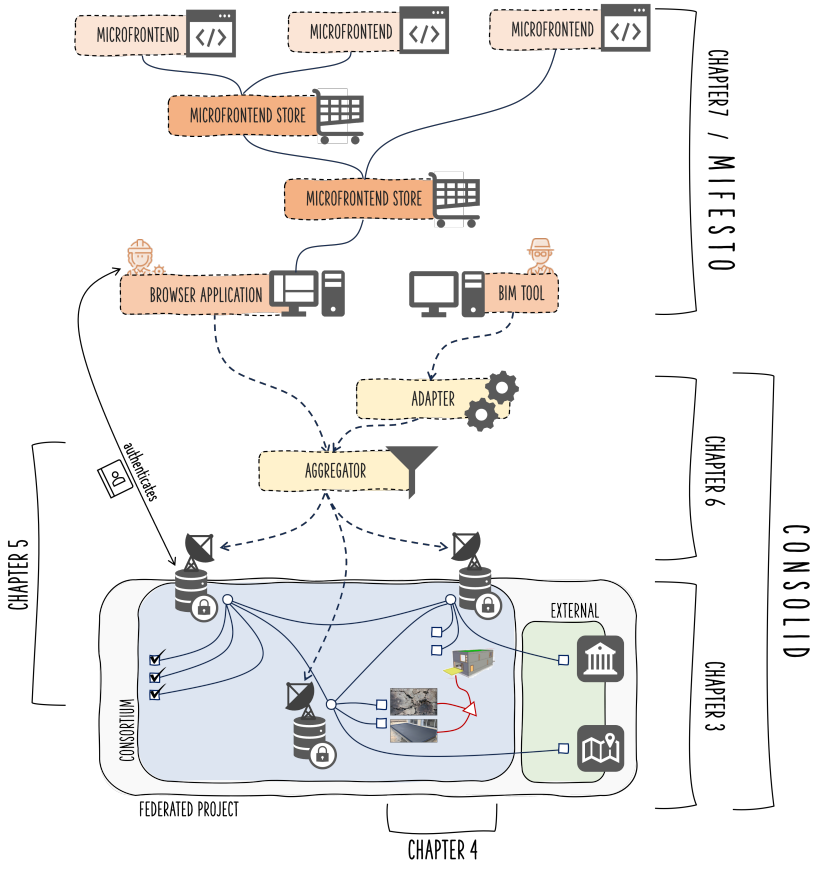

Figure 1.2: Interaction between the different topics discussed in this dissertation.

1.4 Research Approach and Limitations

Overall, this dissertation will take a modular approach to avoid too much dependency on single technologies in a rapidly changing research and technology landscape. To achieve such modularity, this research will be conducted as a continuous interplay of reasoning about the (theoretical) characteristics of the ecosystem, technological validation of those characteristics and data pattern development, identification of exceptions that could not yet be addressed and generalisation of the characteristics to make the ecosystem incorporate those exceptions (Figure 1.3).

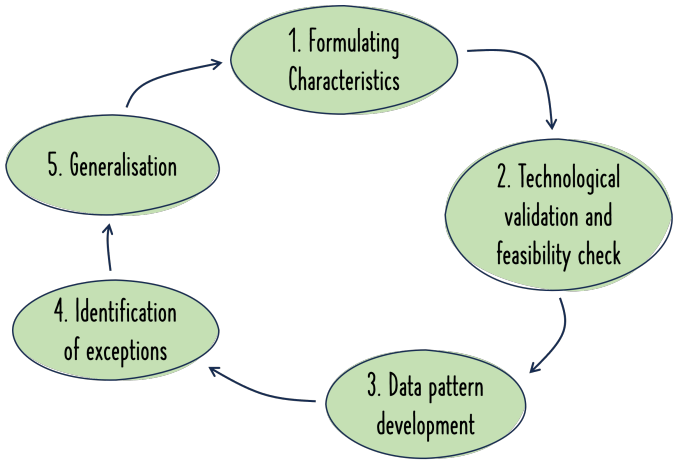

Figure 1.3: Research approach: iterations for increasing the applicability of the framework.

Practically, each of the layers discussed in Section 1.1 will be devised as conceptually independent from the others, although only in their combination the full potential of the ecosystem will be reached. This means that (1) interaction with the federated data ecosystem is possible in many ways (SPA's, micro-frontend configurations, desktop applications), and (2) that the front-end ecosystem will be able to interact with both federated multi-models and centralised multi-models (or even single model projects).

This finetuning of the ecosystem towards generally applicable and domain-independent principles is per definition dictated by the topic of Web-based documentation and linking of data about the built environment. In a Web context, it would be very counterproductive to consider building data as a separate island. On the contrary, if one 'domain' is to be chosen that can be

used in interdisciplinary contexts, it would probably be the built environment. To allow such disciplinary cross-overs without pretending to know which cross-overs will take and which ones do not, a continuous evaluation of the ecosystem's general applicability will be maintained from the early stages of the research.

This dissertation targets a broad spectrum of topics and technologies. As a consequence, the technological design choices for the setup of the federated CDE ConSolid do not imply that there is only one technological possibility to meet the requirements put forward in this dissertation. Similarly, the proof-of-concept applications and services only serve the purpose of demonstrating the feasibility of a particular module using existing technologies. In other words, to prove that the design choices and data patterns can be implemented in at least one way. Performance is not a key requirement here. However, in particular cases an indicative test will be performed to check whether the execution time lies within an acceptable order of magnitude. In a comparison to typical human-computer interactions in a classical CDE environment, an execution time around 1s is considered acceptable for core tasks of the CDE, considering the size of the demo datasets (see Section 1.7).

The main topics of each chapter are clarified with examples originating from the AECO industry. Where relevant, these examples will be related to the case study introduced in Section 1.7. It should be clear that these examples are not to be seen as 'the primary goals' of the ecosystem. Rather, they serve the purpose of illustrating the (generic) considerations and technologies devised in each chapter. Based on the practical example of damage documentation of a federated building catalogue, the reader is then trusted to assess how the ecosystem can support other specific cases as well. The different steps taken can be generalised towards more vaults, resources and domains. In the same way, it is clear that the explanatory cases for data validation (DCAT-AP, OpenCDE), pattern-based access control (Chapter 5) and compatibility with standards (ICDD and the BCF API) (Chapter 6) are illustrative, rather than being rigid end goals of the research project.

Finally, several topics will be only briefly touched upon in this dissertation, without being developed to the level of a working prototype. They remain included in the text because of their potential for future research, or because they offer alternative (non-technological) perspectives on the raised issues. Whenever this is the case, such topics will be labeled as 'out-of-scope'.

1.5 Main Contributions

The contributions of this research can be organised in different categories. The first category encompasses the contributions to the field of 'building informatics' or 'the digital built environment'. A general vision is presented for the 'BIM level 3' in the well-known BIM-levels of maturity [15]. This concept will be extended beyond what is typically understood as BIM: the dissertation will reflect on the benefits, challenges and characteristics of an ecosystem for open (asset) data, based on the FAIR principles – to use the Web as a CDE. Note that, in this case, 'open' means 'using open formats' rather than 'accessible to anyone'. This leads to the following concrete contributions *on the topic of federated multi-models*:

1. Outline of the *characteristics* for a CDE for federated, heterogeneous multi-models ;
2. Identification of existing *technologies* capable of addressing the characteristics;
3. Outline of *design choices* and *data patterns* compatible with the identified technologies;
4. Conceptual **combination** of **micro-frontends** with **semantic, federated catalogues**, and the identification of the benefits this offers.

Secondly, at the time of writing, many of the technologies and specifications used in this dissertation are still undergoing active changes. This applies in particular to the Solid ecosystem, which will be the domain-independent foundation for the organisation of access-controlled, heterogeneous multi-stakeholder projects. Whenever deemed necessary for the functionality of the framework, particular desiderata will be indicated – sometimes accompanied by a custom extension to the current stack of Solid specifications or technologies. Although these custom extensions are not part of any standardisation process at the time of writing, they are considered contributions because of the general benefits they offer compared to the 'vanilla' approach, at least in the context of a federated CDE. As a modular approach is maintained, it is expected that replacing them with equivalent technologies and data patterns devised by any future specifications, will be feasible without affecting other parts of the ecosystem. This allows to prove the technological feasibility of the developed ideas, while at the same time maintaining a bird's-eye perspective on the overall goals of the ecosystem.

The following contributions are considered in context of Solid:

1. Conceptual broadening (e.g. multi-pod collaborative environments);
2. Functional extensions (e.g. queryable union graph of a Solid data Pod, pattern-based access control, RDF aggregators);

Because of the bird-eye perspective maintained in this dissertation, no new *domain* ontologies will be proposed. However, by now, an extensive corpus of modular domain ontologies has already been published by many talented researchers, under the umbrella of the World Wide Web Consortium (W3C) Linked Building Data Community Group (LBD CG). The work in this dissertation aims to be complementary with those approaches, by primary focusing on data discovery, alignment and interaction. Some vocabularies on a (domain-agnostic) metadata and data management level will be devised to support this:

1. The ConSolid vocabulary (https://w3id.org/consolid#);
2. The PBAC vocabulary (https://w3id.org/pbac#);
3. The Mifesto vocabulary (https://w3id.org/mifesto#);

Finally, several codebases were developed in context of this research. They are to be considered mere prototypes, and are not suited for use in industrial settings. The following codebases have been published:

1. Authenticated SPARQL endpoint to Solid Pod (https://github.com/ConSolidProject/sparql-satellite/tree/dissertation);
2. Community Solid Server adaptation for mirroring RDF graphs to a SPARQL endpoint (https://github.com/LBD-Hackers/SolidCommunity_Fuseki/tree/dissertation);
3. ConSolid API and scripts (https://github.com/ConSolidProject/cde-satellite/tree/dissertation);
4. Dataset Aggregation API (https://github.com/LBD-Hackers/daapi/tree/dissertation);
5. Reference Aggregation API (https://github.com/LBD-Hackers/raapi/tree/dissertation).

1.6 Audience

This dissertation is primarily aimed towards an audience that is familiar with recent developments in building informatics and the use of Semantic Web technologies for the built environment. As the rationale starts from well-known concepts and standards, BIM and Digital Twin specialists and CDE developers are also part of its target audience – pointers will be given to more exotic concepts related to Web technologies. On the other hand, computer scientists who are dealing with Web decentralisation and knowledge aggregation will be more acquainted with concepts such as the Solid ecosystem and the Data CATalog vocabulary (DCAT) vocabulary, but might benefit from more context on AECO-related concepts. At the beginning of each chapter, relevant appendices will be indicated, providing basic context for both perspectives. For a more in-depth coverage, the reader will be referred to external sources.

1.7 Case Study

During the text, concepts will be clarified with the case study of the iGent tower of Ghent University, Zwijnaarde, Belgium (Figure 1.4). The available data from the iGent tower originates from partial BIM models, produced by two different stakeholders: Bureau Bouwtechniek (BE) and Arcadis (BE). Both stakeholders agreed to the academic usage of the models. A third stakeholder is introduced as well, namely the Department of Infrastructure and Facility Management of UGent (Directie Gebouwen en Facilitair Beheer (DGFB)), as it is a university building. Because the ecosystem is highly experimental, these stakeholders did not use the ecosystem in a real-world project, which would require a much more mature interface. However, the two stakeholders involved in the eventual enrichment case (Chapter 4), namely the DGFB and Bureau Bouwtechniek, were contacted to confirm the validity of the scenario, which involves enriching the federated project with damage record data.

The infrastructure devised in this dissertation will be based on secure data vaults, which are dereferenceable. The following (fictional) URLs will be used to refer to the stakeholder vaults in the case study:

- Bureau Bouwtechniek: https://b-b.be/data
- Arcadis: https://arcadis.com/data
- DGFB: https://dgfb.ugent.be/data

The case study will be developed throughout different chapters. Chapter 3 will describe the setup of the storage ecosystem and the metadata that allows discovery of information in the federated project. Chapter 4 will describe a sub-document linking activity between heterogeneous datasets, using the case of damage management. Damage enrichment is, being a frequently occurring activity in the operational phase of a building or heritage object [104], considered an exemplary scenario: it covers both the use of heterogeneous data sources (RDF, imagery, geometry) and sources from multiple stakeholders are involved in the process. Finally, Chapter 5 will use the stakeholder network and the project data to illustrate advanced access control mechanisms.

Figure 1.4: The iGent tower (Zwijnaarde, Belgium) will function as a case study to illustrate the topics of this dissertation.

1.8 Publications

During the 5 years of research in context of this PhD, I have authored and co-authored several peer-reviewed papers and book chapters. This dissertation aims to formulate an over-arching narrative that explains how these publications are related to one another. At the end of each chapter, the related publications will be indicated. Not all papers are equally relevant for this dissertation: some papers, especially the ones written in the early research stages, can be considered excursions rather than full-fletched contributions to the overall narrative. Others form a basis upon which subsequent papers iterate, and not seldom do these subsequent papers overrule the data patterns and conclusions of previous ones in favour of higher flexibility and robustness, or in use of established ontologies rather than custom ones. In the same way, this dissertation will revise some of the proposals in published papers into a form that is more consistent with the other aspects of the ecosystem. A complete list of accepted and peer-reviewed publications is given below.

Journal articles:

- Jeroen Werbrouck, Pieter Pauwels, Jakob Beetz, and Erik Mannens. "ConSolid: a Federated Ecosystem for Heterogeneous Multi-Stakeholder Projects". In: *Semantic Web Journal* (2023). Accepted. URL: https://biblio.ugent.be/publication/8633673/file/8633674.pdf (accessed 2024-3-18).
- Jeroen Werbrouck, Pieter Pauwels, Mathias Bonduel, Jakob Beetz, and Willem Bekers. "Scan-to-graph: Semantic enrichment of existing building geometry". In: *Automation in Construction* 119 (2020), p. 103286. URL: https://doi.org/10.1016/j.autcon.2020.103286.
- Jeroen Werbrouck, Oliver Schulz, Jyrki Oraskari, Erik Mannens, Pieter Pauwels, and Jakob Beetz. "A generic framework for federated CDEs applied to Issue Management". In: *Advanced Engineering Informatics* 58 (2023), p. 102136. URL: https://doi.org/10.1016/j.aei.2023.102136 (accessed 2024-3-18).

Book chapters:

- Jeroen Werbrouck, Madhumitha Senthilvel, and Mads Holten Rasmussen. "Federated data storage for the AEC industry". In: *Buildings and Semantics*. CRC Press, 2022, pp. 139–164.

- Anna Wagner, Mathias Bonduel, Jeroen Werbrouck, and Kris McGlinn. "Geometry and geospatial data on the web". In: *Buildings and Semantics*. CRC Press, 2022, pp. 69–99.

Conference contributions:

- Jeroen Werbrouck, Pieter Pauwels, Jakob Beetz, and Léon 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. URL: https://biblio.ugent.be/publication/8633673 (accessed 2024-3-18).

- Jeroen Werbrouck, Pieter Pauwels, Jakob Beetz, and Erik 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. URL: https://biblio.ugent.be/publication/8724183/file/8750812.pdf (accessed 2024-3-18).

- Jeroen Werbrouck, Pieter Pauwels, Jakob Beetz, and Erik Mannens. "Mapping Federated AEC projects to Industry Standards using dynamic Views". In: *10th Linked Data in Architecture and Construction Workshop*. CEUR-WS. org. 2022. URL: https://ceur-ws.org/Vol-3213/paper06.pdf (accessed 2024-3-18).

- Jeroen Werbrouck, Madhumitha Senthilvel, Jakob Beetz, Pierre Bourreau, and Léon Van Berlo. "Semantic query languages for knowledge-based web services in a construction context". In: *26th International Workshop on Intelligent Computing in Engineering, EG-ICE 2019*. Vol. 2394. 2019. URL: https://ceur-ws.org/Vol-2394/paper03.pdf (accessed 2024-3-18).

- Jeroen Werbrouck, Madhumitha Senthilvel, Jakob Beetz, and Pieter Pauwels. "A checking approach for distributed building data". In: *31st forum bauinformatik, Berlin: Universitätsverlag der TU Berlin*. 2019, pp. 173–81. URL: https://biblio.ugent.be/publication/8667508/file/8667516.pdf (accessed 2024-3-18).

- Jeroen Werbrouck, Madhumitha Senthilvel, Jakob Beetz, and Pieter Pauwels. "Querying heterogeneous linked building datasets with context-expanded graphql queries". In: *7th Linked Data in Architecture and Construction Workshop*. Vol. 2389. 2019, pp. 21–34. URL: https://biblio.ugent.be/publication/8623179/file/8623180.pdf (accessed 2024-3-18).

- Jeroen Werbrouck, Ruben Taelman, Ruben Verborgh, Pieter Pauwels, Jakob Beetz, and Erik 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. URL: https://ceur-ws.org/Vol-2636/09paper.pdf (accessed 2024-3-18).
- Pierre Bourreau, Nathalie Charbel, Jeroen Werbrouck, Madhumitha Senthilvel, Pieter Pauwels, and Jakob Beetz. "Multiple inheritance for a modular BIM". In: *Le BIM et l'évolution des pratiques: Ingénierie et architecture, enseignement et recherche* (2020), pp. 63–82. URL: https://community.osarch.org/uploads/editor/a0/1se97k6z8n3v.pdf (accessed 2024-3-18).
- Nuyts Emma, Jeroen Werbrouck, Ruben Verstraeten, and Louise Deprez. "Validation of Building Models against Legislation using SHACL". In: *LDAC2023, the 11th Linked Data in Architecture and Construction Workshop*. 2023. URL: https://linkedbuildingdata.net/ldac2023/files/papers/papers/LDAC2023_paper_8284.pdf (accessed 2024-3-18).
- Andrew Malcolm, Jeroen Werbrouck, and Pieter Pauwels. "LBD server: Visualising Building Graphs in web-based environments using semantic graphs and glTF-models". In: *Formal Methods in Architecture: Proceedings of the 5th International Symposium on Formal Methods in Architecture (5FMA), Lisbon 2020*. Springer. 2021. URL: https://doi.org/10.1007/978-3-030-57509-0_26 (accessed 2024-3-18).
- Jyrki Oraskari, Oliver Schulz, Jeroen Werbrouck, and Jakob Beetz. "Enabling Interoperable Issue Management in a Federated Building and Construction Sector". In: *EG-ICE 2022 Workshop on Intelligent Computing in Engineering*. 2022. URL: https://api.semanticscholar.org/CorpusID:250108125 (accessed 2024-3-18).
- Oliver Schulz, Jeroen Werbrouck, and Jakob Beetz. "Towards Scene Graph Descriptions for Spatial Representations in the Built Environment". In: *30th International Workshop on Intelligent Computing in Engineering, EG-ICE 2023*. 2023. URL: https://www.ucl.ac.uk/bartlett/construction/sites/bartlett_construction/files/towards_scene_graph_descriptions_for_spatial_representations_in_the_built_environment.pdf (accessed 2024-3-18).

Chapter 2
Background and Rationale

This chapter sketches the background and rationale for this research. The topics discussed in this chapter will remain largely technology-agnostic, although sometimes specific technologies will be mentioned.

2.1 The Digital Built Environment

Just like in virtually any other domain, the production and consumption of data and the usage of digital services related to construction and the built environment increased significantly over the last years. However, this adoption is taking place at a slower rate than most other industries [2]. Contrary to many domains, most 'products' in the Architecture, Engineering, Construction and Operations (AECO) industries[1] are unique: a building project lives in the real world, is designed for or adapted to a specific programme, affected by its natural surroundings and social and economic human infrastructure. From design to demolition, people with divergent backgrounds interact with this asset, ranging from direct contributors such as architects and engineers, commissioners, facility managers and inhabitants, to indirect partners such as governmental agencies and product manufacturers. As a consequence, information is stored in a highly decentralised manner, and the amount of interactions with external partners is very high compared with other industries [25]. Because most stakeholders will be involved in multiple projects at the same time, one can speak of a 'double patchwork': a many-to-many relation between industry partners and collaborative projects (Figure 2.1). Intensive collaboration and information exchange between those partners is thus crucial, and the benefits of digital technologies as an aid in this process were recognised already in the early years of digitisation.

The first commercial Computer Aided Manufacturing (CAM) software package, Pronto, came out in 1957, quickly to be followed by the notion of Computer Aided Design (CAD) (1959), in context of the MIT Computer-Aided Design

[1] This dissertation will maintain a plural form when referring to the AECO industries. At the time of writing the author does not deem it sufficient to speak of a single industry - although further integration of these industries is an active field of research – to which this dissertation wishes to contribute. Sometimes, the acronym AEC will be used, indicating a current industry state where the operations sector is typically excluded from digital activities of the other disciplines.

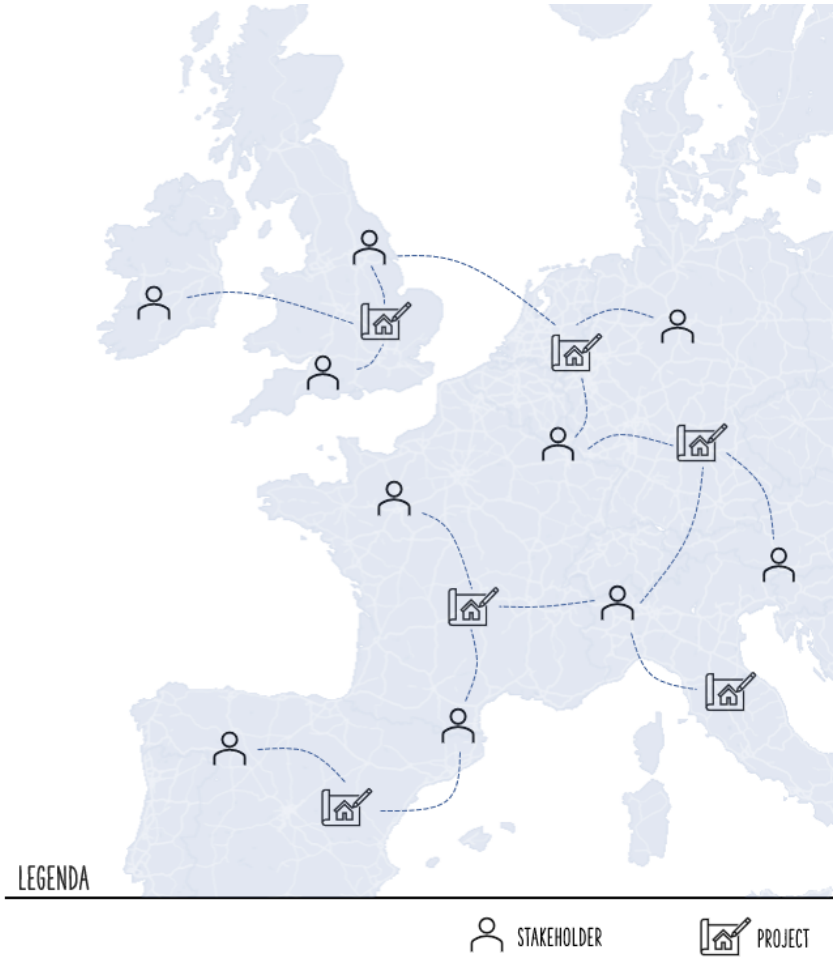

Figure 2.1: Every project counts multiple offices amongst its stakeholders, and every office participates in multiple projects simultaneously, resulting in a 'double patchwork'.

Project [147, 41]. In a visionary 1975 article, Charles Eastman notes that three-dimensional (3D) computer models can be used to generate two-dimensional (2D) drawings, theoretically eliminating design inconsistencies resulting from unsynchronised drawings [51]. Proposing to connect these 3D models with databases containing digital descriptions of elements, this vision is seen as the direct predecessor of the current-day Building Information Modelling (BIM) practice [52]. BIM has brought many benefits in information exchange and

project management for the AECO industries: its object-oriented approach allows to attach detailed information to individual building elements, concerning materials, physical properties, construction planning and more. This way, a comprehensive and semantically rich building model comes into existence, and information can be shared with whomever needs it for their respective tasks in the project. This results in, amongst others, less mistakes, less on-site collisions and better simulations – hence yielding financial, ecological, and qualitative design benefits, from the design phase to retrofitting or demolition activities [22].

Over the last decades, many CAD and BIM tools have been published, both commercially and open-source. Co-evolving with ever-more-powerful processors and Graphics Processing Units (GPU), these CAD, CAM and BIM authoring tools have led to a revolution in designing and shaping the built environment. Any CAD or BIM tool needs a geometric kernel to work with, and it will often occur that different CAD and BIM programs base upon different geometric kernels. For BIM, not only geometry, but also domain- and tool-specific data schemas, which describe the non-geometric semantics in a digital model, need to be taken into account. One can easily see that the entire software stack used in a project quickly becomes an eclectic mix of packages, most of them naturally favouring vendor- or tool-specific data-formats. Consequently, interoperability challenges will rise when information needs to be exchanged from one solution to another one. To address the ever-expanding stack of adjacent disciplines and corresponding tools, the Industry Alliance for Interoperability (IAI) was founded in 1994, to be renamed in 1996 to *International Alliance for Interoperability*. Within IAI, a non-proprietary data exchange schema for the Architecture, Engineering and Construction (AEC) industry was developed (note the absence of 'Operations'): the Industry Foundation Classes (IFC) [83]. In a re-branding effort, IAI changed its name again in 2005 to buildingSMART International. The development of IFC as an open, international standard has continued ever since.

The use of open data exchange formats such as IFC is often denoted with 'Open BIM'. The approach where many domain specialists can contribute to a shared, enduring source of information throughout the building life cycle (BLC), regardless of the software packages they use, is then labelled as '*Big* Open BIM' [134]. The Big Open BIM paradigm has been relatively successful in facilitating the connection between specialised tools involved in the design and construction phases of a built asset, centred around the IFC schema. However, in collaborative environments based on one domain-specific schema, adjacent

domains and activities still have difficulties to integrate their own (external) domain knowledge, although the ability to do so would offer multiple benefits and allow reuse of data for more scenarios. Especially since the built environment is one of the core tissues of society, cross-domain data alignment is essential for embedding its constituent subdisciplines in the broader 'digital economy'. Although the IFC schema has been extended multiple times to incorporate adjacent domains, and the standard also allows to define custom extensions, the main procedure is to draw other domains into the schema rather than allowing building-related information to be expressed in a neutral way - and be drawn as contextual information into the activities of other disciplines. This need for scalability beyond the 'design and construction' domains is acknowledged in the 2020 Technical Roadmap of buildingSMART [31], and is also reflected in the rising interest into the application of Semantic Web technologies for documenting the built environment [129].

In many cases, interdisciplinary data exchange may go far beyond the AEC domains. Firstly, the concept of 'BIM' itself can be broadened, e.g., for retrofitting purposes [91, 54], for the operational phase [138, 50, 112] or in context of heritage [179, 6]. Why not using an HBIM ('Heritage Building Information Modelling') model as a basis for a virtual museum, and connect historical events, pictures and other media fragments to the overall (geometric) model [192, 19]? Likewise, parallel to BIM, the concept of 'digital twins' [112] has been making a furore in the last decade. Digital twins refer to virtual representations of the physical building or system that allow to monitor interactions with the actual asset, or make predictions for future scenarios. These interactions between a digital infrastructure and real-world phenomena are, for example, established via sensor streams (real-digital) and actuators (digital-real). Therefore, they are mostly used during the operational phase, although the benefits of a Digital Twin system during the construction phase are documented [17]. Benefits hold on a bigger scale as well, regarding Geographic Information Systems (GIS) technologies [94, 108], city planning (smart cities) and infrastructure [178], circular economy [117] and urban mining [3]. A final example is the integration of building information with personal data such as user preferences [127].

Rather than a limited set of partial IFC models, the total information set for a given scenario will thus most likely be a heterogeneous set of resources, originating from a wide range of domains. Applied to the built environment, this set might include (structured) semantics, semi-structured geometric object descriptions and unstructured information such as imagery and point clouds. The *multi-model* approach described in [152, 63] and the ISO standard ISO

21597 Information Container for Linked Document Delivery (ICDD) [89] rely upon the idea that a collaborative construction project will always be a heterogeneous set of resources that can be interlinked, based on sub-document identifiers (Figure 2.2). *Heterogeneous* then means that no assumptions can be made on the media types of resources. In other words, whether a resource is structured or unstructured, whether it has a proprietary encoding or not and what tasks it was primarily intended for, should not influence the data patterns for creating a multi-model.

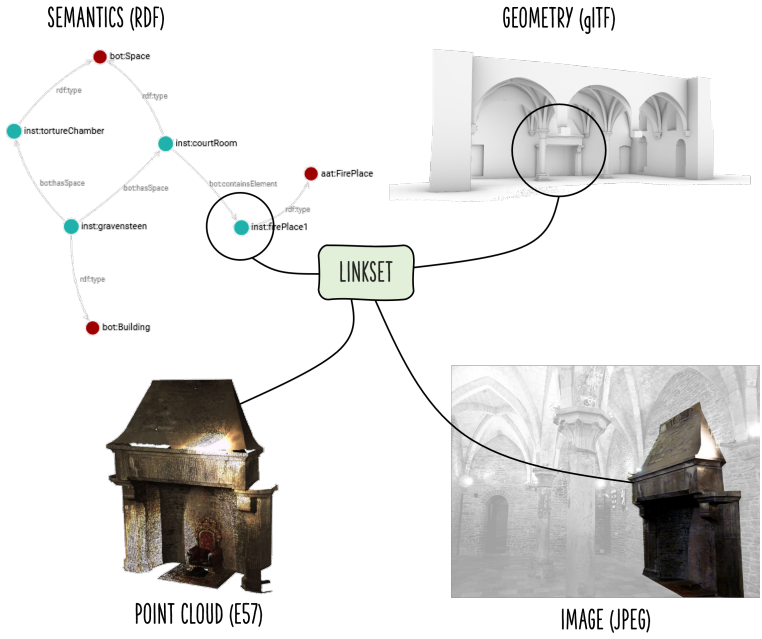

Figure 2.2: A multi-model is a container with disparate resources that can be linked on a sub-document level.

Originally, ICDD was intended for archival of project information, structured as a zipped container. However, it has been identified as a useful data structure for active interaction environments as well [157, 95]. Lately, multiple platforms have been developed to allow interaction with multi-models for construction based on ICDD, among which are the academic RUB-ICDD [64] initiative and the commercial platform Wistor [201]. Both are modular, configurable platforms which offer a number of Semantic Web tools to interact with a central multi-model via a Graphical User Interface (GUI).

2.2 Common Data Environments

In the AECO industries, a shared access point for storing project data is often called a Common Data Environment (CDE). According to ISO 19650 [84], the current international standard for BIM-based project collaboration, a CDE is an '*agreed single source of information for any given project or asset, for collecting, managing and disseminating each information container to a managed process*'. CDE solutions (mostly just called CDEs) are then platforms that provide such Single Source of Information (SSoI) [84] about an asset, used to exchange plans, (BIM) models, and communication and providing tools for cloud-based collaboration, both in multi-disciplinary context [148] and for domain-specific purposes such as HVAC design [156]. In the ideal SSoI, data consistency is preserved because it is only stored once, and each time information is used, it points to exactly this piece of data [197]. A project CDE often revolves around a single *coordination model*, which consists of domain-specific sub-models, often called *partial models*. Before a domain-specific dataset can become part of the coordination model, its quality and compliance to the project standards need to be checked [140]. However, in practice, the alignment between partial models is often limited to spatial co-location in a BIM authoring tool [162], which allows for spatial coordination activities such as clash detection, but does not involve creation and evaluation of semantic links between the constituent sub-models. A coordination model is sometimes called 'federated', as it is the union of multiple individual sub-models. Using this definition, a centralised CDE can host a federated coordination model, but this has, in fact, nothing to do with 'Web federation'. In the remainder of this work, the term *federation* will be reserved for the latter, i.e., situations where the constituent resources of a larger whole can be spread over multiple Web servers [80], rather than for describing the union of partial BIM models.

A CDE can be seen as a layered ecosystem streamlining many aspects of collaborative building projects, with at its core an access-controlled data storage system. According to Preidel et al. [141], there are no real specifications regarding the location of data or the technologies used for data storage. However, it is essential that any of the involved stakeholders can access project data anytime, from any location - which makes Web and Cloud technologies the evident foundations for such platforms. Additional requirements apply to the storage of resources in a CDE, to allow advanced data structuring and filtering, version control and document retention policies. Keeping track of metadata records on project resources is thus an essential feature of the data storage layer of a CDE. As building data often contains sensitive information that should not

be made public on the Web, authentication and authorisation protocols play an important role in governing access to project information on a CDE. This information is often role-based (e.g., 'employees of company X', 'residents of building Y') [124], but will eventually resolve to (human) agents.

Current-day CDEs are mostly proprietary ecosystems, managed by companies which oversee the complete chain of data management, ranging from developing the (often proprietary) data models, hosting the building data and user accounts on their servers and providing the authoring tools and Application Programming Interfaces (API) used to interact with this encoded data. When the data is encoded in a proprietary format, or when the possible interactions with this data are controlled by a single company, the CDE may be considered as de-facto centralised – although it may be duplicated on multiple servers to ensure data availability. In a situation where the intended usage of the digital building model is limited to a few fixed scenarios or project phases, or when all project partners are subscribed to the same CDE solution, the use of centralised platforms will provide certain benefits, among which are an integrated suite of software tools, data availability guarantees and a single helpdesk in case something goes wrong.

However, there are downsides to this dependency on project-external software companies, too. Legally, there may be ambiguities about Intellectual Property (IP) and data sovereignty ('how do I remain in control of my data and how can it be used legally'), which has led to legal disputes in the past [57]. From a technological perspective, the risk of a vendor lock-in grows with a bigger usage scope for the digital project [193]. The tight integration between project data and current CDEs is one of the reasons for two widespread industry misconceptions:

1. The (relatively) limited capabilities of a CDE, one of its tools or even a data model define the limits of what can be digitally described.
2. A SSoI is only possible in a highly centralised ecosystem.

In the following paragraphs, both misconceptions will be discussed.

2.2.1 Tools and Data

The inability to conceptually separate data and tools, i.e., the first misconception, is on the one hand related to the proprietary aspect of file-based BIM data models, and on the other hand to the 'siloed' approach of domain-specific data models. When people say 'you cannot do that in BIM', the concept of

information modelling is narrowed down to a specific data format or application, or a small set of these. This ignores the fact that, ultimately, data can be *anything* – the only necessity besides creation and storage is something (a tool, service...) capable of making sense of the content of the data (i.e., turning data into knowledge). Ideally, the ecosystem and its GUIs thus only serve as a specialised 'window' to the data. However, when there is only one single data format, which might be proprietary and owned by a CDE vendor, the GUI exposed by the CDE vendor becomes the *only* window to this data. This eventually leads towards 'walled garden' ecosystems, which have been documented numerous times in context of social media platforms [177, 53] – but the term is applicable to CDEs as well. In such situation, the limits of the platform set the limits of the digital project indeed.

To lesser extents, this also holds for domain-specific open data models, which are not intended to link with domain-external information. Although the boundaries of a domain model are set by domain specialists, their opinions may differ. Contrasting with domain-specific schema's, the Semantic Web technology stack is often acknowledged as an enabler for domain-agnostic, data-driven alignment of cross-domain information [9, 129].

2.2.2 Centralised vs. Federated SSoI

The second misconception relates to the fact that the central aspect of a CDE is often deemed necessary to maintain a SSoI. This is, again, related to the chain of faulty conversion processes that need to happen to exchange data from one proprietary file format to another one: extracting information from a document-based, proprietary model and exchanging it from one CDE to another one results in significant information losses. The reasons for this are manifold: the use of different geometric kernels, different Level of Detail (LOD), encrypted data formats, incompatible data schemas, lack of specific domain context etc. For example, during data handover phases or when archiving (i.e., 'forcing' a data exchange between potentially disparate ecosystems), a walled-garden situation does not only result in information losses, but also in the creation of parallel, unlinked and unsynchronised duplicates of information. In turn, this weakens the asset's intended SSoI because the risk for ambiguities and inconsistencies increases. This problem is not new: it has been around since the very beginning of digital collaboration in AEC with the founding of the IAI/buildingSMART (see Section 2.1). Almost 30 years later, upcoming initiatives such as the OpenCDE Foundation API [79] aim to define a minimal set of agreed-upon API standards to facilitate better communication between

CDEs, but do not yet facilitate CDE interoperability beyond basic document exchange (Documents API) [30] and issue management (BCF API) [27].

The reliance on proprietary formats and centralised platforms thus impedes a more mature data integration and reuse practice. The main requirement for a SSoI is thus not that all data is maintained by the same CDE provider, but rather that this data is expressed in a machine-readable way, with benefits rising as the data becomes more structured, open and interoperable, conform with Tim Berners-Lee's 5 star deployment scheme for structured data on the Web [13]. In this way, data can be easily (and semantically) related to other data that is not necessarily hosted by the same central platform, without the need to transfer data from one CDE to another one. Using open formats, dedicated services to check project consistency can focus on the data itself instead of being dependent on the interactions that are allowed by CDE platforms. For example, a network of CDE's based on an open domain standard such as IFC would facilitate easy exchange of data in a common format, thereby eliminating faulty conversion processes. However, due to its domain dependency, the semantic potential of the IFC schema remains limited. Therefore, methods are required to interrelate 'siloed', domain-specific data with data from other domains (see multi-models, Section 2.1). Inevitably, these relationships must be expressed independently from the level of heterogeneous project data that describes the actual product, i.e. on a metadata level.

2.3 Web-based BIM and FAIR data

The concept of Web-based BIM is often related to the 'final' level of BIM maturity. The BIM maturity levels were first defined in the famous wedge diagram by Bew and Richards [15], and describe a progression from 2D CAD drawings (Level 0) towards 3D models (Level 1), the concept of BIM and CDEs (Level 2), and the general notion of integrated, interoperable data and Big Open BIM (Level 3 and beyond) (Figure 2.3). They influenced the development of multiple international standards for digital collaboration in construction and asset management, such as the well-known ISO 19650 series [84, 85].

To its full extents, a Web-based Big Open BIM environment would not only allow seamless data integration between the 'classic' AEC disciplines but also with adjacent domains (see Section 2.1). In order to give this multidisciplinarity a place in the narrative, the industry-specific nomenclature 'Big Open BIM' and 'BIM Stage 3' is to be left behind in favour of a more domain-agnostic terminology.

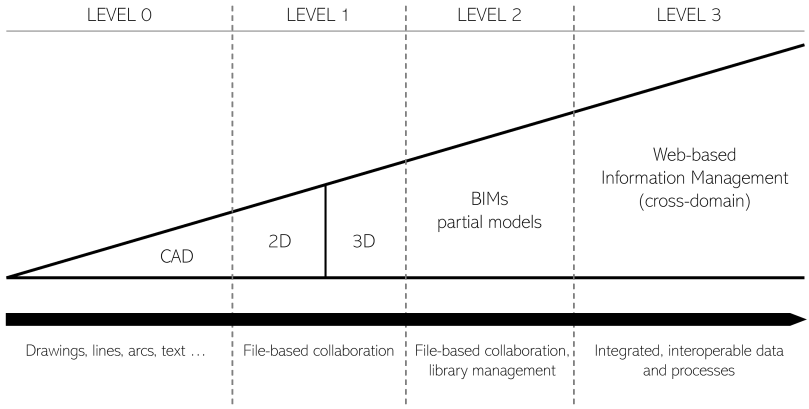

Figure 2.3: The BIM levels of maturity. Based on [15].

2.3.1 The FAIR-principles

From such broader perspective, the available data for a particular task in the project will be a combination of building-specific and contextual datasets, all of them located on the Web. In this light, the Big Open BIM philosophy shows many common traits with the more domain-agnostic FAIR-principles (Findable, Accessible, Interoperable and Reusable) [200], a set of recommendations for data stewardship on the Web which is increasingly being adopted around the globe, well beyond its initial scope of scientific data. This is, amongst others, illustrated by the fact that in 2018, the European Commission devised an action plan for FAIR data in the EU [39]. The FAIR principles are often associated with Open Data [122], but they can be equally applied to access-restricted data: the overlaps have less to do with data being 'open for everyone' than with the need for the data being structured for maximal discovery and reuse potential.

One of the main ideas behind FAIR and the 5-star deployment scheme of open data is that data can be structured to allow unforeseen scenarios, carried out by an unknown number of users [40]. Added value will rise from combining different datasets (or collections of datasets) in order to answer a particular question. In many scenarios, building information will not be the primary input, but it will only provide contextual information for other activities during the construction and operational phases. To cater for such activities, other information streams needs to be integrated as well, such as user preferences, real-time sensor measurements and room access control.

Although the FAIR principles themselves are essentially technology-agnostic, there are only few technologies today that may *fulfil* the requirements for FAIR-compliant data. The Semantic Web technology stack [14] is amongst those technologies [116] and has, for similar reasons, also been identified numerous times as a potential game-changer to achieve a more interoperable, data-oriented AECO practice [8, 129]. An extensive stack of Semantic Web-based data models ('ontologies') exists by now, capable of describing and relating knowledge in an interdisciplinary and Web-wide manner.

2.3.2 Towards a Federated CDE

The semantic richness of a digital building catalogue determines whether it can be used for multiple, diverging purposes. In an optimal scenario, information duplication is avoided as much as possible, while existing information is maximally reused. Since buildings and infrastructure are amongst the essential fabrics of society, the number of scenarios that can make use of this data (be it as primary input or only as an auxiliary) is virtually unlimited, extending the existing scenarios that work on each domain individually. By linking asset data to the broader context of the built environment, and of society in general, more and richer usage scenarios emerge. This does not only relate to 'pure' AEC disciplines, but also for adjacent domains for which buildings and the built environment act as the main theater. Considering such widely-scoped multi-models, it seems very impractical, maybe even impossible, to try to centralise all this information and maintain a single point of access, managed by one project-external company, i.e. the CDE provider. After all, such centralisation would often mean duplication of information which already exists elsewhere on the Web. When this data is not read-only, updates to one of these duplicates will lead to ambiguities, weakening the concept of a SSoI.

On the contrary, a *federated* Web environment consisting of multiple nodes will provide a much more scalable solution for multi-disciplinary data integration – much like the Web itself. This indicates the inherently decentral and interdisciplinary nature of knowledge representation in the built environment, and the impossibility to ever determine a 'complete' data model for digitally describing an asset. However, what comes very close to such complete data model is an extendable yet case-specific combination of data models, as embodied by the concept of multi-models.

The above discussion illustrates that in such open data environment, the boundaries between a (containerised) knowledge base of an individual building and the broader knowledge base that is the Web [169] start blurring. It becomes

difficult to speak of 'a project' as a fixed collection of datasets – which is the case in current-day CDEs. Rather, it is important to be able to *discover* the right subsets of the Web related to a given problem. Specific combinations can be made to allow to formulate an answer to particular (chain of) (sub)problems. This results in a freedom to create a multi-disciplinary, heterogeneous and federated catalogue, recursively aggregating other catalogues and resources in a Web-wide Knowledge Graph (KG) that is asynchronously enriched by numerous people in context of specific interaction tasks. Contrasting with the existence of a fixed 'project container', the data boundaries are defined by the the set of resources required to answer a specific query. This set may be limited to (a submodel of) an individual building or extended to a street, a city, a group of buildings which have the same typology (schools, railway stations, churches) or a collection of projects to which a particular office contributed.

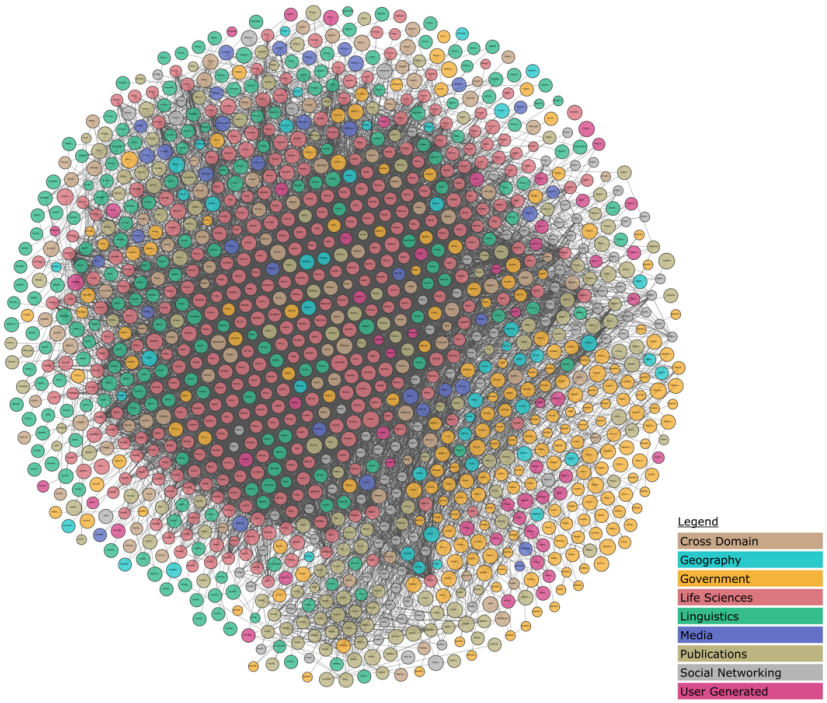

Figure 2.4: The Cloud of Linked Open Data (LOD Cloud) in 2023 (source: https://lod-cloud.net. Accessed 2023-06-16)

A centralised CDE with well-defined internal project boundaries (e.g., the coordination model or individual partial models), can be integrated in such federated CDE as just one of potentially many nodes in this network of interoperable data, alongside nodes for governmental regulation services, geospatial institutions and more (Figure 2.4). Centralised and federated CDEs are thus not mutually exclusive. When this is recognised, it is only a small step to consider project-specific data as an equally federated catalogue: stakeholders can manage their own IP on an access-controlled server of their choice (a '*vault*') and just semantically reference the related data of other stakeholders, on their respective data vaults. In this way, data is *shared* rather than *exchanged*.

To allow for the above-mentioned flexible project boundaries, a shared, Web-wide authentication and authorisation system is necessary between the vaults. Apart from improved data extensibility, this makes it possible to address the double patchwork (Figure 2.1) in an elegant way, mirroring real-world organisational structures. This does not necessarily conflict with the SSoI paradigm, at least not more than centralised solutions. However, unambiguous (and machine-readable) agreements are needed between the stakeholders to be able to automatically validate a known set of published project datasets and identify potential ambiguities or clashes. Given a stakeholder's authority, expertise and roles in the project, their contributions should then be included or rejected for particular usage scenarios throughout the building life cycle.

Several research projects have focused on the usage of Semantic Web technologies for data-based information exchange for AECO purposes. The DRUMBEAT platform [74], which finished in 2017, is an implementation of the Linked Data concept for building information with a RESTful API to access the objects. The platform is open and decentralised, meaning that any party can install it and use it to publish its data. Objects published at different hosts and installations of the platform can refer to each other, and the platform ensures that the remote inverse links get updated appropriately. The ifcOWL ontology [128] forms the core data model of the DRUMBEAT project. Since the completion of the project, several new technologies and specifications have been developed, many of which will be discussed in this dissertation (e.g., ICDD, Solid [111], SHACL [101]).

Another initiative oriented towards Web-based BIM is the SCOPE (Semantic Construction Project Engineering) project [75], which was finalised in 2021. In SCOPE, a system architecture is proposed that allows micro-services to interact with full RDF-based BIM models stored in a triple store. These micro-services

(e.g., a Revit-to-RDF exporter, a rendering service) are then bundled in a Docker network and exposed via a dedicated API gateway, so users can access them in a unified way, e.g., to be used for structural analysis in other tools. SCOPE focuses on the connection of a project CDE with external product datasets, using semantic Web technologies. Therefore, it takes a federated approach to AECO data management.

2.4 User interaction with Multi-models

A project consortium is a changing network of domain specialists, who are seldom IT experts. Specific interactions with digital asset data must thus happen via dedicated, relatively low-threshold GUIs. Compared to lower-level interactions with project information (e.g., by writing queries), auxiliary resources such as geometry, imagery, point clouds and textual files are preferred as a 'proxy' to interact with semantic information about a given object such as qualitative and quantitative properties or classification. In any industry, the presence of GUIs is essential to allow domain specialists to interact with digital data structures.

Desktop-based applications such as BIM and CAD authoring tools have shaped the way people think about computer-aided information creation and design. As discussed earlier, many vendors of BIM authoring tools are shifting focus towards entire Web ecosystems - CDEs. The status of the authoring tool thereby changes from being the one-and-only way to interact with a BIM *file* to being one of many possible windows to data in the cloud. In proprietary CDEs, this data will be primarily exposed through a vendor-specific API, which can be called by external services. Examples of such infrastructures are the BIMserver [10] Javascript API [123], Autodesk Platform Services (APS) [5] and the Trimble Connect API [171]. As such APIs and libraries will base upon a published Javascript library, highly specialised, standalone 'Single page' Web applications (SPA) [115] and headless services that interact with the CDE can be easily created using plain Javascript or any frontend framework (e.g., React [113], Angular [61] or Vue [202]).

A distinction can be made between *creation* of resources in different formats and *linking* these resources in a multi-model. While custom applications are optimised to intuitively create BIM models, spreadsheets, construction details etc., the heterogeneity of multi-models makes it much more difficult to find the right tool for the activity of linking – building the multi-model. After all, theoretically every possible combination of media types and ontologies may

be possible. A multi-model of limited usage scope contains a well-defined set of data models, which implies that the development of a GUI can happen in a rather straightforward way, and be supervised by a single organisation. Such infrastructures are offered by the RUB-ICDD and Wistor projects, which were discussed in Section 2.1. In a non-AECO context, Microsoft's Power BI [114] also offers a commercial visualisation framework to link heterogeneous resources and gain new business intelligence insights. However, earlier in this dissertation it was described that one of the primary reasons of existence for the multi-model concept is that there needs to be a freedom of which data types to include. A federated multi-model extends this promise in that any 'relevant' data can become part of a bigger, federated catalogue, whatever its datatype, wherever it resides. One could see the scope of a federated multi-model as potentially *unlimited*, following the Big Open BIM paradigm but contrasting with those multi-models that are created based on a well-established set of goals and boundaries.

In many cases, data that is already part of a federated multi-model will then serve as a proxy to integrate new data. The example of damage documentation in context of a heritage renovation was discussed in Chapter 1. In that case, there might not be a 3D model yet, although interesting statements can already be made and linked with other available (contextual) information. Damage data can become part of a catalogue by linking it with images, with a draft sketch of a floor plan, a graph representation of the building or just by textual records. In other words, a framework that allows interaction with generic multi-models in a systematic way, will not know their constituent file formats beforehand. Hence, it must be able to dynamically adapt the user interface based on the available datasets (e.g., imagery, geometry, semantics, text) and the task at hand. Ideally, it should thus be possible to initiate and *enrich* a maximally structured multi-model from whatever domain is required by the current task (topology, damage, product information, user data ...), based on whichever auxiliary resource (3D geometry, CAD plans, cityGML, point clouds, imagery, textual documents, sensor data ...) is already available or needs to be created or linked. Consequentially, an indefinite amount of interaction interfaces may exist. Throughout this dissertation, the term 'enrichment' is used to describe the process of connecting different sources to each other and to the Web-based multi-model, with the aim of broadening the usage scope of the (federated) multi-model. For example, classifying an instance as a 'wall', adding physical properties to it, linking it to 3D geometry or pictures, would all be considered 'enrichment'.

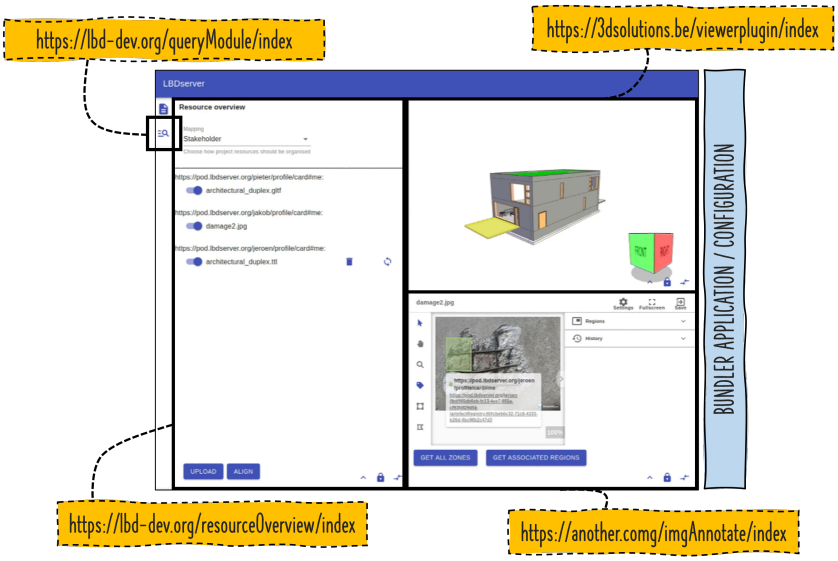

Figure 2.5: Different modules can be combined using micro-frontend federation, allowing a custom GUI for a specific linking activity in a specific project.

When each of those enrichment activities can be seen as an independent, modular part of a bigger configuration, these modules may be reused across multiple scenarios, dynamically combined in a tailor-made GUI that is only based on the available project data (e.g., images) and the current task (e.g., damage documentation) (Figure 2.5). When it is allowed for those independent modules to be published in a federated manner [186, 135], a freedom of innovation is introduced: third parties can create and publish a module for new media types or ontologies allowing flexible linking of heterogeneous multi-model data at a fraction of the development cost of 'monolithic' (Web) applications and without the approval of a single authority. Modules may then be published as libraries, to use while developing a dedicated application, or as a module that can be wired together with other modules in a configuration, as a 'low code / no code' [142] environment for non IT-specialists (i.e., the large majority of domain experts).

Note that these federation arguments are independent from the arguments put forward in Sections 2.3 and 2.3.2, which were related to project data discovery and the notion of a SSoI. However, the arguments for applying the FAIR principles can be used here as well. Consequently, semantic metadata

descriptions of these modules will allow discovery of federated modules and their flexible combination into various GUI-configurations.

With this in mind, this dissertation will also consider an interface-oriented framework that is as versatile as the federated multi-models it ought to interact with. Just as the datasets in a federated multi-model must be discoverable and queryable (Section 2.3.2), a federated UI framework must allow easy client-side discovery (and consequently, decentral querying), aggregation and configuration of dynamic interfaces.

2.5 Conclusion

This chapter covered the background and rationale against which the research presented in the following chapters will be positioned. The decentral nature of the industry was placed against the current centralised approach for CDEs. Opening up the digital built environment to other disciplines will allow better integration, and thus reuse of datasets for various purposes. However, this interdisciplinarity requires a collaboration system other than current centralised approaches for CDEs - instead, the entire Web is envisaged as an infrastructure to discover, aggregate, link and publish heterogeneous information about the built environment. In this light, the Web itself becomes a CDE. An alternative mindset towards a modular configuration of graphical user interfaces, based on semantic descriptions, will offer the handgrips to build such heterogeneous catalogues by linking new information to the existing stack of datasets. In the following chapter, this context will be used as a background to determine high-level requirements for project data in a federated CDE.

2.6 Related Publications

This chapter contains edited fragments or concepts derived from the following publications:

- Jeroen Werbrouck, Pieter Pauwels, Jakob Beetz, and Léon 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. URL: https://biblio.ugent.be/publication/8633673 (accessed 2024-3-18).
- Jeroen Werbrouck, Pieter Pauwels, Jakob Beetz, and Erik Mannens. "ConSolid: a Federated Ecosystem for Heterogeneous Multi-Stakeholder

Projects". In: *Semantic Web Journal* (2023). Accepted. URL: https://biblio.ugent.be/publication/8633673/file/8633674.pdf (accessed 2024-3-18).

- Jeroen Werbrouck, Pieter Pauwels, Mathias Bonduel, Jakob Beetz, and Willem Bekers. "Scan-to-graph: Semantic enrichment of existing building geometry". In: *Automation in Construction* 119 (2020), p. 103286. URL: https://doi.org/10.1016/j.autcon.2020.103286.
- Jeroen Werbrouck, Oliver Schulz, Jyrki Oraskari, Erik Mannens, Pieter Pauwels, and Jakob Beetz. "A generic framework for federated CDEs applied to Issue Management". In: *Advanced Engineering Informatics* 58 (2023), p. 102136. URL: https://doi.org/10.1016/j.aei.2023.102136 (accessed 2024-3-18).
- Jeroen Werbrouck, Madhumitha Senthilvel, and Mads Holten Rasmussen. "Federated data storage for the AEC industry". In: *Buildings and Semantics*. CRC Press, 2022, pp. 139–164.
- Jeroen Werbrouck, Ruben Taelman, Ruben Verborgh, Pieter Pauwels, Jakob Beetz, and Erik 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. URL: https://ceur-ws.org/Vol-2636/09paper.pdf (accessed 2024-3-18).

Chapter 3

Storage and Discovery of Federated Projects

Considering a project as a case-specific subset of resources on the Web allows a very high degree of expressivity, as well as a high scalability and exploratory potential. However, compared with centralised solutions based on known file formats, this introduces additional complexities, which often require a combination of technologies in order to be addressed. In this chapter, the complexities related to the data storage part of the ConSolid ecosystem are considered. The first topics of discussion are the generic data storage characteristics for such ecosystem to function as intended. These will be formulated in a technology-agnostic way (Section 3.1) to illustrate their general applicability, independent from the solutions that will be proposed later in this dissertation. Secondly, technologies for achieving these characteristics will be identified, as well as design choices to make the ecosystem more robust (Section 3.2). This includes a discussion on how the Solid specifications may provide the foundations for such ecosystem of data, how data will be stored on a Solid data vault and the role of metadata. Finally, this section describes the technological consequences of these design choices, regarding discovery and querying of data on a Pod. These consequences to an extent that they cannot longer be addressed by the default Solid specifications. An approach for extending Solid Pods by providing an access-controlled union graph of the stored RDF data will be proposed and implemented as a Pod-external service, or a 'satellite'.

With the basic technologies set, Section 3.3 describes the metadata patterns to create a federated multi-model. This includes the basic setup of a project, a discussion on multi-project catalogues, how ecosystem-external resources can become part of the multi-model, and how query-based (virtual) views on the project can be generated to present the data in a specific form. In Section 3.4, it is discussed how these metadata patterns allow to configure a variety of stakeholder network configurations.

The concepts developed in this chapter will be demonstrated in Section 3.5, using the iGent tower case study (see Section 1.7). The chapter concludes with an outline of the software deliverables related to storage in ConSolid 3.6.

The exact structure of ontologies and vocabularies will always be somewhat arbitrary, since many alternative approaches can be devised. However, at some point, specific data models need to be chosen in order to practically implement the ecosystem. Whenever possible, reuse of existing, recommended web vocabularies is desirable.

Background technologies for this chapter are introduced in the following appendices:

- Appendix B (Semantic Web technologies (RDF, SPARQL, SHACL));
- Appendix C (Solid and federated authentication);
- Appendix D (The FAIR principles).
- Appendix E (Web-Based and Industry Containers);
- Appendix H (The ConSolid vocabulary).

3.1 Characteristics of the ecosystem

This section describes (technology-agnostic) characteristics for an ecosystem capable of supporting *storage* and *discovery* of *heterogeneous*, *federated* data. In this work, the following high-level characteristics, which will be argumented in the next paragraphs, are identified:

- **C 1:** ***Decentral, Secure Storage****: data is stored decentrally, in access-controlled data vaults hosted by different storage providers.*
- **C 2:** ***Decentral Authentication****: a visitor's identity is verifiable based on protocols for decentral identity providers. This identity can be used for both authentication and authorisation.*
- **C 3:** ***Guaranteed Data Heterogeneity****: multi-models consist of structured (e.g., semantics), semi-structured (e.g., geometry) and unstructured datasets (e.g., images). The ecosystem safeguards this heterogeneity and make no assumptions on the formats and schemas it will host.*
- **C 4:** ***Uniform Metadata Descriptions****: although datasets may differ in schema or data type (R3), they have uniform metadata description patterns as to allow general discoverability and filtering of datasets. These metadata descriptions should allow FAIR data stewardship.*
- **C 5:** ***Uniform Query Language****: metadata is queryable with a general-purpose query language.*

In Section 2.2, access-controlled data storage was mentioned as the core functionality of a CDE upon which all other (higher-level) functionality bases. Characteristic C1 indicates that, if the environment is to be scalable across the Web, a decentral storage infrastructure is necessary. As the Web itself is a decentral storage system, this characteristic can easily be addressed when considered standalone, in a context of openly published (read-only) knowledge. However, additional complexities arise when considering confidential project-specific data: project servers in a federated CDE cannot not be organised as if they were 'open' Web APIs, which implies that authentication and authorisation will play a role. Numerous technologies exist to authenticate and authorise clients who request data access, but in a Web-wide ecosystem, their implementation is less evident than in a closed environment maintained by a single CDE provider. Of course, this is not unique to the AECO industries. Decentral authentication technologies (Characteristic C2) avoid dependency on central identity providers (IDPs) and allow any actor to maintain a verifiable identity on the Web, either by themselves or by outsourcing to a trusted provider. Effectively, a standardised identity management protocol is essential to verify identities between disparate identity providers and allow more streamlined information exchange between CDEs.

Characteristic C3 (Heterogeneity) dictates that the ecosystem cannot impose which data types to include in the project graph, so their degree of structuredness can vary heavily. This can be evaluated by checking whether there are any dependencies between the ecosystem and the media types of the resources it hosts. Avoiding such dependencies allows a more domain-independent and time-resilient usage of the ecosystem. On the other hand, the potential for data reuse increases with a higher degree of structure: there are benefits for each incremental star on the 5-star deployment schema for open data proposed by Tim Berners-Lee [13]. The more structured the data becomes, the less difficult it becomes to extract the knowledge it describes; consequently it is also easier to reconfigure or adapt this information to data structures suitable for a particular scenario or conforming to an existing data standard (Chapter 6). Although use of highly structured data is thus *recommended*, in order to facilitate cross-domain data exchange and be resilient in supporting current and future data formats, the ecosystem cannot *enforce* (or assume) any degree of structure for project datasets.

Some resources will remain semantically ambiguous because of their inherent unstructuredness or because a less structured form is preferred over the highest possible degree of machine-readability (e.g., because they are more

standardised or have better tooling). Examples of the former are binary data formats for storing imagery or point clouds; the latter could refer to geometric serialisations. Note that it is possible to express geometry using Semantic Web technologies, but doing so offers only limited benefits compared to the increased complexity and processing cost [132, 182, 136].

In other words, datasets from any BIM maturity level may be part of the multi-model (a multi-model containing exactly one Revit model could still be considered a multi-model). Hence, the threshold for adopting the framework is significantly lowered: the creators of building datasets can be fully unaware of the over-arching federated CDE infrastructure.

The characteristic on heterogeneity (C3) also inherently means that resources may be encoded using proprietary formats as well, although this will jeopardise reusability and hence diminish compliance with the FAIR principles.

To allow uniform discovery of otherwise heterogeneous project datasets, it is necessary to maintain metadata records (C4), i.e., data about the datasets. These metadata records need to be structured in a uniform way acrossthe ecosystem. Section 2.3 briefly discussed the FAIR principles as a set of recommendations for data management, regarding findability, accessibility, interoperability and reusability. The FAIR principles are heavily oriented towards machine-readable metadata description. Hence, they offer a sound framework to organise the metadata layer of the ecosystem, in tandem with the 5-star paradigm. A uniformly structured metadata layer thus leaves room for safeguarding the heterogeneity characteristic: via the metadata descriptions, actual project datasets can be discovered and (pre)filtered for usage in a particular scenario, independent from the internal schema's used for the project data, using a single query language that is compatible with the data structure used for expressing metadata (Characteristic C5).

3.2 Technologies and Design Choices

Now that the characteristics are clear, they can be mapped to existing technologies that are capable of addressing them. It should be clear that the identified technology stack is not the only possible one to fulfil the characteristics. In many cases, alternative options will exist. Although an explicit comparison between multiple technological options will not be made in this dissertation, the choice for specific technologies will be motivated based on their intrinsic properties and capability to address the characteristics defined in Section 3.1.

3.2.1 Secure Storage of Heterogeneous Datasets

As for characteristics C1 (Decentral, secure storage) and C2 (Decentral authentication), the Solid ecosystem [111] and the associated WebID-OIDC [37] protocol are identified as suitable candidates. Solid consists of a set of standards and specifications for decentral data management on the Web. At the core of Solid lies a URL that can be used for authentication and authorisation: a WebID (an unambiguous identity on the Web), accompanied by a personal data vault (a 'Pod'). A Solid Pod can be seen as a resource server with a decentralised authentication layer on top. The resources on a Pod can have any file extension, which addresses C3 (Heterogeneity) to a sufficient degree in context of this dissertation. Incorporating database-hosted knowledge as well is considered out-of-scope for this dissertation, although a theoretical roadmap will be proposed in Section 3.3.3.

The authentication layer allows a Solid server to communicate with the Identity Provider (IDP) of the visiting agent and check if they are allowed to interact with resources on the Pod. At the core of the Solid ecosystem lies the Solid Protocol [33], based on the Linked Data Platform (LDP) specification (Appendix E). Access control to a Pod is regulated with WebID-OIDC authentication, with WebID-based verification allowing to maintain a single identity on the Web. This does not eliminate the business case of trusted third-party identity providers (e.g., Google, Facebook, Autodesk), but in any case a communication layer between them needs to be in place.

Existing CDE solutions implementing the WebID-OIDC protocol would make a giant step towards enabling cross-CDE data access, functioning as one of multiple nodes in a bigger, multi-project and multi-stakeholder ecosystem of (both commercial and open) CDEs. For example, to allow cross-CDE information exchange, current standards such as DIN SPEC 91391-2 still require corresponding user accounts to be set up at both CDEs [49] – possibly resulting in an overload of accounts with corresponding security issues. An independent authentication protocol such as WebID-OIDC would avoid this.

3.2.2 Design of the Vault

Since Solid is based on the LDP specification, every resource on a Solid Pod 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 [44], its current application

hard-wires 'implicit' 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' [33]) and inheritance of 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 parent folders. This while these parent directories may change over time, thereby invalidating the resource URL. In this sense, this discussion is related to the recommendations for 'Cool URIs' and URI design strategies [12]. Consider the following example, related to the stages of publication (Work-in-progress, Shared, Published, Archived) as defined in ISO 19650 [84, 85]. 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 usage scenario (see Section 3.3.4). When the container structure is embedded in the resource's URL, this is not possible. 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 + identifier'. Note that this approach – URLs composed of unique identifiers instead of implicit, human-readable semantics – is already common in central cloud storage providers (e.g. Google Drive or Microsoft OneDrive), but is not part of the Solid ecosystem.

The 'meaningless' identifier of a resource can now, for example, be a GUID or a cryptographic hash (e.g., using trusty URIs [103]) of the resource's content. In the latter case, the hash can be used to verify whether the content has been changed since its creation. While this is a form of protection against data tampering, it also makes it impossible to update document-based information without invalidating the hash. Because a building project includes resources that can be subject to changes (cf. the example on publication status explained earlier), such as geometric models and metadata (see further), this dissertation will opt for the GUID approach which bears no relation to the resource's content. This 'form' still follows the Solid Protocol and LDP, as it allows to

interact with resources via HTTP – there are just no slash semantics beyond the root of the Pod, which essentially is a (very large) `ldp:Container`.

Because this approach eliminates implicit URL-based semantics, the 'meaning' that allows semantic containerisation and filtering on higher-level (domain-specific) layers must now come from an explicit metadata record that 'enriches' the heterogeneous resource (instead of from the parent container). When the metadata records are themselves dereferenceable with a URL of the form 'root + identifier', a metadata record will be handled in exactly the same way as any other resource on a vault: it is just a regular resource which happens to advertise itself as being the metadata record of another resource. This is illustrated in Figure 3.1. In the next section, ConSolid metadata records will be discussed in more detail.

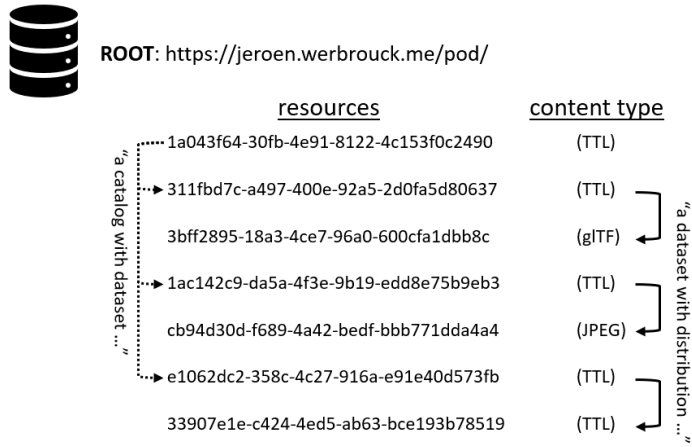

Figure 3.1: A flat list of resources in a data vault. Metadata records are regular resources that identify themselves as containing metadata about another resource.

3.2.3 Metadata-based Resource Dicovery

In order to allow discovery and semantic filtering of relevant datasets in the federated multi-model, a common syntax is needed for metadata (Characteristic C4). To allow maximum expressivity and domain-independence, it must be possible to add custom statements to metadata records – the ecosystem should not impose what can or cannot be said about the project datasets it contains.

To achieve the highest potential for the resource to be findable, accessable, interoperable and reusable, this metadata syntax needs to comply with the FAIR principles. In Section 2.3, the Semantic Web technology stack was mentioned as one of the few that may fully address said principles [116]. The use of the Resource Description Framework (RDF), the cornerstone of the Semantic Web, therefore seems an apt choice. RDF is a domain-agnostic and expressive syntax for graph-based data structures, with inherent Web federation functionality. The choice for which technology to use for expressing metadata is inherently related to Characteristic C5, which describes it is necessary to be able to uniformly query the metadata. Because the SPARQL Protocol And RDF Query Language (SPARQL) query language is the W3C standard for querying RDF-based data, it is the evident choice to address C5.

RDF can be used to describe both metadata as domain-specific data, using *vocabularies* (or *ontologies*). However, as per C3 (Heterogeneity), the ConSolid ecosystem leaves the choice open to whether or not use RDF ontologies or other for describing project data. Agreeing on the use of RDF on a *metadata* level, however, already offers a lot of benefits without changing the nature of the resources themselves. Non-RDF documents cannot be queried directly with SPARQL – but a metadata record allows registration of queryable semantic descriptions and therefore discovery and filtering of the dataset within the larger federated multi-model. The form of a metadata record as a specific type of document is still very much under discussion in the Solid ecosystem. However, this does not introduce additional complexity in the ConSolid project: Section 3.2.2 mentioned that, being regular RDF resources, metadata records will be handled in exactly the same way as any other resource on a vault.

The vocabulary chosen to structure this metadata network, is the DCAT vocabulary (see Appendix E). This is because DCAT is the preferred metadata vocabulary for data points compliant to the FAIR principles [119]. Furthermore, DCAT allows to:

1. address 'containerisation' 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;
3. avoid conflicts with Solid's default usage of containers via LDP.

Listing 3.1 gives an example of an RDF-based metadata record in the ecosystem, aggregated in a larger catalogue, using DCAT.

```
# The catalogue resource, with URL pod:0d1ffe69
bb:0d1ffe69 a dcat:Catalog ;
    dcat:dataset bb:1e19ed7c .

# The dataset resource, with URL pod:1e19ed7c
bb:1e19ed7c a dcat:Dataset , consolid:ProjectResource ;
    ex:publicationStatus ex:workInProgress ;
    dct:created "2022-07-29T14:13:28.167000"^^xsd:dateTime ;
    dcat:distribution bb:90329a28 .

bb:90329a28 a dcat:Distribution ;
    # Where to find the actual project file.
    dcat:accessURL bb:90329a28 .
```

Listing 3.1: Catalog and metadata record of a resource, using the DCAT vocabulary.

Note that metadata records can now be updated without changing the resource or its URL, contrasting with the 'implicit' slash semantics used in the default folder-based approach (Section 3.2.2). Revisiting the example on publication statuses, changing a metadata tag from 'work-in-progress' to 'shared' allows the original URL of the resource to remain intact. Datasets can now be filtered based on fine-grained query parameters. However, for this to be possible, there must be a queryable union graph of all metadata records on a vault, which respects the access rights of individual resources. In this dissertation, this is achieved via a custom extension to the Solid technology stack, which will be discussed in the following section.

3.2.4 Extension: A SPARQL interface to the Data Vault

Two options will be considered to allow query-based resource discovery on a Pod. The first one is based on link traversal [72, 167] technologies, while the second one includes the usage of an access-controlled SPARQL interface. Link Traversal Query Processing (LTQP) is possible because of the URL-based nature of Linked Data: upon discovery, URLs can be followed to find additional knowledge. As Taelman et al. [167] note, it is a theoretically interesting concept, but its practical use has been limited so far due to performance considerations: as query execution and source discovery happen simultaneously, pre-execution optimisation is not possible. Feasibility and optimisation of LTQP strategies in a Solid environment are actively researched [167], but are not included in this dissertation. Therefore, the focus of this section lies on the creation of an access-controlled SPARQL endpoint to the Pod.

A (private) SPARQL endpoint can be considered an apt interface to query the union of knowledge graphs on a data vault. Such interface is already implemented by the Enterprise Solid Server (ESS, Inrupt) [78]. However, it is permissioned such that only the Pod owner can access it – external agents are denied access to this functionality. In this section, a more fine-grained setup will be described.

Access control on a Solid Pod currently works on the level of (RDF and non-RDF) documents, by means of Access Control List (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 3.2.2). To maintain compatibility with the current 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 authorisations and the authorisations granted to the public (mostly `acl:Read`). A query to retrieve this union from the SPARQL endpoint is given in Listing 3.2. This query needs to be executed by an agent with full access to the endpoint, i.e. the Pod owner or a delegate service. This indicates the need for a middleware layer between the union graph and the requesting agent, in order to filter the results before responding to the request.

```
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
  {?acl acl:accessTo ?resource } UNION {?acl acl:default ?resource }

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

Listing 3.2: SPARQL query to retrieve the resources a given agent is able to query. This query assumes that every resource on the Pod has its own ACL. Richer descriptions of the ?resource variable are possible, to yield a smaller but more detailed result set.

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 3.3. More information on the differences between 'FROM' and 'FROM NAMED' in SPARQL queries can be found in [99].

Considering that a typical SPARQL endpoint does not implement such access control and query adaptation functionality, a proof-of-concept was developed in context of this thesis, where all requests to the SPARQL endpoint pass through a proxy service which has been granted full read permission, acting on behalf of the Pod owner. This accounts for the middleware layer discussed earlier in this section. This service extracts 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 dissertation, the proxy service and database service will be considered as one, acting as a 'satellite' to the Pod. The SPARQL satellite should be easily discoverable. Here, this is done by registering the triple pattern in Listing 3.4

in the WebID of the owner of the Pod. An alternative approach is to having the Pod provider expose a fixed, reserved endpoint (e.g., {pod-url}/sparql), eliminating the discovery step.

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

# The modified SPARQL query (option 1: FROM)
# Reconstruct a "permitted union graph" of the Pod, but comes at the expense of query execution time
SELECT ?element ?dam ?cause
FROM <https://b-b.be/data/0e12ae3b>
...
FROM <https://b-b.be/data/2cb2e8a2>
WHERE {
    ?md a dcat:Dataset ;
        dcat:distribution ?dist .
    ?dist dcat:accessURL ?durl ;
       dcat:mediaType ?mt .
}

# The modified SPARQL query (option 2: FROM NAMED)
# Triple patterns must be enclosed in a named graph to query, i.e., all wrapped triples must occur in this graph.
SELECT ?element ?dam ?cause
FROM NAMED <https://b-b.be/data/0e12ae3b>
...
FROM NAMED <https://b-b.be/data/2cb2e8a2>
WHERE {
    GRAPH ?g {
        ?md a dcat:Dataset ;
            dcat:distribution ?dist .
        ?dist dcat:accessURL ?durl ;
            dcat:mediaType ?mt .
}}
```

Listing 3.3: The SPARQL satellite dynamically updates any query to only include those named graphs to which the visitor has read access.

```
<https://b-b.be/data/profile/card#me>
        consolid:hasSparqlSatellite <https://satellite.b-b.be/sparql> .
```

Listing 3.4: Triple pattern to find the SPARQL satellite to a Pod, via the WebID of the Pod's owner.

The above-described approach can be considered an intermediary 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 by Dedecker et al. [44], 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.

The prototypical implementation of the SPARQL satellite is based on a stack of existing, open source components. The Community Solid Server (v3.0.0) codebase [173] was modified 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[2]. As a SPARQL store, Apache Jena Fuseki[3] 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[4]. WebIDs are retrieved from authenticated requests (WebID-OIDC and OAuth 2.0 [69]), after which the query in Listing 3.2 is executed and its results injected in the original query, before sending it to the SPARQL store.

Performance optimisation of this setup is out of the scope of this thesis. 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. For each of these resources, the owner was given full access rights, resulting in 644 permitted resources (as every resource has a corresponding ACL resource). 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 3.3 yields the results listed in Table 3.1, using a machine for which the specifications are listed in Table 3.2. For each modification, Table 3.1 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 recources (Listing 3.2)). A bypass can be created for the Pod owner, which would allow to query the full union graph

[2]Solid Community Server + SPARQL store: https://github.com/LBD-Hackers/SolidCommunity_Fuseki/tree/dissertation. Accessed 2023-10-23.

[3]Apache Jena Fuseki: https://dlcdn.apache.org/jena/binaries/apache-jena-fuseki-4.10.0.zip. Accessed 2023-01-27

[4]SPARQL satellite prototype: https://github.com/ConSolidProject/auth-satellite/tree/dissertation. Accessed 2023-02-23.

without restrictions (cf. the Enterprise Solid Server by Inrupt). In this case, the original query is executed on the default graph without modification (column 'Default' in Table 3.1).

	#permitted resources	ACL query (Listing 3.2) (ms)	FROM (ms)		FROM NAMED (ms)		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	

Table 3.1: Query execution time for the query in listing 4.
*** Query execution time directly on the SPARQL endpoint .**
**** Total query execution time for: "client > satellite > SPARQL store (ACL) > satellite > SPARQL store (query) > satellite > client".**

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

Table 3.2: Specifications of the machine hosting the satellite implementation and the Fuseki SPARQL store.

In this minimal example, it can be observed 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 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 reside in a single named metadata graph ('disjoint graphs'). 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 thus still have its applications. Note that reduction of the set of queried resources can already be integrated in the access control query (Listing 3.2), 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') may only be acceptable for some applications. However, it is still a possibility when the list of targeted resources is limited.

3.3 Data Patterns

Now that the basic infrastructure of resources (i.e., metadata and the SPARQL satellite) has been set up, we can investigate how multiple datasets can be grouped into larger catalogues: collections of pointers to relevant datasets on the Web. As decided in Section 3.2.3, the DCAT vocabulary will provide the means to do this. This section will explore how this allows to integrate multiple local, vault-specific catalogues into larger aggregations that can be discovered via single URLs – access points of the federated multi-model.

3.3.1 Dataset Collections

A main requirement for a stakeholder to be able to integrate their knowledge into a larger federated project, is to be able to bundle their own contributions into a *local* project catalogue. As shown in Listing 3.5, 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.

```
# the catalogue resource, at URL bb:d07af06a
bb:d07af06a a dcat:Catalog, consolid:Project ;
    rdfs:label "myFirstProject" ;
    dct:identifier "d07af06a" ;
    dcat:dataset bb:1e19ed7c , bb:32ad4402 ;  # local aggregation

    # 2nd level aggregation, including access points on other vaults
    dcat:dataset dgfb:aa3c09de , arcadis:bd503663 .
```

Listing 3.5: A catalogue which is only one aggregation level away from the aggregated dcat:Dataset instances on a Pod.

All stakeholders can thus create their own local catalogue for a collaborative project, independently from the others, forming an *access point* on the vault to find project data. An interesting feature of DCAT catalogues (`dcat:Catalog`) is that they can aggregate other (lower-level) catalogues as well. This allows to easily reference the catalogues of other stakeholders (on their vaults), making it straightforward for a client to discover and query also the contributions of these other stakeholders. Moreover, this aggregation is done with the same `dcat:dataset` relationship that is used to aggregate `dcat:Dataset`s into a catalogue. No matter the depth level of a catalogue, it should eventually lead to discovery of all resources in the catalogue. 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 way. A query that allows to discover the datasets of a federated project, without knowing the aggregation depth from the access point to the eventual datasets, is given in Listing 3.6. This query will work for both local catalogues, single-project federated catalogues and multi-project federated catalogues (see Section 3.3.2).

```
SELECT ?dataset WHERE {
    <{access-point-URL}> dcat:dataset+ ?dataset .
}
```

Listing 3.6: Query to discover the metadata records of all datasets in a catalogue, using SPARQL property paths of arbitrary length.

The term 'partial project' will be used 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. This is illustrated in Figure 3.2.

This approach of nesting partial projects also allows (but doesn't oblige) a project team to agree on having a single registry about who is part of the team, e.g. on the Pod of the appointing party [84] or the project manager. While every stakeholder should still maintain a local access point that groups their contributions, to allow discovery of other project data, they can simply point to this one catalogue to find the local access points of other stakeholders (Figure 3.3). Especially in the design and construction phases, where almost all information will be produced by a well-known group of stakeholders, such shared catalogue will streamline information discovery and avoid ambiguity about the vaults that contain project information.

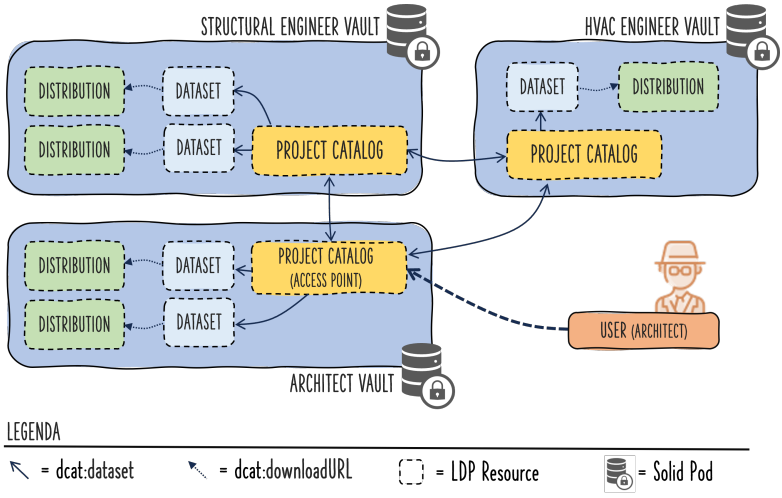

Figure 3.2: Starting from one catalogue on a specific data vault, it is possible to find all datasets in the federated project.

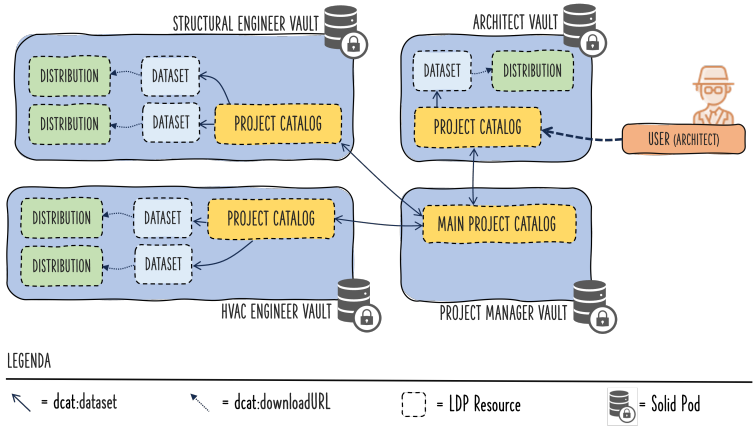

Figure 3.3: Via their own catalogue of the Project, a stakeholder can discover a main catalogue listing other stakeholders. This main catalogue can be used to propagate and find the other project datasets.

Because bi-directional links will typically be present between access points of different stakeholders (during well-coordinated phases of the building life cycle), a mechanism that prevents infinite loops should be part of the querying algorithm. Such mechanism is implemented in query engines powered by the Comunica framework [166].

3.3.2 Multi-project Catalogues

Another feature enabled by the chained aggregation of catalogues are higher-level aggregations, which allow for a discovery-based approach of nested DCAT catalogues. This means that the boundaries of a 'project' can be very flexible, depending on the catalogue that is used as an access point: it may range from single-building level over neighbourhoods or a geographically scattered group of buildings from the same typology ('all libraries in Flanders', 'all bridges in Germany'). Figure 3.4 illustrates that a federated project can be part of (multiple) higher-level aggregations: starting from one higher-level catalogue, the same query that is listed in Listing 3.6 allows to propagate through the network and discover all project datasets.

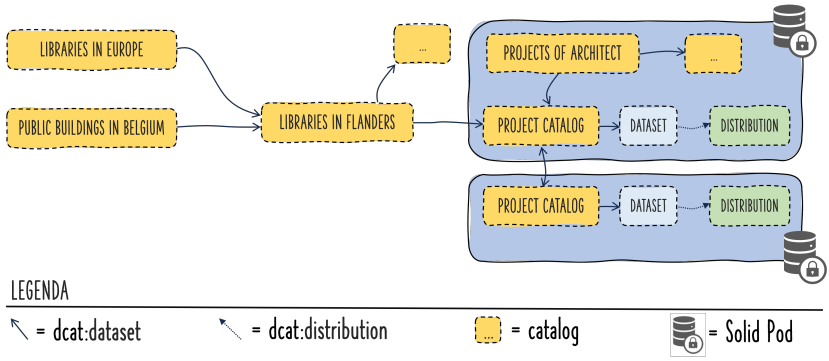

Figure 3.4: Specific project data may be aggregated by multiple (external) catalogues.

One option to do this, is to use a query engine that supports link traversal, such as the Comunica [166] Link Traversal Engine [59] in combination with the LDP interface to the vaults. This can be sped up by only using link traversal for discovering the partial projects and their associated SPARQL endpoints, after which regular queries can be sent to these SPARQL endpoints. Higher-level aggregations will make use of this pattern to scale up the level of aggregation without increasing the complexity of the queries.

3.3.3 Integrating external resources and databases

Until this point, datasets and their distributions were considered to be both stored on a vault. However, this is not an absolute requirement: if datasets (e.g. geospatial or governmental) are open and persistent, they can be easily integrated in the federated multi-model by creating a DCAT metadata record

that is aggregated by the project access point, pointing to this external resource or database as a distribution (Listing 3.7). Client applications can now discover this dataset and present its distributions to the end user as part of the project.

```
@prefix archive: <https://beeld.ugent.be/media/photos/.63696/>

# The dataset resource on the Pod, with URL bb:1e19ed7c
dgfb:1e19ed7c a dcat:Dataset , consolid:ProjectResource ;
    dcat:distribution dgfb:90329a28 .

dgfb:90329a28 a dcat:Distribution ;
    # Where to find the actual resource (picture of iGent tower).
    dcat:accessURL archive:w940q85_Z2016_012_029.jpg .
```

Listing 3.7: A metadata record on a vault can indicate a vault-external resource as its distribution, integrating it into the overall catalogue.

In a similar way, metadata records can point to *access-controlled* data that is not strictly document-based (e.g., sensor data or SQL tables), extending the heterogeneity characteristic C3. A description on complete flow on how to do this is out of scope for this dissertation. However, a few notes can be made on the nature of such setup. Although the Solid Community Server [173] allows to instantiate configurations with different storage options (file-based, in-memory etc.), some questions remain unanswered at the time of writing. Firstly, in order to be accessible via an LDP interface, a resource must be completely identifiable with its URL, i.e. there can be no further specifications in the request body. Naturally, a database API often expects a body, e.g. including a specific query. Secondly, there is no specification on how access-control would happen for such specialised databases.

As an example, let us consider the setup of a time-series database, based on the InfluxDB 2.7 Open Source (OSS) server [77]. An InfluxDB database can contain multiple 'buckets', which have specific access rights, retention policies etc. Just like with the SPARQL satellite, ACL resources will not map directly to the API of a specific database, but rather to specific endpoints on a satellite service that negotiates between the Solid server and the external database (Figure 3.5). In this example, these endpoints offer functionality to query a specific bucket. An InfluxDB bucket has an identifier and a name, but is not specifically queryable with a dedicated URL – instead, the bucket is normally included in the query. Using this identifier, however, the satellite service can easily expose a URL that *does* include this identifier. Additionally, the URL can include query parameters, such as a specific query for the bucket. The

correct query can then be reconstructed internally by the satellite service. This addresses the first question, namely to have a specific response for a specific URL, that can be retrieved by a simple HTTP GET request.

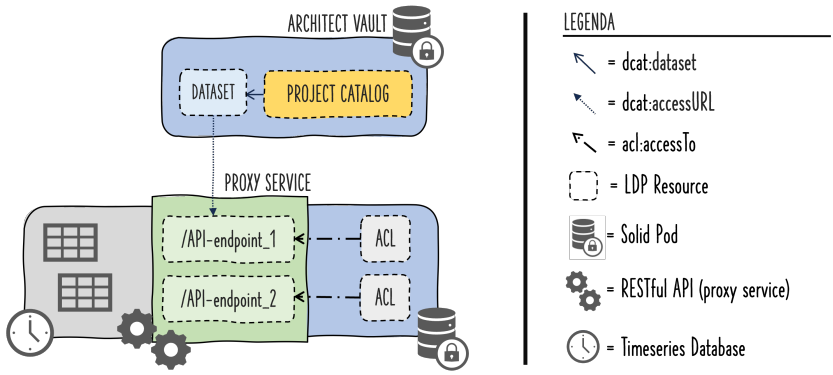

Figure 3.5: Infrastructure for protecting project information in a database that does not support WebID-OIDC.

The second question, namely how to map specific access control rules to such URLs, depends on the link of the database and the proxy server with the Solid Server. In this example (Figure 3.5), the satellite manages its own integrated Solid Server, where ACL documents point to specific endpoints using the default property `acl:accessTo`. The proxy service is then able to reconstruct database-specific queries from URLs, and determine whether the client has the correct access rights. This concludes the excursion towards integration of external databases in the ConSolid ecosystem; further research is needed to verify this approach, and check its compatibility with the Solid specifications.

3.3.4 Virtual Containers

When a ConSolid project has been discovered, a union of all partial projects and their aggregated resources forms the (hybrid) knowledge graph of the project. This has been compared to a federated multi-model earlier; it was discussed how multi-models base upon information containers for information exchange. Furthermore, it was described how all project information on a data vault can be aggregated in a catalogue, and how these catalogues recursively aggregate similar catalogues on other vaults. As these catalogues only reference their contained datasets – they don't effectively *contain* them – they may be seen as a global *virtual* container for project information. This is in line

with the SSoI paradigm, which implies that data snippets have one single location; everything else will be a reference to this original location. Note that the approach described above, where a resource may be aggregated by multiple catalogues (i.e., as a graph), differs from the common definition of information containers in the AECO industries, where resources can often only be aggregated in a single parent container (i.e., as a tree). However, for practical AECO scenarios, a tree-based view on the project can be more easily generated based on a fundamental graph description than vice versa: the tree is the set of resources that can be discovered downstream, starting from one access point (see Listing 3.6).

Apart from defining the overall project access points, DCAT catalogues can also be used for establishing more fine-grained information containers, or specific 'views' (filters) on project data. For example, based on the publication status of the resource, on data type, topic or ownership. We can differentiate between persistent containers and dynamically generated containers. The former relates to semantic catalogue descriptions that are stored on the data vault as a named graph; the latter are short-living as query response bodies in the form of RDF documents, to be directly processed by the client application.

Revisiting the example of a resource's publication status, a virtual catalogue can be created, referencing all resources in the project that have a publication status "Shared". A SPARQL CONSTRUCT query can immediately generate such virtual container, which can now be the input for higher-level services. For example, a GUI that presents the project as a tree-based folder structure. An example CONSTRUCT query for a DCAT-based virtual container is given in Listing 3.8.

```
CONSTRUCT {
    ?virtualContainer a dcat:Catalog ;
        dcat:dataset ?accessURL .
} WHERE {
    ?ds ex:publicationStatus "Shared" ;
        dcat:distribution/dcat:accessURL ?accessURL.

  # dynamically generated container
  BIND(UUID() as ?virtualContainer)
}
```

Listing 3.8: SPARQL query to filter project datasets that are 'shared' and return their distributions a (temporary) virtual container.

3.4 Project Configurations

There are several options to apply the above-described infrastructure of vault-based, federated catalogues to organisational structures in the built environment, more specifically to the project phases of design and construction, involving the AECO disciplines and a well-known project team. In this section, the following configurations will be discussed:

1. ConSolid as a centralised CDE;
2. ConSolid as a federated CDE, involving office employees;
3. ConSolid as a federated CDE, involving subcontractors.

The first configuration, i.e., a single data vault for design and construction data, most closely resembles current centralised CDE infrastructures. Specific project partners still have their own WebID, to allow provenance of project data on the Pod. A Solid Pod does not impose any restrictions on data structures, but for a project of this type, this might be necessary, for example to define the minimal requirements for metadata in the project, such as its creator and modification history. Therefore, direct access to data on the vault (via the LDP URLs of resources or the SPARQL satellite) is to be limited; instead an external proxy service can facilitate more specific interactions with project data (see Chapter 5 and 6). Although this can be seen as a centralised storage approach, it can be extended to other vaults from the moment that this is required: as discussed in Chapter 2.3.2, a centralised CDE can easily be adopted as part of a bigger, federated CDE. Since this option still bases upon decentral authentication with WebIDs, it allows an office to avoid account management for every CDE that would normally host their project information. In other words, only data storage happens on a central vault, identity management is still done by each office individually. However, this solution does not address the double patchwork (Section 2.1), since information will not reside with the individual stakeholders – instead the environment will be hosted by one of the project partners, or a third party that is responsible for hosting.

A (slightly) more advanced setup is to allow every office to maintain their own data vault. Throughout this chapter, this setup has been the default one. It addresses the double patchwork, as every stakeholder can decide where information is hosted: either locally or by a trusted party that offers specialised hosting services. For illustrative purposes, the office's WebID was repeatedly used to interact with data in the federated project. However, in a practical case, requests will be made not only by the authenticated office account, but

mostly by individual employees, resolving to their own WebIDs, or WebIDs related to their roles in the project.

When a company in the project does not want to keep track of all the employees of other companies, they can refer to groups (`vcard:Group`) of WebIDs instead of individuals, using the property `acl:agentGroup` instead of `acl:agent`. An example group definition is given in Listing 3.9. As these groups can be defined in dereferenceable resources, every office can maintain groups and subgroups of their own employees, allowing company-external project partners to point to these groups instead.

```
bb:members a vcard:Group ;
  vcard:hasMember <https://b-b.be/jan/profile/card#me>,
    <https://b-b.be/paulus/profile/card#me>,
    <https://b-b.be/data/profile/card#me> .
```

Listing 3.9: Groups meant for access control can be defined outside the data vault hosting the ACL resources, but must be publicly available to allow other Solid Servers to fetch the resource.

Some risks occur, however, when agent groups are defined externally from Pods that reference them: the owner of the Pod where the group is defined will now control who has access to the protected resource on the other Pod. Moreover, the Solid server hosting the ACL resource and the document it protects should be able to access this group each time access rights need to be checked. This requires quite some trust between the Pod owners. Therefore, some alternatives can be identified:

1. a copy of the agent group is kept at the pod with the resource, and needs approval for synchronisation whenever the original changes. This can be done using a functional satellite (Chapter 6).
2. a trusted project partner (e.g. the project manager) maintains the lists of trusted WebIDs of all stakeholders.
3. a certificate-based procedure for decentral pattern-based access control is used. A primary workflow for this is outlined in Chapter 5.

Yet, further organisational breakdowns are still possible, and might offer more benefits for data management within an organisation. For example, a company vault can be maintained as an aggregator (see Chapter 6) and unified access point for external collaborators, but project information is stored on the vaults of responsible employees, or alternatively on vaults corresponding with tasks or functions in the project.

Although the data patterns allow to federate project contributions to an arbitrary depth, a critical note should be made on discovery performance: following property paths of arbitrary length in a federated environment will likely make use of LTQP – which will quickly meet performance limitations [167]. However, a differentiation can be made between tools that may maintain application state (such as browser applications) and stateless services (such as RESTful servers). A browser application may propagate the network at startup and keep a list of SPARQL satellites that should be queried. This will avoid execution of costly link traversal queries for each request. However, RESTful servers do not 'remember' information from previous requests by design, so either the project discovery step must be re-executed with each request, or all requests must contain a list of endpoints to query. An intermediary solution is to allow the data from sub-company vaults (employees or roles) to be aggregated by a higher-level service (see Chapter 6). This would expose a unified endpoint for the entire office (or even the entire project), while internally maintaining the more fine-grained data management benefits.

A final multi-vault configuration exists when offices appoint subcontractors for specific tasks in the project. According to ISO 19650, a project counts a an *appointing party*, a few *lead appointed parties* (e.g. an architect, structural engineer, HVAC engineer, owner) and then *appointed parties*, i.e. parties appointed by one of the lead appointed parties. Maintaining the federation logic, appointed parties will not work on the lead appointed parties' main data vaults, but will have their own vaults to store their contributions to the project, based on datasets provided by the lead appointing party. When combined with the potential single access point mentioned in Section 3.3.1, an aggregation structure can be set up of a project in its design and construction phases that looks like the breakdown structure of parties in ISO 19650 (Figure 3.6): any of the stakeholders can now just aggregate (1) the shared access point provided by the appointing party (see Figure 3.3) and (2) their own appointed parties' access points. This establishes a chain between the different levels of stakeholders, avoiding the need for every party to keep track of the access points of all other parties.

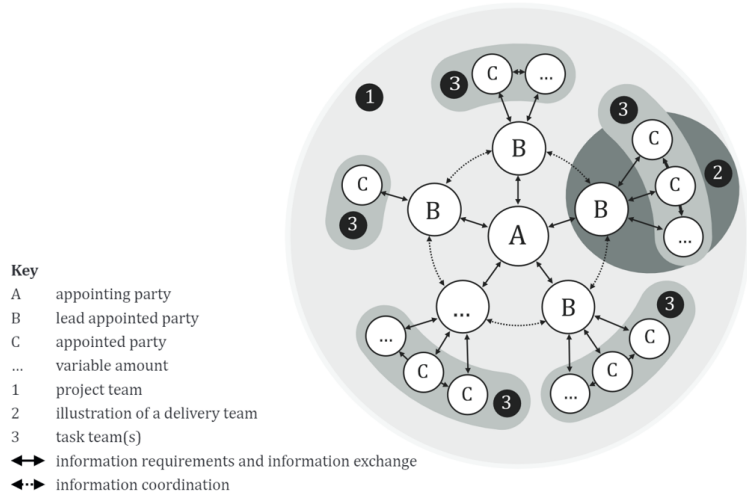

Figure 3.6: The ISO 19650 standard specifies an organisational hierarchy between parties and teams. Source: [85].

3.5 Case study: iGent Tower

In this section, the technologies and data patterns outlined in Sections 3.2 and 3.3 will be illustrated with the case of the iGent tower. As described in Section 1.7, three stakeholders will be part of the project: Bureau Bouwtechniek, Arcadis, and the DGFB (i.e., the Facility Management office of Ghent University), since it concerns a University Building. Every stakeholder maintains their own Solid Pod and associated WebID and a SPARQL satellite. Bureau Bouwtechniek will host the architectural model, Arcadis the three others. At this starting point of the operational phase, the DGFB does not host any project resources yet. The setup is visualised in Figure 3.7.

The steps described in this section can be reproduced by following the instructions published on the Github[5]. Since the datasets used in the iGent case study are non-public, the open source datasets of the benchmark duplex project have been included in the reproduceable demo. The configuration file for the duplex project is available as an attachment to the case study reproduction guidelines, and can be used as a general template to start using the ecosystem using any set of existing partial models.

[5]Case Study reproduction guidelines, https://github.com/ConSolidProject/cde-satellite/tree/dissertation/demo. Accessed 2023-10-30.

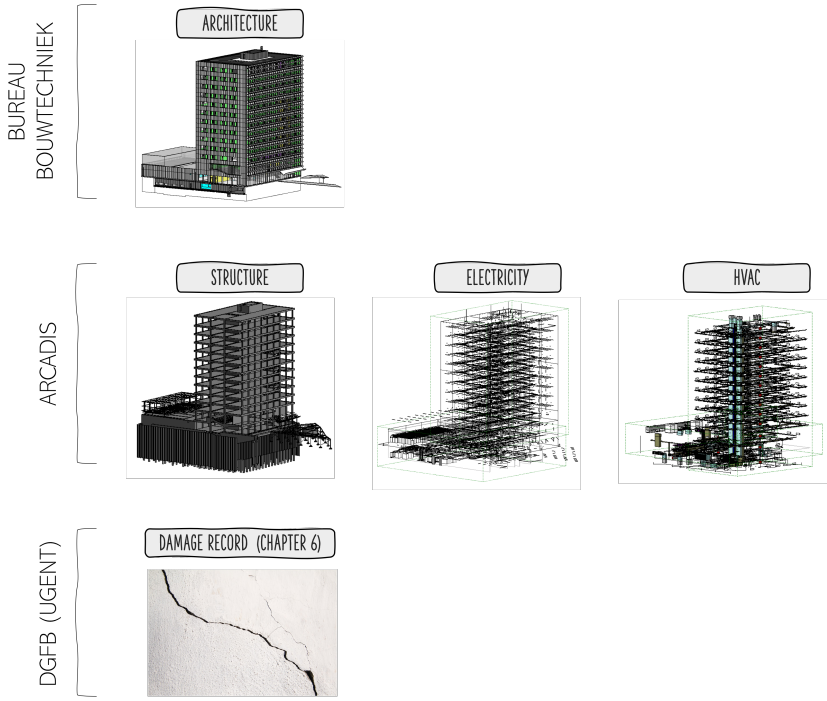

Figure 3.7: Case study setup.

The project setup indicated in Figure 3.3 is chosen. In this setup, the central catalogue is provided by the owner of the asset, i.e. Ghent University (here acting through the DGFB). The initial step taken by the DGFB is to create a new catalogue on their vault, and invite the other stakeholders to participate via a Linked Data Notification (LDN) [34]. An example invitation is given in Listing 3.10.

```
<> a as:Announce ;
  as:actor <https://b-b.be/data/profile/card#me> ;
  foaf:primaryTopic <http://w3id.org/conSolid/ProjectCreation> ;
  dct:description "You are invited to join the project {projectURL}." ;
  as:object <{projectURL}> ;
  dct:created "2023-08-28T12:30:39.593Z"^^xsd:dateTime .
```

Listing 3.10: Linked Data Notification (LDN) [34] inviting a specific actor to participate in a project.

The recipients of this invitation now need to (1) create a local catalogue on their own vault, aggregating the remote catalogue mentioned in the invitation and (2) confirm their involvement by sending a response notification including *their* partial project, which can now be recursively aggregated by the main project catalogue of the DGFB. This sequence is illustrated in Figure 3.8.

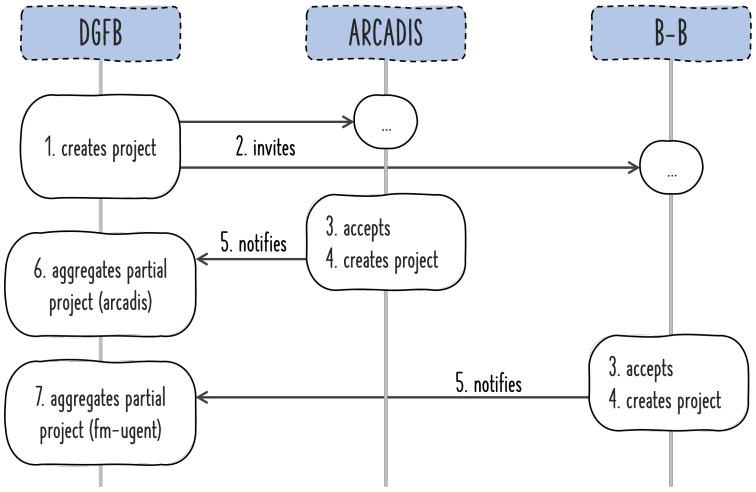

Figure 3.8: Sequence Diagram for project setup.

As the framework encourages the use of open data formats, the available original Autodesk Revit partial BIM models for Architecture (modelled by Bureau Bouwtechniek (BE)), Engineering, Electricity, and HVAC (Heating, Ventilation and Air-Conditioning) (all three modelled by Arcadis (BE)) are converted to a semantic model for RDF (Turtle) and a geometric model (glTF) [62], resulting in 8 separate resources. The conversion of a BIM model (IFC) to Linked Building Data (LBD) ontologies can be done using the existing IFCtoLBD converter [20], resulting in an LBD graph based on the Building Topology Ontology (BOT) [146] and the BuildingElement Ontology (BEO) [130]. The partial models with their respective amounts of triples are listed in Table 3.3.

Owner	Model	#triples
B-B	Architecture	192178
Arcadis	Electricity	101860
Arcadis	HVAC	273649
Arcadis	Structure	32892

Table 3.3: Amount of triples per partial BIM model after conversion to an LBD graph (PROPS depth level [20]: 1).

All stakeholders now store each converted semantic RDF model and geometric glTF model as a distribution with attached metadata (dataset) on their Pod. Metadata records and RDF-based semantic conversions of the project models are mirrored as named graphs on the SPARQL satellite. 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 in the project.

This concludes the setup of the stakeholder infrastructure for the iGent tower case study. This case study will be further developed in Chapter 4 and Chapter 5. Functionality to execute the above-described steps automatically will be included in the ConSolid satellite, which is described in Chapter 6.

3.6 Implementation

A prototypical API to interact with the basic setup described in Section 3.3 using the above mentioned patterns for storage and discovery of data, was created in context of this thesis. The API is available on Github[6] and NPM[7] as the Dataset Aggregation API ('Daapi').

3.7 Conclusion

In this Chapter, the characteristics for a federated, access-controlled ecosystem for heterogeneous data were identified, as well as technologies capable of addressing them. Design choices were applied to data storage on Solid Pod, to achieve the possibility of query-based views on a Pod. Using SPARQL as the main query language, fine-grained filtering of project datasets then becomes

[6]Daapi (Github): https://github.com/LBD-Hackers/daapi/tree/dissertation. Accessed 2023-10-31.

[7]Daapi (NPM): https://www.npmjs.com/package/consolid-daapi. Accessed 2023-10-31.

possible, using a combination of Link-Traversal-based querying and SPARQL satellites. A novel approach to query an access-controlled union graph of a Solid Pod was proposed, using 'FROM' and 'FROM NAMED' queries. Although query execution time increased in both situations compared to querying the default graph, the query time for 'FROM NAMED' graphs stayed around the same order of magnitude. Querying the Pod using 'FROM' queries resulted in significantly higher execution time. However, for the case of discovery and filtering, querying disjoint graphs (FROM NAMED) suffices. Opportunities were identified to further decrease query execution time.

Using recursive DCAT catalogues, pointers can be set to relevant project metadata, not only locally on the Pod of the client, but also on other Pods and even to datasets that are not aware of the ecosystem. Several options for multi-Pod collaborative environments were described, and although these options differ in organisational structure, the query patterns remain the same, illustrating the robustness of the framework and the freedom of a project consortium to organise project data as preferred. Of course, a higher complexity results in a higher discovery time to identify the actors in the network and their vaults. However, caching project metadata structures client-side or using a central middleware service (Chapter 6) will make abstraction of this and hence allow a better performance of the ecosystem.

Table 3.4 gives a summary of the characteristics of a federated CDE (as defined in this dissertation), focusing on this chapter's topic of storage and discovery. The characteristics are related to the technologies that were identified as suitable for supporting them. When a technology is a 'data technology' (e.g., RDF), more details are provided regarding the data patterns that were developed or adopted.

Characteristic	Technology	Data Patterns
C1 - Decentral, Secure Storage	HTTP(S), Web servers	n.a.
C2 - Decentral Authentication	Solid, WebID-OIDC	n.a.
C3 - Guaranteed Data Heterogeneity	Decoupling ecosystem and project data	dcat:Distribution / accessURL
C4 - Uniform Metadata Descriptions	RDF (DCAT)	dcat:Dataset / dcat:Distribution
C5 - Uniform Query Language	SPARQL	consolid:hasSPARQLsatellite

Table 3.4: Characteristics for a federated CDE (storage and discovery), corresponding technologies and data patterns as implemented in the ConSolid ecosystem.

3.8 Related Publications

This chapter contains edited fragments or concepts derived from the following publications:

- Jeroen Werbrouck, Pieter Pauwels, Jakob Beetz, and Léon 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. URL: https://biblio.ugent.be/publication/8633673 (accessed 2024-3-18).
- Jeroen Werbrouck, Pieter Pauwels, Jakob Beetz, and Erik Mannens. "ConSolid: a Federated Ecosystem for Heterogeneous Multi-Stakeholder Projects". In: *Semantic Web Journal* (2023). Accepted. URL: https://biblio.ugent.be/publication/8633673/file/8633674.pdf (accessed 2024-3-18).
- Jeroen Werbrouck, Oliver Schulz, Jyrki Oraskari, Erik Mannens, Pieter Pauwels, and Jakob Beetz. "A generic framework for federated CDEs applied to Issue Management". In: *Advanced Engineering Informatics* 58 (2023), p. 102136. URL: https://doi.org/10.1016/j.aei.2023.102136 (accessed 2024-3-18).

Chapter 4

Resource Linking and Annotation

With the technologies and data patterns described in Chapter 3, we can create a catalogue of separate datasets in a federated environment. Starting from a single access point, a client can discover the other vaults containing project material, and filter relevant resources using expressive SPARQL queries on their metadata. However, these datasets are not yet connected with one another, unless they are Linked Data resources that are fully compliant up to the 5th star of Linked Data and internally refer to each other. This is an intrinsic feature of RDF which is sometimes called *deep linking*. In this chapter, data patterns are discussed for connecting *heterogeneous* resources in a sub-document granularity.

In Section 4.1, two types of cross-document links are identified. The first type of links involves document annotation using the Web Annotation Data Model (WADM). The used data model will help in establishing the second type of links, related to *conceptual* cross-document identifier alignment. Furthermore, this section introduces the Reference Registry, a specific dataset in the ConSolid ecosystem for registering both types of links. The first type is then discussed in Section 4.2. Discussing the second type is the topic of Section 4.3. The data patterns will be illustrated with the iGent case study in Section 4.4. The chapter concludes with an outline of the software deliverables related to reference management in ConSolid (Section 4.5).

Background technologies for this chapter are introduced in the following appendices:

- Appendix B(Semantic Web technologies (RDF, SPARQL, SHACL));
- Appendix E (Industry Containers - ICDD);
- Appendix G (Sub-document identifiers for different document types);
- Appendix H (The ConSolid vocabulary).

4.1 Cross-document Links

In terms of the goals for the ConSolid ecosystem, it is not only necessary to discover project information by filtering metadata, but also by traversing related content in heterogeneous resources. In this thesis, two types of links are identified that need to be supported by the ecosystem on a data level:

- **C 6:** ***Cross-document Annotation****: this type of link describes relationships between datasets, documents and sub-document references, 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.*
- **C 7:** ***Cross-document Linking****: this type of link is related to the fact that many digital resources contain representations of the same real-world entities. Linking these indirectly together via an 'abstract concept' allows propagating from one representation to another, and use the best fit representation for the intended interaction.*

Naturally, solutions that address these characteristics must be compatible with the requirements stated in Chapter 3 as well. In particular, C3 (Heterogeneity) implies that media types cannot be assumed (or imposed). In other words: sub-document linking and annotation must take place on a metadata level rather than in the documents themselves, to 'even the field' between disparate resources and open up the silos of unstructured and semi-structured datasets, and make them be more permeable for external referencing. This approach aligns with current standards for multi-models, as illustrated by the Linkset specification in the ICDD standard [90]). Furthermore, maintaining a federated approach (C1) for cross-document linking is important to remain fully compatible with the rationale of Chapter 2. Following the same reasoning as Section 3.2.3, the RDF data model will be used for addressing both C6 and C7.

Both types of cross-resource links will be registered in specific resources on the vault, called Reference Registries. Using the storage patterns devised in Section 3.3.1, this means that a metadata record is created describing a `dcat:Dataset` that is also an instance of `consolid:ReferenceRegistry`, pointing to an RDF-based distribution that contains the links. This way, the SPARQL satellites can be used to quickly find the Reference Registries in a specific project, and allow a client to reconstruct the cross-document links throughout the project.

4.2 Annotation of digital documents

The Web Annotation Data Model (WADM)[149] and accompanying Web Annotation Vocabulary (WAV) [150] provide the patterns and vocabulary for annotation of resources on the Web. The data model describes a framework to annotate heterogeneous information on the Web, connecting a body (`oa:hasBody`) and a target (`oa:hasTarget`) via an annotation (`oa:Annotation`). The body describes the content of the annotation, whilst the target identifies its topic. Both body and target can have sub-document selectors and sources, although other options exist for fine-grained selection. Sub-document identifiers, which are called *segments* (of interest), can be registered in a metadata resource (such as the Reference Registry introduced in Section 4.1) using *selectors*. Practically, a metadata description of a segment (`oa:ResourceSelection`) refers to a source document (`oa:hasSource`), and a selector (`oa:hasSelector`). The registration of digital annotations conforming to the WADM will happen in a Reference Registry on the vault.

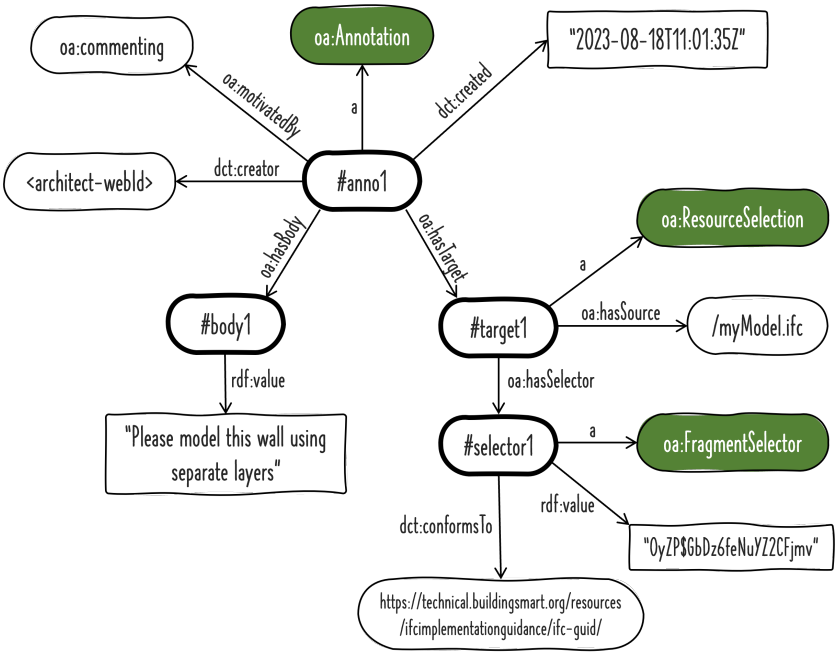

Figure 4.1: An annotation refers to a body and a target instance. The Web Annotation Data Model allows for fine-grained selection of sub-document identifiers.

4.3 Reference Collections

The second cross-document relation describes digital resources that reference the same real-world entity. Further in this dissertation, the set of all resources that refer to a single concept will be called a 'Reference Collection'. These federated Reference Collections can be seen as proxies for all known information about a specific entity, whose meaning comes entirely from the digital descriptions they aggregate - without the context of their representations they are just abstract concepts. This aligns with the notion of 'Linksets' in existing multi-model approaches. It is important to differentiate these representations from the real entities - a 3D element in a BIM model is not the real wall; neither is a picture: both are proxies helping us to digitally represent knowledge and make statements about a certain 'thing'.

This stands in contrast to common industry interpretations of BIM, which are often dominated by 3D models. Moreover, it also contrasts with some other academic interpretations in the field of Linked Building Data: while the multi-model approach handles all resources and identifiers alike, approaches such as the ones devised in the Ontology for Managing Geometry (OMG) [182] and the File Ontology for Geometry Formats (FOG) [21] considers the RDF building model as the main entry point to retrieve information about the asset, including referenced geometry and their metadata – making the existence of an RDF graph of the building mandatory prior to further enrichment. Due to the separation of cross-document links (which are considered a specific type of metadata) and document content, the data patterns put forward when using linksets are more generic. An RDF description will be one of many potential representations of a specific concept, albeit one that can be very flexibly enriched.

Using Reference Collections (Linksets), there is no hierarchy between resources: a pixel zone in an image can as much be a proxy for semantic enrichment as a 3D model; a semantic description can be equally complemented by a point cloud, a textual reference or a spreadsheet. By traversing over the interlinked selectors in a federated Reference Collection, one can discover an entity's relevant representations for a particular interaction scenario. This way, any perceived 'hierarchy' between project resources will be virtual and flexible - created by a thin UI layer that allows more intuitive interactions in a given scenario (see Chapter 7).

There are some subtle differences between common multi-model Linksets and Reference Collections as well. Current approaches are often oriented

towards archiving rather than providing the backbone of an actively used CDE. However, in an active CDE, linking sub-document identifiers throughout the life of a digital built asset will happen in an asynchronous and likely error-prone way. As one cannot know when alignment will take place, it is well possible that there will exist multiple 'aliases' of the same Linkset at the same time. For example, one stakeholder creates a Linkset containing a point cloud segment and an image representation of a specific door, while another stakeholder has already linked a 3D representation of this door with a semantic representation – in another Linkset. Over time, these Linksets should be aligned, as all their representations refer to the same concept.

The existence of one single metadata-level identifier (cf. Linksets) is thus fictitious anyway: at some point, these metadata-level identifiers will again need alignment, and so on, resulting in an indefinite depth level. In a federated CDE, this plays even more: as a project is not a fixed set of resources, the contributions of individual stakeholders must be able to function in a standalone way, and there will be no central repository to keep track of all identifiers of abstract concepts. Instead of grouping these aggregations under a fixed identifier, a symmetric data pattern is proposed to allow dynamic discovery of relevant representations of this concept on other vaults, without the need to define ever-more-abstract alignments between identifiers. Usage of these dynamically reconstructed 'aliases' allows to access project-wide collections starting from any Reference Registry on any data vault, similar to the symmetric aggregation in the DCAT catalogues introduced in Section 3.3.1. Note that the reasoning behind the creation of aliases does not apply in the case of annotations (Section 4.2): an annotation effectively connects only two parts, which contrasts with Reference Collections that may connect an indefinite number of references.

The following pattern, which will be based on the Resource Selection patterns of the WADM, can be used to aggregate sub-document identifiers. It has a four-part breakdown structure (Figure 4.2):

1. A Reference Collection (`consolid:ReferenceCollection`) at the highest level, used for identifying the concept but not for directly enriching it. It aggregates (`consolid:aggregates`) other Reference Collections or references (`consolid:Reference`). This allows discovery of aliases of a concept on other data vaults, using a query similar to the one in Listing 3.6, but using `consolid:aggregates` instead of `dcat:dataset` (Listing 4.1).

2. A Reference level (`consolid:Reference`), which includes the source (`oa:hasSource`) and the sub-document selector (`oa:hasSelector`). Other metadata (e.g. creation date) can be linked to the reference as well. A `consolid:Reference` will also be an instance of `oa:ResourceSelection`, as this is the `rdfs:domain` of `oa:hasSource` and `oa:hasSelector`.

3. A Selector level; a higher-level URI used to 'even the field' between RDF and non-RDF resources. This higher-level URI relates the value of the identifier to the Selector instance, as well as the specification to which the identifier conforms (`dct:conformsTo`). A list with suggested URIs for indicating conformance with specific file types is added to this thesis as Appendix G.

4. An Identifier level. Although more options exist in WADM, in this thesis, the identifier will take the form of a simple value, referring to a Literal using `rdf:value`. To maintain a unified approach, this holds for RDF resources as well - these are registered as Literals with datatype `xsd:anyURI`. Example values for different media types are given in Appendix G.

```
SELECT ?refOrAlias
WHERE {
    <https://b-b.be/data#b4d0ceb1> consolid:aggregates+ ?refOrAlias .
}
```

Listing 4.1: Query to discover the references and aliases of a given Reference Collection.

Selectors can be seen as higher-level identifiers, which even the field between RDF and non-RDF resources, because they indicate a value which is generally only valid within the boundaries of a certain dataset. In other words, identification of sub-document objects and their allocation is expressed in separate statements. In the case of RDF resources, deep links can still be present, although a unified aggregation strategy requires them to be connected via Reference Collections as well. Thus: any identifier, whether RDF or not, is initially only assumed valid within the document that mentions it. This separation is a core aspect of the asynchronous enrichment, allowed by the combination of the metadata-level Reference Collections 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

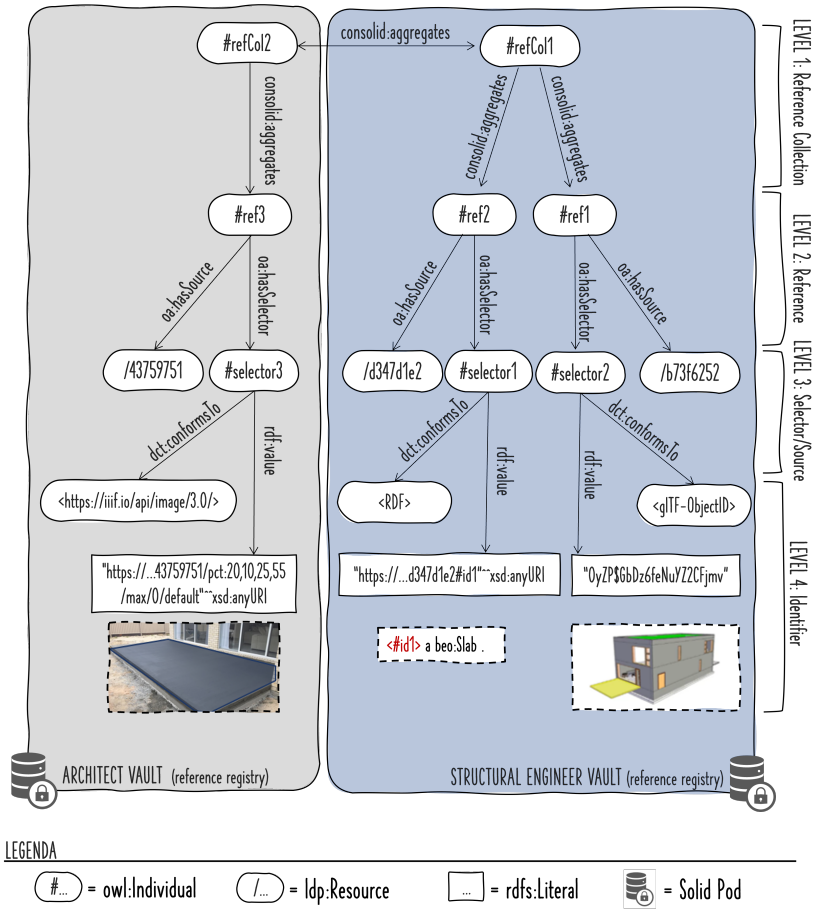

Figure 4.2: RDF patterns from federated Reference Collection to sub-document identifier.

phase where information is transferred from one vault to another) does not impact its sub-document identifiers, and it becomes possible to asynchronously register the new location of the resource in the network. This also holds when the identifier is URL-bound (e.g. IIIF, RDF). When the document is re-registered on another data vault, a new alias is created, referencing the new location and potential updated sub-document identifiers, resulting in a full equivalent of the original resource. When this alias has been created in the Reference Registry in the new data vault, the original alias may be removed from the first vault.

As illustrated in Figure 4.2, the recursive property `consolid:aggregates` allows a local Reference Collection to aggregate equivalent Reference Collections on other stakeholder vaults, i.e. its aliases. Since 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. For example, a bi-directional aggregation may exist between the local Reference Collection in the Facility Management (FM) company's vault and the one on the vault of the architecture company. In this example, the architecture company hosts a 3D model; the FM company can load this model in a viewer and use a 3D element of a wall instance as a proxy to link a picture of a damage on the real wall, as well as a semantic description of the damage, both located on his own vault. In a dashboard GUI, a representation of interest can be selected, after which the distributions, datasets and sub-document identifiers of its other (local and remote) representations can be found with a federated query, yielding an overview of all that is known about this concept, within the boundaries of the federated multi-model. The topic of user interaction with federated multi-models will be further discussed in Chapter 7.

4.4 Case Study: iGent Tower

In Section 3.5, a ConSolid project was initialised for the iGent tower, featuring 3 stakeholders. The partial models of the building were hosted by their creators, i.e. the vault of B-B hosts the architectural models (geometry and semantics), and Arcadis manages the Structural, HVAC and electrical models. Because for each model, the semantics and the geometry originate from the same IFC model, object identifiers can be maintained. Linking these as representations of the same abstract concept can thus be done automatically, following the method described in [109]. The resulting amount of Reference Collections is indicated in Table 4.1, per data vault.

Owner	#Reference Collections
B-B	19853
Arcadis	50779

Table 4.1: Amount of Reference Collections generated in the respective reference registries, connecting the glTF geometric models and the semantic resources (RDF).

In this case study, it will be described how a representative of the DGFB documents a damage occurrence in the building. The case study involves

the creation of a semantic representation of the damage using the DOT ontology [68], and a visual representation of the damage in a picture. The semantic description of the damage is linked to a representation of an actual element.

When the damage is, for example, registered via a GUI which displays the architectural model of B-B, this new representation can be immediately related to all the other representations of this element that are reachable via this geometric representation. The detailed flow of actions is illustrated in Figure 4.3; the consecutive steps are explained in the following sections.

4.4.1 Creation of Damage Graph by the DGFB

After selecting the damaged element via an existing 3D representation hosted by B-B (step 1-5), the Reference Collection associated with that element can be retrieved, along with its aliases on other vaults and any other possible representations (step 6-9). Following the documentation of the damages in the GUI (step 10), a metadata record is created on the vault of the DGFB (`damageMetaFM` in Listing 4.2), referencing an RDF-based distribution (`damageDistFM` in Listing 4.2), which is created simultaneously (step 11-12).

At the creation stage of the semantic damage record (step 12), two graph-specific identifiers are created, namely one for the damaged element (`damageDistFM:inst_a003d5d4`) and one for the damage area (`damageDistFM:inst_8aa5bc1b`) (Listing 4.2). They only exist in this specific document and have not yet been linked to any other existing project information.

```
# The RDF resource on the DGFB Pod that documents the damage
@prefix damageDistFM: <https://dgfb.ugent.be/data/cac6d088#> .
@prefix damageMetaFM: <https://dgfb.ugent.be/data/3c1c9a18#> .

damageDistFM:inst_a003d5d4
    dot:hasDamageArea damageDistFM:inst_8aa5bc1b ;
    a dot:ClassifiedDamage, cdo:SurfaceCrack ;
    cdo:crackWidth "35" .

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

Listing 4.2: Triples in the distribution of the Damage dataset, maintained by the DGFB. The metadata graph will be similar to the one listed in Listing 3.1.

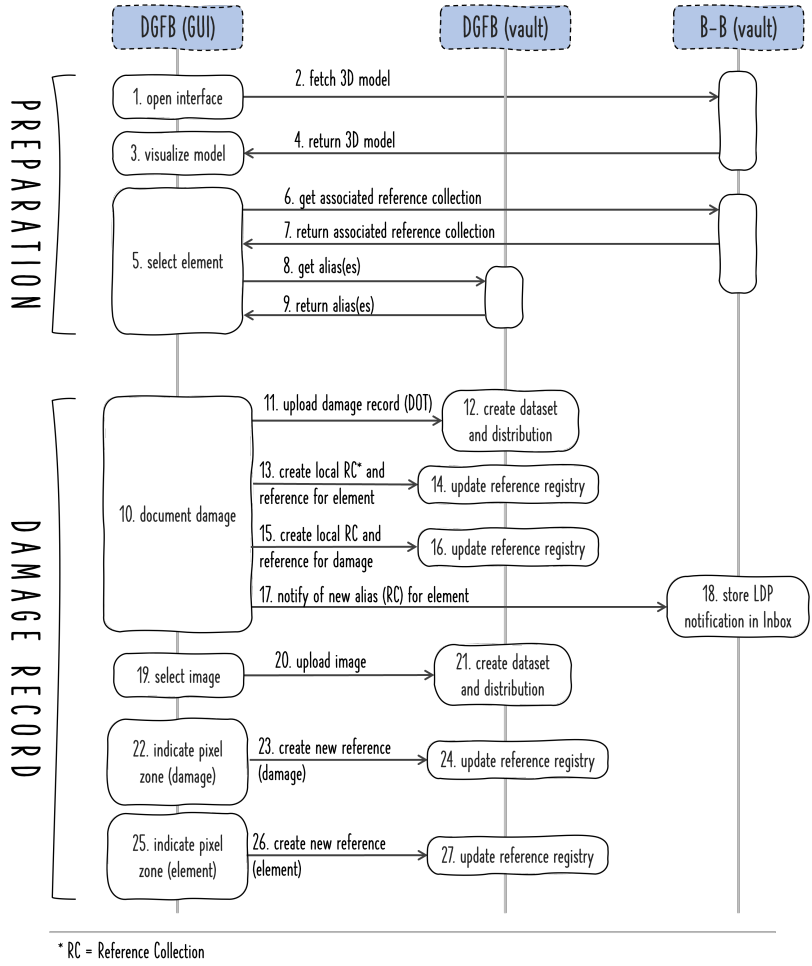

Figure 4.3: Sequence of actions for the damage documentation via a GUI.

4.4.2 Lifting local identifiers to project level

To be able to reference them in other resources in a later stage, both the damaged element and the damage area need to be aggregated by a local Reference Collection in the project's Reference Registry on the Pod of the DGFB (step 13-16). In this example, this is done with respectively `<#rc_bdb4526d>` and `<#rc_c2427dd9>` (Listing 4.3). These (relative) URIs represent the real-world entities and form the primary points for enrichment via a federated 'digital

twin'. Using the patterns described in Section 4.3, these abstract concepts are mapped to the local identifiers created for documenting the damage.

```
# The Reference Registries of the DGFB and B-B
# the local Reference Registry of the DGFB (i.e., the location of this
    snippet)
@prefix refFM: <https://dgfb.ugent.be/data/b677ce97#> .

# the (remote) Reference Registry of B-B
@prefix refArch: <https://b-b.be/data/f0c8cb37#> .

# The RDF resource on the DGFB Pod that documents the damage
@prefix damageDistFM: <https://dgfb.ugent.be/data/cac6d088#> .

# a Reference Collection is created for all references to the element
# local aggregation
refFM:rc_bdb4526d consolid:aggregates refFM:ref_a80854ae ,
# 2nd level aggregation of remote concept (related to the 3D element)
        refArch:rc_3de17fbe .

# the damaged element as identified in the damage semantics graph
refFM:ref_a80854ae oa:hasSelector refFM:sel_c69531c5 ;
    oa:hasSource damageDistFM: ;
refFM:sel_c69531c5 dct:conformsTo rdf: ;
    rdf:value "https://dgfb.ugent.be/data/cac6d088#inst_a003d5d4"^^xsd:
     anyURI .

# a concept is created for all references to the damage area
# the new reference is the only reference of this concept so far
refFM:rc_c2427dd9 consolid:aggregates refFM:ref\_f1f2704e .
refFM:ref\_f1f2704e oa:hasSelector refFM:sel\_27801276 ;
    oa:hasSource damageDistFM: ;
refFM:sel\_27801276 dct:conformsTo rdf: ;
    rdf:value "https://dgfb.ugent.be/data/cac6d088#inst_8aa5bc1b"^^xsd:
     anyURI .
```

Listing 4.3: Linking the local identifiers in the damage documentation graph to a 'Reference Collection', i.e. a neutral concept in the Reference Registry of the stakeholder.

Because the element was selected via a 3D representation, the concept located at B-B'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 in the DGFB's Reference Registry (Listing 4.3). A notification must now be sent to B-B, allowing them to automatically or manually aggregate this new concept as well. This makes bi-directional discovery possible (step 17-18).

4.4.3 Enrichment of sub-document identifiers: pixel region

The DGFB 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 (step 19-21). As only a 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 (step 22-24). This can be done using the relationships listed in Listing 4.4, which uses the International Image Interoperability Framework (IIIF) specification's image API [161]. A similar enrichment can be made for a pixel zone that represents the damaged *object* (step 25-27). These relationships are expressed in the DGFB's local Reference Registry.

```
# The Reference Registries of the DGFB and the B-B
@prefix refFM: <https://dgfb.ugent.be/data/b677ce97#> .
@prefix refArch: <https://b-b.be/data/f0c8cb37#> .

# The image resource on the DGFB's Pod showing the damage
@prefix pictureDistFM: <https://dgfb.ugent.be/data/9f9b1794> .

# this abstract concept links the image zone with the damage area.
# it is the same concept indicated in Listing 4.3
refFM:rc_c2427dd9 consolid:aggregates refFM:ref_ba0fe231 .

refFM:ref_ba0fe231 oa:hasSelector refFM:sel_1d2eee0e ;
    oa:hasSource pictureDistFM: ;

    # The identifier notation conforms to the IIIF specification
refFM:sel_1d2eee0e dct:conformsTo <https://iiif.io/api/image/3.0/> ;
    rdf:value "https://dgfb.ugent.be/data/9f9b1794/pct:20,10,25,55/max/0/
     default"^^xsd:string .
```

Listing 4.4: Defining an image's pixel zone as a specific representation of an abstract concept, identified by a Reference Collection.

This concludes the enrichment scenario. All stakeholders in the project can now discover and query this information starting from their own project access point. In Chapter 7, a method will be devised to dynamically generate user interfaces. An example interface for image annotation is given in Figure 4.4.

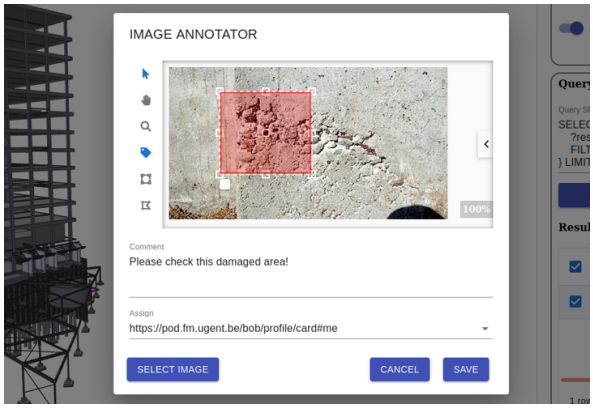

Figure 4.4: An image region annotator helps in interacting with the ecosystem in a more intuitive way.

4.4.4 Workflow: 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 project data; the Reference Registry delivers the corresponding Reference Collections and their identifiers, and the non-semantic resources (e.g., 3D models, imagery) can be used to visualise the results and access knowledge about an object documented in other resources via dedicated GUIs. In the case described above, a GUI that offers a perspective of damage management would allow an (authenticated) agent to perform the following steps:

1. Select the project (access point) of interest and discover the SPARQL satellites (Section 3.2.4) of its aggregated catalogs in order to query the permissioned union graphs containing these catalogs.
2. Query the project for damage assessments (Listing 4.5) and find the damaged element. In Listing 4.5, this corresponds with the variables '?el' and '?doc'. Results are included in Table 4.2. Optionally, other semantic details, such as type and size can be retrieved with this query as well.
3. Discover the Reference Registries in each partial project.
4. Query the Reference Registries for Reference Collections that aggregate a Reference with the correct source and identifier.
5. Find the aliases of the resulting Reference Collection and their References (and corresponding source, identifier and conformance).

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

Listing 4.5: 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.

?el	damageDistFM:inst_a003d5d4
?doc	https://dgfb.ugent.be/data/cac6d088

Table 4.2: Results of the query in Listing 4.5. The prefix damageDistFM corresponds with the result of the '?doc' variable.

Two strategies can be identified for retrieving the total set of federated References associated with the same concept. The first one makes use of a local aggregation of the Reference Registries. After allocating the Reference Registries for each stakeholder via their SPARQL satellite, the Reference Registries are fetched via the vault's LDP interface and merged in a client-side triple store that can be directly queried using a single SPARQL query (Listing 4.6). Some post-processing of the results will be needed to combine aliases and selectors into a single object that can be used for internal application logic.

```
SELECT * WHERE {
    ?rc a consolid:ReferenceCollection ;
     consolid:aggregates+ ?ref, ?otherRef .
    ?ref oa:hasSource <{source}> ; # known source
        oa:hasSelector/rdf:value "{value}". # known selector value
    ?otherRef oa:hasSource ?otherSource ;
        oa:hasSelector/rdf:value ?otherValue .

    OPTIONAL {
        ?rc consolid:aggregates ?alias .
        ?alias a consolid:ReferenceCollection .
}}
```

Listing 4.6: Recursive query to find all representations, identifiers and datasets for a given selected concept. The query is to be executed on a triple store that aggregates all relevant Reference Registries.

Using the demo scripts available on Github[8], it was found that a Reference Collection retrieval using cached Reference Registries of the iGent case study

[8] ConSolid demo; https://github.com/ConSolidProject/cde-satellite/tree/dissertation/demo. Accessed 2023-11-30.

generally takes about 0.22s, on a machine with the specifications listed in table 3.2 (Chapter 3) (100 iterations). Although optimisation of query time is out of scope for this dissertation, this proves the feasibility of the approach.

Again, it is possible to differentiate between clients that can hold state (e.g. browser applications) and clients that cannot (e.g. RESTful APIs): a browser application can instantiate an in-memory triple store and allow all interactions regarding reference discovery to be executed on this local store. Since a RESTful server will not maintain a state, it needs to setup this in-memory store with every request. Subsequent requests to SPARQL stores will in some cases be more efficient. This option is explained in the following paragraphs.

The basic queries given in Listings 4.7, 4.8 and 4.9 allow to find the references aggregated by the same Reference Collection. Consider an active project document which has been opened in a viewer application, similar to the situation depicted in Figure 4.3. The first query (Listing 4.7) is to be executed whenever a sub-document identifier is selected in this active document. In the case of non-discrete identifiers (such as pixel or voxel zones), existing identifiers can be preloaded to allow selection. The query is executed on the Reference Registry of the partial project 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. This query yields the Reference Collection's identifier and its aliases, which can now be used to find other representations in the vaults of the other consortium members (Table 4.3).

```
SELECT ?refcol ?local ?alias
WHERE {
    ?refcol a consolid:ReferenceCollection;
        consolid:aggregates ?reference .
    ?reference oa:hasSource <{document}> ; # known source
        oa:hasSelector/rdf:value "{identifier}" . # known identifier
    ?refcol consolid:aggregates ?local .

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

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

Listing 4.7: Query pattern to find the local references of a specific Reference Collection (given a known reference), and its potential remote aliases.

?refcol	https://dgfb.ugent.be/data/b677ce97#rc_bdb4526d
?local	https://dgfb.ugent.be/data/b677ce97#ref_a80854ae
?alias	https://b-b.be/data/f0c8cb37#rc_3de17fbe

Table 4.3: Concept identifier and representation identifiers for the results of the query in Listing 4.5.

This is illustrated in the query in Listing 4.8, which is executed on the same vault as the first query (Listing 4.7) – the retrieved *local references* (`?local`) are further resolved to find other documents (?doc) and identifiers (`?value`). When metadata resources are included in the set of queryable resources, metadata (`?meta`) of other local references can be included to further finetune the result set (e.g. based on `dcat:mediaType`). Results for this query in context of the iGent case study are given in Table 4.4. Additional metadata filters may be applied to only retrieve specific references. This can be done by attaching a metadata triple pattern to the query, an example of which is commented out in Listings 4.8 and 4.9.

?rc	https://dgfb.ugent.be/data/b677ce97#**ra**_bdb4526d *(abstract concept (alias in DGFB pod))*
?reference	https://dgfb.ugent.be/data/b677ce97#ref_a80854ae
?value	https://dgfb.ugent.be/data/cac6d088#inst_a003d5d4
?doc	https://dgfb.ugent.be/data/cac6d088
?meta	https://dgfb.ugent.be/data/3c1c9a18

Table 4.4: Results of the query in Listing 4.5. In this case, the prefix damageDistFM corresponds with the result of the '?doc' variable.

```
SELECT ?rc ?reference ?value ?doc ?meta
WHERE {
    BIND(<{referenceCollection}> as ?rc) # known Reference Collection
    BIND(<{reference}> as ?reference) # known reference

    <{reference}> oa:hasSelector/rdf:value ?value ;
        oa:hasSource ?doc .

    # if the set of queryable resources includes metadata records
    ?meta dcat:distribution/dcat:accessURL ?doc . # metadata filters
}
```

Listing 4.8: Recursive query to find all local representations, identifiers and datasets for a given selected concept.

Finally, in the third query (Listing 4.9), 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 (as it includes the vault URL). The results for this query are given in Table 4.5.

?rc	https://dgfb.ugent.be/data/b677ce97#rc_bdb4526d *(Reference Collection (alias in DGFB pod))*
?alias	<https://b-b.be/data/f0c8cb37#rc_3de17fbe *(Reference Collection (alias in b-b Pod))*
?reference	https://b-b.be/data/f0c8cb37#ref_8894fd07 *
?value	1uxwRB00H01B1i8VRQsVcL *(the geometric identifier in context of ?doc)* *
?doc	https://b-b.be/data/12008088 *(the URL of the glTF model)* *
?meta	https://b-b.be/data/e7d45e01 *(the metadata URL of the glTF model)* *
?rc	https://dgfb.ugent.be/data/b677ce97#rc_bdb4526d *(Reference Collection (alias in DGFB pod)*
?alias	<https://b-b.be/data/f0c8cb37#rc_3de17fbe *Reference Collection (alias in b-b Pod))*
?reference	https://b-b.be/data/f0c8cb37#ref_2ced93d3 *
?value	https://b-b.be/data/12008088#instance_6f8474e0 *(the semantic identifier in context of ?doc)* *
?doc	https://b-b.be/data/12008088 *(the URL of the semantic model)* *
?meta	https://b-b.be/data/6a4303c2 *(the metadata URL of the semantic model)* *

Table 4.5: Resulting concept alias and its two external references data on the scale of the federated project.
*** = identifiers not indicated in earlier listings, hosted in resources on the Pod of B-B.**

```
SELECT ?rc ?reference ?value ?doc ?meta ?alias
WHERE {
    BIND(<{referenceCollection}> as ?rc) # known Reference Collection
    BIND(<{alias}> as ?alias) # known alias
    ?alias consolid:aggregates ?reference .
    ?reference oa:hasSelector/rdf:value ?value ;
        oa:hasSource ?doc .
}
```

Listing 4.9: Recursive query to find all remote representations, identifiers and datasets for a given selected concept.

These steps are generally applicable for retrieving heterogeneous sub-document identifiers in aggregated projects, based on a combination of domain-specific queries (Listing 4.5) and ConSolid data patterns (Listings 4.7 and 4.8 and 4.9).

This Reference Collection retrieval pattern avoids the need to cache a union graph of the reference registries, and allows to further finetune the queries with filters for metadata records. Using the demo scripts available on Github[9], it was found that a Reference Collection retrieval without caching Reference Registries generally takes about 1s. However, this duration will vary with a different number of project vaults hosting an alias of this Reference Collection. The fact that in the prototype, all requests are being rerouted via the ExpressJS middleware between the client and the SPARQL satellite also increases query time. The queries used in this paper generally retrieve only one concept at the time, although optimisations 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 4.7 for every concept. Furthermore, a dedicated endpoint can be set up in the SPARQL satellite to perform the access control checking step only once for multiple queries to the same vault.

4.5 Implementation

A prototypical API was created to interact with Reference Aggregations and Reference Registries. As it makes use of the data patterns described in Chapter 3, this API depends on the lower-level implementation of *Daapi*. The API is available on Github[10] and NPM[11] as the Reference Aggregation API ('Raapi').

4.6 Conclusion

This chapter, described a set of data patterns that allow annotation and cross-document linking in a federated ecosystem. The existing framework of the Web Annotation Data Model formed the main input for the annotation part, and offered the definitions for sub-document selection of identifiers. Reference Collections show similarities with centralised linksets in existing multi-model specifications. However, some adaptations were made to make this work in a

[9] ConSolid demo; https://github.com/ConSolidProject/cde-satellite/tree/dissertation/demo. Accessed 2023-10-31.

[10] Raapi (Github): https://github.com/LBD-Hackers/raapi/tree/dissertation. Accessed 2023-10-30.

[11] Raapi (NPM): https://www.npmjs.com/package/consolid-raapi. Accessed 2023-10-30.

federated environment as well. Allowing alignment of Reference Collections with aliases in disparate Reference Registries, which can mutually aggregate each other, each Reference Registry can function standalone as well as be part of a federated project.

While multi-models address the linking of disparate project documents on a sub-document level, they do not address the fact that this alignment is still a human-driven process which sometimes requires intensive manual mapping [129, 170]. The activity of linking references to an alias of the same concept is not directly handled by the ecosystem, which only provides the data patterns to do so. Depending on the goal and size of the project, this may 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. In many cases, the fact that resources are linked will go by unnoticed by the end user. For example, any assignment of element properties and any classification activity can be categorised as a linking and enrichment activity. Also, a case of damage enrichment as documented in Section 4.4 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 unlikely, given the amount of existing concepts. For identifying many aliases of the same concept, mapping algorithms such as proposed in [109] 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. 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 thesis.

Another critical note is that Reference Collections are most stable in situations where bi-directional linking is possible. This stands in contrast with dataset collections, which may be established with unidirectional links to create a sub-vault hierarchy. Within the boundaries of a known project consortium, during the early project phases, the creation of a new alias can propagate through the known network, updating existing aliases to make the aggregation more robust. Registering backlinks on the vaults of other people will likely require some additional steps, as only the owners of a Reference Registry or their delegates will have the necessary writing permissions. A publishing mechanism will therefore be necessary to broadcast the alignment of an alias or the creation of an annotation related to a resource on another vault. Several technologies exist

that enable this, such as Linked Data Notification (LDN) or a CDE satellite with a dedicated endpoint. This topic will be further discussed in Chapter 6.

Table 4.6 gives a summary of the characteristics of a federated CDE (as defined in this dissertation), focusing on this chapter's topic of resource linking and annotation. The characteristics are related to the technologies that were identified as suitable for supporting them. When a technology is a 'data technology' (e.g., RDF), more details are provided regarding the data patterns that were developed or adopted.

Characteristic	Technology	Data Patterns
C6 - Cross-document Annotation	RDF (WADM)	WADM Annotations
C7 - Cross-document Linking	RDF (ConSolid)	Reference Collections / WADM Selectors

Table 4.6: Characteristics for a federated CDE (resource linking and annotation), corresponding technologies and data patterns as implemented in the ConSolid ecosystem.

4.7 Related Publications

This chapter contains edited fragments or concepts derived from the following publications:

- Jeroen Werbrouck, Pieter Pauwels, Jakob Beetz, and Erik Mannens. "ConSolid: a Federated Ecosystem for Heterogeneous Multi-Stakeholder Projects". In: *Semantic Web Journal* (2023). Accepted. URL: https://biblio.ugent.be/publication/8633673/file/8633674.pdf (accessed 2024-3-18).

Chapter 5
Data Validation

At this point, the data structures of the ConSolid ecosystem have been described. However, checks and balances need to be in place at various levels, to ensure the stability and trustworthiness of the ecosystem and the data that circulates in it. Several standards exist for container-based data exchange between stakeholders. In the ConSolid ecosystem data remains on a fixed location on the Web, and can be discovered for specific usage scenarios – hence, it is not really 'exchanged'. Still, international data exchange standards exist to ensure interoperability between applications and CDEs. Some of these standards focus on structuring project data into multi-models, and mainly target project metadata (e.g. ICDD [89]), while others are focused on project information (e.g. BCF API [29]). In the first part of this chapter (Section 5.1), the concept of Shape Collections is introduced. As an illustration, it will be shown how Shape Collections can be used to validate project metadata (Section 5.2). This will be done using the W3C recommendation SHApes Constraint Language (SHACL) [101].

Section 5.3 then deals with an entirely different type of validation, namely validating requesting agent's properties access control rights. In this section, the default ACL mechanism used in Solid will be extended with a framework for Pattern-based Access Control (PBAC), applicable to both visitors and datasets.

Relevant appendices for this chapter are:

- Appendix B (Semantic Web technologies (RDF, SPARQL, SHACL));
- Appendix C (Solid and federated authentication);
- Appendix I (The PBAC vocabulary).

5.1 Shape Collections

A Validation Resource can be registered on the vault just like any other ConSolid resource, namely as the combination of a metadata record and a distribution. In this dissertation, the focus lies on validation with SHACL. Just like a Reference Registry is an instance of `consolid:ReferenceRegistry`

(Section 4.1), a dataset with the purpose of validation will be an instance of `consolid:ValidationResource`. However, similar to `dcat:Dataset`s, registering shapes on a vault is not enough to actually integrate them in a project: they must be explicitly aggregated by the corresponding project catalogue.

```
# the project catalogue
bb:73a16f30 a dcat:Catalog ;
    dct:title "iGent-BB" ;
    dct:description "Partial project of the iGent project, provided by
     Bureau Bouwtechniek" ;
    dct:publisher <https://b-b.be/data/profile/card#me> ;
    dct:identifier "73a16f30" ;

    # a regular project dataset (cf. Chapter 3)
    dcat:dataset bb:c6a2a668 , dgfb:3c670851 .

    # a Shape Collection (subproperty of dcat:dataset)
    consolid:hasShapeCollection bb:ce07d756, # office-specific collection
                                dgfb:723ad1b4 . # project-wide collection

# Shape Collection dgfb:723ad1b4 (on the vault of the DGFB / client)
dgfb:723ad1b4 a dcat:Catalog, consolid:ShapeCollection ;
    # contains specific Validation Resources (dataset and distribution)
    dcat:dataset dgfb:9de26fa3, dgfb:41aee90b, dgfb:72f97754 .

# Shape Collection bb:ce07d756 (in a separate resource, but on the same
     Pod)
bb:ce07d756 a dcat:Catalog, consolid:ShapeCollection ;
    dcat:dataset bb:a72269b8 .
```

Listing 5.1: Registration of a Shape Collection in a ConSolid project catalogue.

The concept of *Shape Collections* (`consolid:ShapeCollection`) is now introduced as a subclass of `dcat:Catalog`s. This way, they can be easily embedded in the existing ConSolid fabric, linking them to an existing ConSolid project via the `consolid:hasShapeCollection` property, a subclass of `dcat:dataset`. Starting from a Shape Collection catalogue, the same notion of a 'property path of arbitrary length' (in SPARQL denoted with a '+'-sign) can be used to discover the underlying shapes, both on the main vault of the Shape Collection and elsewhere on the Web. This allows to re-use shapes (and Shape Collections) provided by other project partners or regulatory instances (e.g. standardisation bodies or governments), in the same way ecosystem-external project datasets can be included by creating a metadata record pointing to a distribution on the Web (Section 3.3.3). Listing 5.1 illustrates this with a Shape Collection applied to a regular ConSolid project catalogue. Note that the only

required access right for a validation process is to have `acl:Read` permission: a validation activity does not have any impact on the content of the checked resources – it is only responsible for generating a report. This report can then serve as input for a data validation activity by any agent with `acl:Write` permissions.

Just like with the Dataset Collections (Section 3.3.1), especially in early project phases, it is important to keep a unified approach between the multiple stakeholders contributing to the project. Similar to the aggregation of a main project catalogue maintained by the project manager (see Figure 3.3), the federated aspect of DCAT catalogues allows each partial project to point to a common Shape Collection (Figure 5.1). Such setup can be easily verified because the partial projects of each stakeholder will be aggregated by the main project catalogue anyway. This is a minimum set of shapes to which all project datasets, hosted by anyone in the consortium must comply. Partial projects can still indicate other Shape Collections, e.g. to make datasets conform to company-specific requirements.

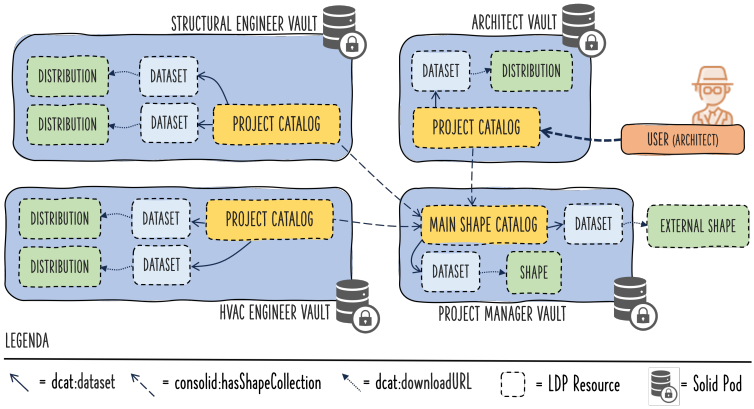

Figure 5.1: Different partial projects share the same common Shape Collections: one provided by the consortium itself, one provided by a standardisation body. Individual stakeholders can still extend this agreed-upon minimum set with office-specific data policies.

5.2 Metadata Validation

A first case where it is important to apply validation is metadata. To illustrate the validation of ConSolid metadata records on an upper-level, the DCAT-AP of the EU will be used. The DCAT vocabulary has been adopted by governments

all over Europe to increase consistency in the metadata descriptions of public datasets [98]. In this context, DCAT application profiles (AP) were developed, both in a EU-wide context [45] and regionally [47]. SHACL shapes are available to validate a metadata record against a DCAT-AP.

This means that for instances of `dcat:Catalog`, only four properties need to be minimally included, namely `dct:title`, `dct:description`, `dct:publisher` and `dcat:dataset` (i.e. 'empty' catalogues are not allowed). Other properties, such as modification date (`dct:modified`) or homepage (`foaf:homepage`) are optional. An instance of `dcat:Dataset` must minimally have values for `dct:title` and `dct:description`, while a `dcat:Distribution` only needs a value for the property `dcat:accessURL`. This information can be included in the overall project by aggregating the resource in Listing 5.2 in the shared Shape Collection (see Figure 5.1). This is a metadata record that references the actual Shape Collection provided by the European Commission – Joinup [56] as a project-external distribution (see also Section 3.3.3).

```
dgfb:9de26fa3 a dcat:Dataset, consolid:ValidationResource ;
    dct:description "DCAT-AP for DCAT data
    dct:title "DCAT-AP" ;
    dcat:distribution dgfb:c652cfe1 .

dgfb:c652cfe1 a dcat:Distribution ;
    dct:conformsTo <https://www.w3.org/TR/shacl> ;
    dcat:accessURL <https://raw.githubusercontent.com/SEMICeu/DCAT-AP/
     cea5a96bb4a6f120c20b7a2b3fb4d86bcd725952/releases/2.0.0/Draft/dcat-
     ap_2.0.0_shacl_shapes.ttl> .
```

Listing 5.2: Metadata record including the DCAT-AP (v2.0.0) as provided by the European Commission - Joinup.

Furthermore, different shapes may be applied to different subtypes of metadata records. For example, when the ConSolid infrastructure is to be used as a CDE, it makes sense to align project datasets with the requirements for a container in DIN SPEC 91391 [49]. A Shape can be created to ensure every dataset of type `consolid:ProjectResource` has the right properties attached to be a valid CDE container (Listing 5.3).

```
dgfb:OpenCDEShape
  dct:description "This shape validates instances of consolid:
     ProjectResources against the requirements for OpenCDE containers as
     described in DIN SPEC 91391." ;
  a sh:NodeShape ;
  sh:targetClass consolid:ProjectResource ;
  sh:property [
    sh:path dct:identifier ;
    sh:minCount 1 ;
    sh:maxCount 1 ;
    sh:severity sh:Violation ;
    sh:resultMessage "OpenCDE containers require an 'Id' field. In
     ConSolid, this resolves to dct:identifier of a dcat:Dataset."@en ;
  ] ;
  sh:property [
    sh:path rdfs:label ;
    sh:minCount 1 ;
    sh:severity sh:Violation ;
    sh:resultMessage "OpenCDE containers require a 'Name' field. In
     ConSolid, this resolves to rdfs:label of a dcat:Dataset."@en ;
  ] ;
  sh:property [
    sh:path rdf:type ;
    sh:minCount 1 ;
    sh:resultMessage "OpenCDE containers require a 'Type' field. In
     ConSolid, this resolves to rdf:type of a dcat:Dataset. The options
     are provided by the CDC ontology (<https://w3id.org/cdc#>)."@en ;
    sh:severity sh:Violation ;
  ] ;
  sh:property [
    sh:path doap:revision ;
    sh:minCount 1 ;
    sh:maxCount 1 ;
    sh:resultMessage "OpenCDE containers require a 'Revision' field. In
     ConSolid, this resolves to doap:revision of a dcat:Dataset."@en ;
  ] ;
  sh:property [
    sh:path consolid:projectId ;
    sh:minCount 1 ;
    sh:maxCount 1 ;
    a consolid:RuntimeProperty ;
    sh:resultMessage "OpenCDE containers require a 'projectId' field. In
     ConSolid, this cannot be mapped to a specific property, as project
     scalability dictates that a dataset may be part of many (aggregated)
     projects. This property should be derived at runtime, before
     validation."@en ;
    sh:severity sh:Violation ;
  ] .
```

Listing 5.3: SHACL shape for verifying consolid:ProjectResource instance must also be a valid OpenCDE container with given properties.

Another shape is included in Listing 5.4, stating that every Project Resource must have an associated publication status (`ex:publicationStatus`), conform ISO 19650.

```
dgfb:PubStageShape
  dct:description "This shape validates whether instances of consolid:
     ProjectResources have a publication status attached, as described in
     ISO 19650." ;
  a sh:NodeShape ;
  sh:targetClass consolid:ProjectResource ;
  sh:property [
    sh:path ex:publicationStatus ;
    sh:minCount 1 ;
    sh:maxCount 1 ;
    sh:in ("WorkInProgress" "Published" "Archived" "Shared") ;
    sh:resultMessage "Every resource must have an ISO19650 publication
     status attached."@en ;
  ] .
```

Listing 5.4: SHACL shape mandating that every consolid:ProjectResource must also have a publication status, mapped to instances of consolid:ProjectResource.

Both shapes can be included in the project with a metadata record similar to the one in Listing 5.2. Some fields, however, conflict with the ConSolid data patterns due to the graph-based, federated structure of ConSolid projects and the centralisation aspect of information containers in the AEC domain. For example, OpenCDE containers require a 'projectId' field. In ConSolid, this cannot be mapped to a specific property, as the requirement for project scalability dictates that a dataset may be part of many (aggregated) projects. This property should be derived at runtime, before validation - in that case, the project ID will be derived from the indicated project access point (see Section 3.3.1).

In Chapter 6, it will be discussed how a high-level proxy service to the vault (the ConSolid API) can fetch these shapes, on the one hand in order to inform the client about the necessary metadata fields when creating a project or registering a dataset; on the other hand to effectively validate metadata structures during a data validation process.

5.3 Extended Access Control Validation

As discussed in Chapter 3, the Solid ecosystem includes two access control mechanisms, both of them mainly based on the WebID of a visitor. Compared to the Web Access Control (WAC), the upcoming Access Control Policy (ACP) [23]

specification includes the option to apply further restrictions or allowances, based on client properties like the issuer of the identity and the client through which the request is made. Furthermore, the working draft of ACP defines properties that can be used to create policies applying to any resource with a specific tag attached, although it does not define yet where this tag statement happens. At the moment of writing, such tags are class-based, but in order to flexibly address the double patchwork of projects, contractors, subcontractors, employees etc. (Chapter 2), more expressive extensions are needed for access control, such as sets of properties or even sub-graphs with more complex data patterns. Furthermore, the ACP specification currently does not include the notion of trusted authorities on specific (graph-based) statements, or a system for access delegation. In this dissertation, an alternative approach is devised which takes these topics into account. This approach does not necessary conflict with the existing access control specifications, but may occur on a higher level, i.e., applied by a satellite service. Future work must determine the compatibility of PBAC with standardised specifications such as ACP.

As indicated in [124], a decentralised environment for hosting building data benefits from more advanced access control mechanisms, such as a Role-Based Access Control (RBAC) mechanism or a property-based one (also called Attribute-Based Access Control (ABAC)), which includes the possibility to express 'arbitrary' access rules, such as:

- All employees of company X working in project Y;
- Inhabitants of the respective building;
- The facility manager of Project Z.

An analogy to describe such context- or property-based approach in a general sense is that one might not be able to name firefighters or paramedics beforehand, but they can be given access if they are able to prove their function, e.g. with valid certificates. In context of the construction industry, this could be a (partial) delegation of access rights from contractors to subcontractors. Or, before getting access to a certain resource, a client should prove she is involved in the project as a 'leading architect' and at the same time demonstrate her membership of a recognised association of architects.

Ideally, such patterns should be reusable by anyone in the ConSolid ecosystem. In other words: implicit references should be used rather than explicit ones. I.e. 'all employees of B-B' may already be expressed to some extent using ACL agent groups; 'all employees of the company that is responsible for

architectural design of the project this resource belongs to' cannot. For such scenario to happen, at least two certificates may be needed: one stating that the employee works for B-B (signed by B-B or one of its 'full' delegates) and another one indicating that B-B is indeed involved in the project (e.g., signed by the commissioner). Although most of these patterns can be expressed using the default ACL implementation in Solid (e.g. by hardcoding the WebIDs of these people into agent groups), complex patterns that pose multiple requirements to visitors will be expressed and verified with more ease using a *pattern-based* approach.

In this framework for 'Pattern-based Access Control' (PBAC), two components are needed. Firstly, a technology for making, exchanging and validating assertions is required. Secondly, these assertions need to be related to access-control policies: 'if {client} can prove that they have {propertySetX} as asserted by {trustedparty}, they get *read* access to resources with {propertySetY} as asserted in the data vault where the resource resides'. In the following sections, these two components will be discussed in greater detail. The resulting Access Control framework is meant to extend beyond typical scenarios of the AECO industry and be generally applicable, although regulating access to information within construction projects remains the primary target. Any RDF definitions in this context will be included in the PBAC vocabulary [187]. This dissertation does not propose a full framework, but rather identifies high-level strategies for achieving a pattern-based access control framework. These strategies are intended as a basis for further research.

5.3.1 Requirements to the Requester

By default, the Web is an open framework, where people can express anything they want. For a pattern-based access control to function properly, a mechanism to prove the statements thus needs to exist: how to know that the assigned properties are valid, without resolving to a centralised ecosystem? Typically, four actors are needed there: an issuer, a holder, a verifier and a Verifiable Data Registry (VDR) [163]. The issuer must be a trusted authority on the topic of the assertion, so the verifier has good reasons for believing the statements made in it. The holder can be the subject of the assertion, but does not necessarily need to be. A VDR is used as a common registry to make sure a common schema is used, improving data interoperability. Asymmetric encryption using a public-private key pair allows verification that the assertion was indeed made by the trusted authority.

Earlier work by the author of this thesis based upon the use of nanopublications [102] for establishing a PBAC framework [198]. However, an alternative that has recently gained the status of a W3C recommendation is the concept of Verifiable Credentials: digital, cryptographically signed assertions that represent a claim, qualification, or attribute about an individual, allowing these individuals to share and prove specific information without disclosing unnecessary personal details. Authorship of a VC is stated using a Decentralised Identifier (DID) [165]: a decentralised, self-sovereign and persistent identifier on the Web. DID is a W3C recommendation since 2022, but has not been implemented as part of the Solid project [203]. Full coverage of the details of VCs is considered beyond the scope of this dissertation. As VCs evolved into an official Web recommendation, they are considered a more robust choice than nanopublications. This being said, after the assertion (stated in RDF) has been verified, the technology that was used to make it verifiable does not play a role anymore in the PBAC framework. For demonstration purposes, however, this dissertation will just use basic graphs which are cryptographically signed and verifyable as part of a JSON Web Token (JWT).

When considering PBAC independently from ConSolid, it is possible that an agent has lots of credentials, and that it is unclear which credentials need to be presented to a specific Solid server in order to be granted access to a specific resource. To prevent a waste of precious bandwidth and server resources it is undesirable to present them all along with the request. Applications may therefore impose some negotiation steps [96]. Different strategies may be used here, depending on the level of trust between visitor and owner of the resource. The most open strategy here is to refer to a public shape, a more controlled one could be a step-by-step release of requirements. Relating this to construction projects, such step-by-step approach may balance the need to keep (access) information internal to the project and the need to explain to stakeholders why they cannot access certain information, and who they should contact if this is to be changed. After a first requirement is met ('the visitor is a stakeholder in the project ...'), the server could choose to 'release' the other shapes, thereby providing specific information about any other conditions that need to be fulfilled ('... with the task of performing a structural analysis'). As the SHACL specification includes the possibility to generate detailed validation reports, textual as well as machine-readable explanations may be sent to a stakeholder whose request just got rejected, which is one of the challenges mentioned in [96].

5.3.2 Requirements to Project Data

Similar to requirements made to the visitor, a rule can also apply to all resources with specific metadata properties. For example: 'all project resources that are JPEG images', or 'all resources that have a publication status *shared*'. Combined with the requirements for the requesting agent, this yields concise and powerful rules: 'any resource with publication status *shared* can be shared with anyone who can prove they are part of the project, and edited by anyone who can prove they are employed by office X.' However, this approach inverts the access control approach maintained in Solid, as access control rules are no longer tied to specific resources, but become part of a query flow – coming at the cost of response speed. To achieve this in an acceptable timeframe, a queryable union graph of the vault needs to exist. Such functionality is offered by the SPARQL satellite described in Chapter 3. Conceptually, SHACL shapes as well as SPARQL queries may be used to verify whether a dataset conforms to a requirement – where the strengths of SHACL will typically lie in expressing cardinalities and restrictions to specific nodes, SPARQL will be more useful in activities of pattern matching, especially when more complex graph patterns are involved. SHACL-SPARQL [101] incorporates these advantages of SPARQL to use in more complex validation scenarios.

Since the requirement to project data results in a boolean answer, either SHACL, SHACL-SPARQL and SPARQL ASK can be used, depending on the imlementation. Since the SPARQL satellite does not implement SHACL validation, the example rule will use a SPARQL ASK query (`pbac:askQuery`). In PBAC, the resource URL can be dynamically injected in the query prior to execution, replacing the '*$resource$*' in the query string (see Listing 5.5).

5.3.3 Access control rules

Both project data requirements and visitor requirements can be integrated in a pattern-based access control (PBAC) framework. The rules are registered on the data vault as distributions of `pbac:AccessResource`s (metadata records), and are categorised as `pbac:DynamicRule`s. Along with listings of the requirements and the trusted authorities, a dynamic rule (`pbac:DynamicRule`) contains information about the ACL modes it grants. If the requirements put forward by a pattern-based access rule are met, ACL modes (Read, Write, Append, Control) will be allowed for a given visitor.

Relating an access rule to the properties of the visitor is done via `pbac:visitorRequirement`, which refers to one or multiple (local or re-

mote) SHACL shapes. For requirements to resources, the property `pbac:resourceRequirement` is used. Both are subclasses of `pbac:Requirement`. The difference between an 'inclusive' rule (a visitor or resource needs to conform to only one of the mentioned shapes), or an 'exclusive' one (all shapes need to be met before the visitor is granted access), may be established by linking `pbac:visitorRequirements` to a locally defined shape, which can combine different (possibly remote) shapes through various Boolean operators (`sh:and` or `sh:or`).

Section 5.3.1 mentions the need for trusted authorities to be indicated. The trusted authorities can be included into specific shapes using `pbac:hasTrustedAuthority`. This allows to indicate authority for certain statements while not trusting them to say anything. A trusted authority may be explicit, resolving to a WebID. Another option is to implicitly refer to a trusted authority, i.e. to a SHACL shape to which the issuer must comply, in combination with another trusted authority (`pbac:issuerRequirement`). Ultimately, a chain of implicit mentions of trusted authorities must resolve to an explicit authority. For instance, in Listing 5.5, the visitor requirement (`arcadis:fef427c3`) can be signed by two explicitly mentioned authorities, i.e., the owner of the vault (Arcadis) and the client (in our example this resolves to UGent, via the DGFB). The requirement may also be signed by every authority that can prove they comply to a specific shape, i.e. an *implicit* authority. In this case, the same shape is used for visitors and implicit issuers - everyone who participates in the project can issue a certificate to employees or subcontractors that they are part of the project, too.

A caveat for credentials that specifically refer to ConSolid projects (and is thus not related to PBAC) is that there is no fixed identifier for 'the project' (see Chapter 3) – at this point only the partial projects have an identifier in the form of a URL. This can be fixed by either issuing certificates for each partial project (possibly combined into one larger certificate), or agreeing on a common identifier for the project rules, that does not need to resolve to actual resources but allows to identify the project (e.g. a GUID or URI). This allows reuse of credentials and shapes within the consortium network.

```
# the metadata record of the dynamic rule
arcadis:432d2d04 a dcat:Dataset, pbac:AccessResource ;
    dcat:distribution arcadis:1c421255 .

arcadis:1c421255 a dcat:Distribution ;
    dcat:accessURL arcadis:1c421255 .

# the contents of the rule
arcadis:ReadRule a pbac:DynamicRule;
    acl:mode acl:Read ;
    rdfs:comment "Allows employees of offices that participate in the project
     to READ the resources of interest.";
    pbac:visitorRequirement arcadis:fef427c3 ;
    pbac:resourceRequirement arcadis:d9bd0cb6 .

# a shape on the vault
arcadis:fef427c3 a sh:NodeShape, pbac:VisitorRequirement, pbac:
     IssuerRequirement ;
    sh:targetClass pbac:Visitor, pbac:Issuer ;
    pbac:hasTrustedAuthority arcadis:authority1, arcadis:authority2, arcadis:
     authority3 ;
    sh:property [
        sh:path consolid:participatesIn / dct:identifier ;
        sh:hasValue "0d1ffe69" ; # project identifier
    ] .

# a trusted authority can be registered in a separate resource
# the issuer is explicitly mentioned by their WebID
arcadis:authority1 a pbac:trustedAuthority ;
  dct:identifier "https://arcadis.com/data/profile/card#me"^^xsd:anyURI .

# the issuer is explicitly mentioned by their WebID
arcadis:authority2 a pbac:trustedAuthority ;
dct:identifier "https://dgfb.ugent.be/data/profile/card#me"^^xsd:anyURI .

# any implicit issuer must conform to the original shape for participants
arcadis:authority3 a pbac:trustedAuthority ;
    pbac:issuerRequirement arcadis:fef427c3 .

# the resource requirement is executed on the union graph of the vault
# to produce a valid SPARQL ASK query, the resource URL needs to be injected
     at the place of $dataset$
arcadis:d9bd0cb6 pbac:ResourceRequirement ;
    rdfs:comment "The resource must be a dataset or distribution in the
     project with ProjectId arcadis:0d1ffe69 with publication status 'shared
     '." ;
    pbac:askQuery """ASK WHERE {{
        arcadis:0d1ffe69 dct:identifier "0d1ffe69" ;
          dcat:dataset+ $resource$ .
          $resource$ ex:publicationStatus "Shared" .
        } UNION {
          arcadis:0d1ffe69 dcat:dataset+/dcat:distribution ?distribution .
          ?distribution dcat:downloadURL $resource$ ;
            ex:publicationStatus "Shared" .
}} """ .
```

Listing 5.5: Example ACL file enhanced with a PBAC rule

5.3.4 Workflow

The following workflow is executed when an agent requests access to a specific resource on a data vault. Firstly, the default ACL rules that apply to the resource are checked. If these grant access to this resource, its content can be returned immediately. This avoids unnecessary checking of complex PBAC rules. If access is not granted, but PBAC certificates are sent along with the request, a verification will take place using the technologies described above. A rule can be retrieved using the query in Listing 5.6, which is sent to the SPARQL satellite of the vault containing the requested resource.

```
SELECT * WHERE {
    ?rule a pbac:DynamicRule ;
        acl:mode <{mode}> ;
        pbac:resourceRequirement ?rr ;
        pbac:visitorRequirement ?vr .
    ?rr pbac:askQuery ?askQuery .
    ?vr pbac:hasTrustedAuthority ?authority .
    ?authority ?p ?o .
    VALUES ?p { dcterms:identifier pbac:issuerRequirement }
}
```

Listing 5.6: Query to the SPARQL satellite, retrieving any dynamic rules for the requested access mode. A dynamic rule includes visitor requirements and data requirements.

During the validation step, SHACL shapes in each *relevant* rule are validated against the verifiable union of received credentials. A rule is relevant when it yields an ACL mode that has not been granted already (e.g. because the requester is already mentioned explicitly in the ACL graph for the requested ACL mode). A union graph will be made of all valid certificates. This union graph may vary per rule, since authorities may differ per rule. A certificate is valid for a specific rule when:

1. The signature on the JWT can be verified with the public key of its issuer;
2. The issuer is an authority (explicit or implicit) for the rule to be checked.

As the `sh:TargetClass` (i.e. the nodes against which the shape constraints are checked) of the SHACL shape related to a `pbac:VisitorRequirement` applies the rule to all instances of `pbac:Visitor`, a triple that classifies the WebID of the visitor as an instance of `pbac:Visitor` is be added to the union graph at runtime.

The PBAC endpoint implemented in the ConSolid satellite (Chapter 6) uses the following checking procedure, based on certificates sent along with a custom 'PBAC' header:

1. Verify the signature on the certificates sent along with the request as a JWT, using the public key exposed by the issuer of the certificate ;
2. Check whether there exist any `pbac:DynamicRule`s that have the requested mode as `acl:mode`, if this mode has not been granted in the original ACL graph. With the same query (Listing 5.6), resource requirements, visitor requirements and their associated (explicit and implicit) authorities can be retrieved.
3. Withhold only those rules that apply to the requested resource, using SPARQL ASK query linked to the `pbac:ResourceRequirement`. Multiple rules can co-exist and be evaluated independently. A discussion on conflicting rules is out of scope for this dissertation.
4. To validate the remaining rules:
   (a) Check the issuers of each certificate. If the issuer is an explicit trusted authority for the rule, the statements in the certificate may be added to the eventual data graph that will be checked against the shape of the rule.
   (b) If the issuer is an implicit trusted authority, check if there are any certificates *about this issuer* that were signed by an *explicit* authority – as indicated in the `pbac:issuerRequirements` of the implicit authority (see Listing 5.5). The `pbac:issuerRequirements` may resolve to another `pbac:Requirement`, or to the original one. The set of certificates about this authority must be joined and validated against the issuer requirement. If this validation is successful, any certificates signed by this implicit authority can be added to the eventual data graph.
   (c) Validate the joint set of valid certificates against the `pbac:VisitorRequirement` of the rule. Prior to checking, a statement that the visiting agent is an instance of `pbac:Visitor` needs to be added. If this validation succeeds, the access mode is granted. Else, the next applicable rule can be checked.

This workflow is illustrated in Figure 5.2.

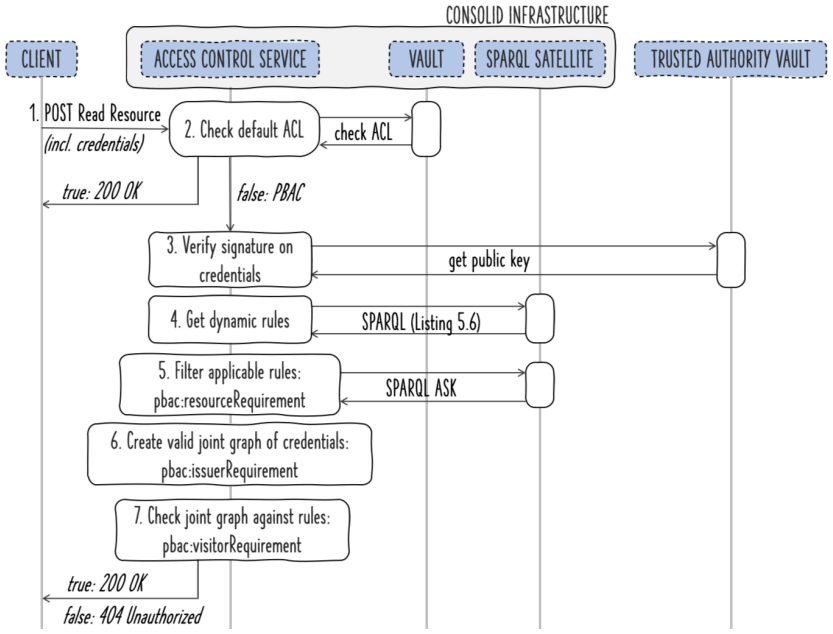

Figure 5.2: Sequence of actions to verify a client's access rights using PBAC. This is based on valid certificates, implicit and explicit trusted authorities and dynamic rules.

This workflow also implies that the higher the amount of `pbac:DynamicRules`, the longer the request will take. The same holds for the amount of certificates, and the amount of authorities to be checked. Performance optimisation for this workflow is out of scope for this dissertation, as the goal is semantic flexibility rather than quick response time. As an indication of the duration of a PBAC request, the case study explained in the next paragraph was verified using the ConSolid satellite (Chapter 6). The eventual PBAC request in this scenario takes about 1s.

5.3.5 Case Study

As a tangible example of the above-described workflow, let us consider the following situation, related to the actors of the iGent example used in Chapter 3 and 4. Alice, who works for Bureau Bouwtechniek (B-B), has two certificates. The first one states that she is indeed employed by B-B and is signed by the WebID associated with B-B. The second one is signed by the assigning party in the project (DGFB) and states that B-B is a member of the project.

When loading her project interface, Alice wants to check whether there are any clashes between the structural model provided by Arcadis and the architectural model provided by B-B. On the vault of B-B, a rule exists that indicates that anyone who is employed by B-B (B-B acts as the trusted authority) can read and edit any resources on the B-B Pod that have publication statuses `WorkInProgress` or `Shared`, but only read those with status `Published` or `Archived`. The vault of Arcadis, on their end, contains a rule which states that people who can prove that they are part of the project have read access to datasets that have a publication status `Shared` (see Listing 5.5).

The query in Listing 5.6 queries the vault for dynamic rules, which may combine visitor requirements and resource requirements. The identity of the authority is hardcoded as the WebID of the DGFB, which takes the role of the client in the project consortium. This query returns the Visitor Requirement and the Resource Requirement defined in Listing 5.5, including the ASK query. Before executing the ASK query, the parameter $resource$ should be replaced by the URL of the structural model (distribution). The metadata of the structural model indicates that it indeed has a publication status `"Shared"`, so the ASK query resolves to true; the Resource Requirement is met. As Alice is in possession of the necessary credentials, the Visitor requirements are also met. Alice can fetch the structural model and load it into the BIM authoring tool of her choice.

This case study can be replicated by executing the steps described at https://github.com/ConSolidProject/cde-satellite/tree/dissertation/demo/PBAC, using the ConSolid satellite described in Chapter 6.

5.4 Conclusion

In this chapter, the topic of validation was integrated into the ConSolid ecosystem. A first case was the validation of project (meta)data, in order to plan its use for specific scenarios. By integrating the concept of Shape Collections, a project consortium can set its own requirements for internally published data. At the same time, office-specific requirements can be included in the set of requirements. Of course, the more requirements a dataset should adhere to, the more stringent the data creation.

The validation framework in ConSolid allows services (such as the ConSolid satellite which will be discussed in Chapter 6) to take project-specific requirements into account for data creation and adaptation activities. When specific data usage or transformation scenarios are foreseen in the project planning,

Shape Collections and Validation Resources make sure the required properties will be present upon creation. When new Validation Resources are added to a project that is ongoing, non-compliant datasets can be identified and corrected in a data sanitation process.

Independent from validating the presence of (meta)data properties, a prototypical framework was introduced for pattern-based access control (PBAC). Also in this context, the SPARQL satellite (Chapter 3) proved its worth, as the union graph of a data vault via the SPARQL satellite allows to establish rules for visitors as well as the resources they request in a flexible, implicit way. The PBAC framework does not yet cover the entire complexity of pattern-based access control. For example, the topic of conflicting rules, or hierarchies of rules is considered out-of-scope for this thesis, but may be covered in future research. However, the framework introduces a novel approach to access control which complements and extends existing access protocols in Solid. Neither the validation of (meta)data and the PBAC framework are part of the Solid Specifications, so a proxy service is required to interact with data on the Pod. Chapter 6 will include the setup of a proxy service that includes this functionality.

5.5 Related Publications

This chapter contains edited fragments or concepts derived from the following publications:

- Jeroen Werbrouck, Madhumitha Senthilvel, Jakob Beetz, and Pieter Pauwels. "A checking approach for distributed building data". In: *31st forum bauinformatik, Berlin: Universitätsverlag der TU Berlin*. 2019, pp. 173–81. URL: https://biblio.ugent.be/publication/8667508/file/8667516.pdf (accessed 2024-3-18).
- Jeroen Werbrouck, Ruben Taelman, Ruben Verborgh, Pieter Pauwels, Jakob Beetz, and Erik 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. URL: https://ceur-ws.org/Vol-2636/09paper.pdf (accessed 2024-3-18).

Chapter 6
Middleware Services

The infrastructure discussed in Chapters 3 and 4 allows a multi-vault, heterogeneous and interconnected network of project resources to be formed. However, all interactions currently take place via LDP or SPARQL. These technologies are too low-level to enable appropriate usability as a CDE. In this chapter we identify higher-level interfaces that are capable of interpreting the data in the intended way, without blocking the potential for data expressivity at storage level. First, the main components that enable this are identified, i.e., Aggregators and Adaptors (Section 6.1). Then, we take a look at the data exchange flow between vaults, middleware and clients, which may serve as a template for almost any specific interaction (Section 6.2). In Section 6.3, the ConSolid API is discussed as an example of a functional satellite, which facilitates the high-level interactions mentioned above. The topic of RDF Aggregators, which allow to query the RDF data in a federated catalogue in an access-controlled manner, is discussed in Section 6.4. Finally, the concept of Adaptors is illustrated with a mapping of the ConSolid metadata structures to the industry standards ICDD (ISO 21597) and the BCF API [27] (Section 6.5).

Background technologies for this chapter are introduced in the following appendices:

- Appendix B (Semantic Web technologies (RDF, SPARQL, SHACL));
- Appendix E (Industry Containers - ICDD);
- Appendix G (Sub-document identifiers for different document types);

6.1 Components

The following components are essential for a federated ecosystem that is effectively useful as a CDE, and are thus labelled as characteristics:

1. **C 8:** ***Aggregators****: Our envisaged CDE does not include a single storage point - instead the Web is considered to be the CDE. Thus, at some point, the information needs to be aggregated (client-side or by a middleware service). The ecosystem must allow temporary and synchronised aggregation of its content into central access points intended for reading data.*

2. **C 9:** ***Adaptors****: Adaptors fulfil the task of reconfiguring generic (reusable) data patterns in a format readable by external applications. These may be application-specific or provide domain-specific standardised views on the data, useful for multiple applications. Adaptors can be micro-services or algorithms that are directly embedded in client application code.*

A single data access point will often be expected, either again for compatibility reasons, or for the sake of querying performance. This role is fulfilled by Aggregators (C8), which aggregate federated data and expose it in a specific way. Aggregators can host synchronised copies of project data in order to speed up the retrieval and querying process, or fetch and serve the data dynamically. The concept of Aggregators as a network of query and reasoning agents compatible with the Solid ecosystem is an active and complex field of research [121, 175]. This dissertation only acknowledges the necessity of an Aggregator component in the federated CDE; a proof-of-concept will be included to illustrate the role of Aggregators in the ecosystem. Aggregators may be included in middleware services, or directly integrated in client-side applications.

Reusability is one of the main reasons to keep data in a generic and structured way. However, existing tools and standards often expect a specific schema or data format, which seldom aligns with generic structured data. Adaptor services (C9) facilitate the conversion from structured, interconnected data to the formats expected by client-side tools that, using ISO 19650 terminology for CDEs, have a 'lower maturity level' (Chapter 2). In a federated ecosystem, Adaptors will often be combined with Aggregators to dynamically expose the data in the desired form. This way, they can offer a window to the federated data, which complies with a standardised or proprietary API.

The Aggregator-Adaptor combination can serve multiple purposes, some of which will be discussed in this chapter:

1. To facilitate a functional CDE layer on top of the existing infrastructure. Such layer may take the form of a *functional* satellite, differing from *storage* satellites as introduced in Chapter 3. A functional satellite will allow more complex interactions with one or more data vaults, interactions that go well beyond the ones offered by the LDP interface to a Pod. In the case of the ConSolid ecosystem, this will, amongst others, include CRUD interactions with project catalogues and datasets, Reference Collections and Annotations.

2. To offer a synchronised SPARQL endpoint for the entire project, given a specific project access point. This purpose only implements the Aggregator functionality and is largely performance-oriented – although providing data storage redundancy may be a reason as well.

3. To facilitate industry-specific APIs on top of the (filtered) resources on a data vault, or even on the entire federated project – allowing interaction with e.g. BIM authoring tools. This may be a standardised view on the project or on specific files, but mappings into proprietary formats or custom data structures and APIs are possible as well.

First, the general flow of requests and data between the data ecosystem, middleware services and clients will be discussed. Then, the first purpose is illustrated by describing the ConSolid Satellite, effectively rendering the ecosystem useful as a basic CDE. The second purpose is implemented using an in-memory triple store that synchronises with the (federated) project files it is allowed to see. The third purpose will be illustrated with two cases: a mapping of a federated ConSolid project to an ICDD-compatible dump archive and a demonstration of interacting with federated projects through the BCF API.

Different services may be chained to facilitate a specific flow of information. However, a longer chain of services often means more back-and-forth requests, and consequently a negative impact on performance. For example, a BCF API service may base upon ConSolid satellites for its discovery of project data, which increases the number of requests with the amount of ConSolid satellites. The impact on performance will not always be negative: when an Aggregator SPARQL endpoint is available, a ConSolid API can use this to speed up querying, instead of sending requests to individual SPARQL satellites. Nothing prevents multiple purposes to be implemented in a single service as well – especially if the implemented functionality is based on open standards. For example, the above-mentioned chaining of a BCF API and a ConSolid API can be avoided with the creation of a service that implements both.

Theoretically, the service layer may be extended with other services, such as data cleaning and validation services, services for geometric computations, notification services etc. However, apart from performance challenges, services in the chain will likely require different data structures, requiring more Adaptors, with increasing interoperability challenges and risks for incorrect conversions. The experimental validation of such service networks based on the ConSolid ecosystem is considered out of scope for this dissertation, but should be considered in future research.

6.2 Interactions between vaults and middleware

Whether the Aggregator-Adaptor service is implemented client-side, as middleware or as a satellite, does not really impact the flow of information exchange. It does impact, however, authentication and trust. At the time of writing, a fully specified delegation protocol implemented in the Solid ecosystem is still work-in-progress. The Community Solid Server [173] supports OAuth Token Exchange [93], but this feature is not included in the official Solid protocol [33]. This means that the current most secure options to authenticate to a Solid Pod are setting up a dedicated satellite (e.g. at the side of the Identity Provider (IDP)), or initialising a server which facilitates these interactions (e.g. at office level). Depending on the intended permissions for the service, either a dedicated WebID or the credentials of the vault owner may be used for authentication. This immediately illustrates why it is better to keep your satellites close (and why they are called satellites). The first option allows to limit the resources the service has access to, but requires to adapt access control rules for all affected files.

The interaction patterns where a client interacts with the federated ecosystem using an Aggregator and Adaptor follow a systematic flow. As SPARQL satellites (Section 3.2.4) offer the most complete views on metadata resources in project Pods, they will serve as the primary interfaces to feed Aggregators and Adaptors. The interaction flow consists of the following steps – 'vault' represents here the entire infrastructure of Pod, SPARQL satellite and ConSolid satellite (section 6.3):

1. The client sends an authenticated request to the middleware.
2. The middleware service extracts the WebID from the authenticated request.
3. The middleware service (Aggregator) queries the intended vault – e.g. the vault associated with the client or the owner of the middleware, or a satellite endpoint included in the request parameters to determine the boundaries of the project.
4. The queried vault determines the authorised union graph for the client and the middleware. If this includes the project access point, it yields the other endpoints in the project catalogue. All queryable endpoints are now known.
5. The middleware (Aggregator) can now execute the main query on the

partial projects of all stakeholders. Depending on the original request, it may be necessary to execute multiple requests.

6. The middleware (Aggregator) can combine the partial results into a single query result format.
7. (Optional) The results of the SPARQL queries serve as an input for the Adaptor part of the middleware service, which shapes them to the desired output format.
8. The middleware service responds to the original request.

These steps are visualised in Figure 6.1.

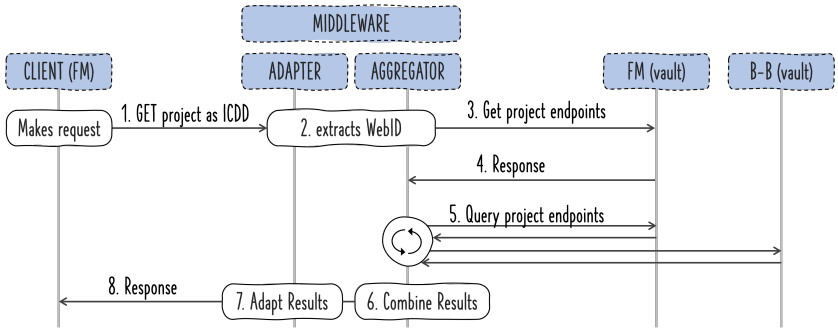

Figure 6.1: Flow of requests that allow a client to interact with a federated CDE through a single endpoint.

The satellites of a vault can be registered on the vault as well, namely as `dcat:Services`. As soon as the SPARQL satellite is known (see Listing 3.4), it can be queried to discover other satellites, for instance, the ConSolid satellites linked to project vaults (Section 6.3). By indicating to which standard the service conforms (`dct:conformsTo`), a client that expects a certain API or form can quickly check whether there is a satellite that offers this particular view. An example RDF description is given in Listing 6.1, indicating an interface that offers a view which is compliant to the BCF API, an industry specification for issue management in AECO projects.

When a particular satellite is required by a specific application, but it cannot be found at the side of the vault, a local instance of this satellite can be initiated client-side, facilitating this communication – provided that the satellite code is open sourced.

```
<> a dcat:DataService ;
    # The BCF API is a standardised API to communicate issues related to
     BIM models
    dct:conformsTo <https://github.com/BuildingSMART/BCF-API> ;
    dcat:endpointURL <https://example.org/bcf-satellite> ;
    dct:description "Service for mapping ConSolid projects to the BCF (BIM
      Collaboration Format) API" .
```

Listing 6.1: RDF description of a BCF API-compliant interface to the Pod, registered as a `dcat:Service`.

6.3 Functional satellites: the ConSolid API

The ConSolid API is a primary example of how functional satellites are essential for streamlining high-level interactions with data vaults. Since data vaults are essentially just catering for the storage of data, external validation is required to check whether the storage structure adheres to specific requirements, in this case the requirements imposed by the ConSolid ecosystem outlined in Chapters 3, 4 and 5, and the PBAC access control extension as outlined in Chapter 5. In many cases, a client will only be aware of the higher-level API offered by the satellite; the lower-level LDP interactions of a Solid server will only be used by the satellite to communicate with the data vault.

To force all interactions with a data vault to take place via the satellite (e.g. to avoid erroneous interactions), it is possible to only assign full editing rights to the WebID with which the satellite authenticates. Other actors get read access at most, even the (default) account of the vault owner. The satellite may then allow the owner of the vault to create new resources on the Pod, such as catalogues or Reference Collections, but only if the provided data complies to specific requirements. For example, the shape for OpenCDE containers as listed in Listing 5.3 (see Chapter 5).

A prototypical satellite is available on Github[12], created in NodeJS using the ExpressJS framework. In this prototype, the ConSolid satellite can be discovered by linking the property `consolid:hasConSolidSatellite` to the WebID of the vault owner. This is similar to the discovery pattern of the SPARQL satellite (Listing 3.4, Chapter 3).

[12] ConSolid API, https://github.com/ConSolidProject/cde-satellite. Accessed on 20/09/2023.

6.4 RDF Aggregators

The design and construction phases of an asset are phases with a lot of information production and exchange. At the same time, the project boundaries are quite well-defined and hosted on well-known data vaults. This makes it possible to create the equivalent of a SPARQL satellite for the entire project, synchronising with all RDF resources that can be found starting from a specific project access point, with given credentials. In the case of a central project access point, which was illustrated in Figure 3.3, this is quite straightforward. When a project access point has an RDF Aggregator attached, this is indicated by linking the `dcat:DataService` to the access point using `dcat:service` (Listing 6.2). The service maintains its own description. For example, the list of project resources (i.e. the total dataset aggregation) then dynamically changes as other partners change or update their project data. It is the responsibility of the service to stay synchronised and update its metadata (in this case the object of `dcat:servesDataset`) when necessary.

```
bb:d07af06a a dcat:Catalog, consolid:Project ;
    dcat:service bb:56f11919 ;

    dcat:dataset bb:1e19ed7c ,
        bb:32ad4402 ,
        otherPod1:aa3c09de ,
        otherPod2:bd503663 .

bb:56f11919 dcat:endpointURL <https://b-b.be/aggregator/d07af06a> .
```

Listing 6.2: Linking a Project Access point to an RDF Aggregator that mirrors its content.

Besides discovery of datasets, project-wide RDF Aggregators are specifically of use for a more performant retrieval of Reference Collections. The default discovery pattern for Reference Collections includes numerous requests back and forth between client, initial Reference Registry and the Reference Registries of the aggregated references (see Section 4.4.4). An RDF Aggregator can bundle the Reference Registries into one RDF store, making their alignment, and hence the retrieval of Reference Collections, much easier. Advanced RDF Aggregators may even apply reasoning to further speed up the process. For example, an `owl:sameAs` relation between the collection aliases (in parallel with `consolid:aggregates`) will effectively allow to treat the aliases as one dynamically aggregated concept.

A ConSolid RDF Aggregator as defined in this dissertation minimally has the following functionality:

- Query: query the project with SPARQL. This is the main functionality of the Aggregator.
- Get project access point: get the upper-level catalogue which resolves to metadata and RDF resources of the project. This is necessary to define the boundaries of the datasets to be included.
- Synchronisation: synchronise with the (remote) included datasets.

In the following paragraphs we will take a look at access control handling in RDF Aggregators, which differs from the procedure used for access control on individual data vaults. This is because some Aggregators will collect information from multiple vaults, without knowing the access rights attached to the original documents – which are mostly only accessible by the owner of this document. Therefore, an obvious requirement is that the owner of the RDF Aggregator must be trusted by the owners of the original dataset. This is partly done by granting them read access rights to these resources. However, the owner of the Aggregator must also be trusted to share resources according to the original access rights – to which they may not even have full access. This is both a legal challenge and a technical one. The legal part is considered out of scope for this dissertation; an approach for the technical one will be devised in the following paragraphs, more specifically related to granting access rights without having access to the original ACL document.

Logically, an RDF Aggregator can only expose resources to which it has read access itself. Different approaches will be devised for short-living RDF Aggregators that are only used by one agent (e.g. a client-side Aggregator) and persistent middleware services with RDF Aggregator functionality, to be used by different agents that are unknown to the service. The first scenario is straightforward: the set of aggregated resources coincides with the set of resources the agent is allowed to query. A variation of this scenario occurs when the Aggregator only mirrors in-office resources, i.e., the owner of the Aggregator is also the owner of all aggregated resources. In that case, the ACL resources can be mirrored along with the resources.

In the second scenario (the Aggregator can be queried by multiple agents), this will not be the case: the client will only be able to query the intersection of the resource set aggregated by the RDF Aggregator, and the resource set to which they have read access themselves. Because access to ACL resources is typically

limited to only the vault owner, the RDF Aggregator will not be able to go to the other project vaults and request a list of permitted resources for the client. Such list must be provided by the client themselves, and its integrity must be verified by the vault (or the ConSolid satellite on its behalf). To achieve this, ConSolid satellites must allow clients to request a list of resources to which they have read access (Listing 6.3). This list is to be signed digitally by the satellite, and may expire at a given point in time. For this example, the signature is created via asymmetric encryption [158], which also means that the public key must be available to verify the integrity of the signature. The subset of queryable resources can now be reconstructed by the RDF Aggregator, as the intersection of (1) the signed (and verified) lists provided by the client and (2) its own internal list of contained project resources.

```
{
  "allowed": [
    "https://b-b.be/data/b4b93478-2bc1-400e-acad-184a0ac208ad",
    "https://b-b.be/data/2880af2b-9d1a-4601-8559-5c08c3ac5962",
    "https://b-b.be/data/6d57703d-c4a7-4326-a657-8beec42e00e7",
    "https://b-b.be/data/ba994822-db41-4da3-8154-508ae68e3c4a",
    "https://b-b.be/data/aaff7fcc-3f9f-4840-9c4e-7666d71f54bc"
  ],
  "mode": "http://www.w3.org/ns/auth/acl#Read",
  "publicKey": "https://b-b.be/data/profile/publicKey.pem",
  "verifyUrl": "https://b-b.be/services/consolid/verify",
  "issuer": "https://b-b.be/data/profile/card#me",
  "actor": "https://b-b.be/data/profile/card#me"
}
```

Listing 6.3: A list of available resources (READ) on a specific data vault, for a specific agent. The list is digitally signed using asymmetric encryption, and can be send as a JSON web token (JWT).

A prototypical RDF Aggregator was developed for this dissertation, implemented as a read-only, in-memory SPARQL store based on the Comunica engine [166]. The satellite has a dedicated project access point, from where it oversees the aggregated RDF resources, both metadata and RDF project data. These resources can be loaded into the Aggregator via their LDP interface, and cached to allow a more performant response time for future requests. As the RDF Aggregator exposes a single, access-controlled endpoint for the entire project, compatibility with clients that are developed to work on a single SPARQL endpoint is possible. The prototype has been published on Github[13].

[13] ConSolid RDF Aggregator, https://github.com/ConSolidProject/RDF-aggregator/tree/dissertation. Accessed on 2023-10-30.

6.5 Mapping ConSolid Projects to industry standards

Although DCAT catalogues may serve as the main mechanism for container exchange between ConSolid-compatible agents, another compatibility layer is needed for those agents that are not directly built on top of ConSolid – in fact, virtually all existing applications and services apart from those described in this dissertation. This compatibility layer can be quite easily implemented when the original data is structured using open standards. The reusability of an Adaptor service will also increase when it exposes the data according to open standards as well. When this is the case, end-user applications such as BIM authoring tools will be able to communicate with the Adaptor, and hence with the federated project.

Whether a mapping to an industry standard succeeds, depends on multiple factors. Firstly, a higher degree of data structure will allow a more easy conversion to other expected data schema's and types. However, this is not sufficient, as obviously even with a high degree of structured data, information must first exist before it can be converted into a data structure that is compliant with a particular standard. For example, in Chapter 5, a shape was constructed linking mandatory metadata parameters for information containers conforming to the DIN SPEC 91391 OpenCDE specification (Listing 5.3). Other standards will require other fields to be present. Specific shapes can be set during project setup, or in a later phase. However, addition of dataset requirements after the creation of project datasets might result in non-valid datasets after a sanity check, requiring further enrichment of these datasets.

As the ConSolid satellite is to be used for the creation of ConSolid-compatible data on a vault, it can ask the user to provide all required (meta)data before proceeding with the creation of the dataset. However, as the ConSolid API is by design just one of multiple views on a vault, this can be bypassed by directly accessing the resource via LDP interface. Unless the ConSolid satellite is the only actor with editing rights to the data vault, regular sanity checks on existing project data are thus necessary as well, as elaborated in Chapter 5.

In previous chapters, the ISO 19650 stages of publication have served as an example for filtering datasets. In this chapter, two more extensive examples will be covered. The first example describes how to create a data dump of a federated ConSolid project in the form of an ICDD container. This example will not take into account the content of the datasets, but will be based purely on DCAT catalogues and datasets, the Reference Registry, and the definitions provided by the ICDD ontologies. In the second example, the content of the documents

will be taken into account by sketching how a service compatible with the BCF API may query the project data for issues documented using bcfOWL [154], an OWL ontology based on buildingSMART's BIM Collaboration Format (BCF). Depending on the original resource, one or more steps are needed to adapt the resource's content to the desired output. It is not the aim of this dissertation to provide a complete implementation of these specifications, but rather to conceptually outline how Aggregators and Adaptors can be combined with the data patterns described in earlier chapters.

6.6 Case Study: ISO 21597 - ICDD

Referring to Appendix E.2, an ICDD container has a determined structure, consisting of 3 folders (*Ontology resources*, *Payload documents* and *Payload triples*) and an index file describing its content and basic metadata for contained resources (e.g., format, creation date, label and original file name). The *Payload triples* folder contains the linksets (if any) which describe the sub-document links. For this example, it is assumed that an RDF Aggregator has been set up, either standalone or integrated as part of the middleware service.

```
CONSTRUCT {
  ?index a ct:ContainerDescription;
    ct:containsDocument ?dUrl ;
    ct:creationDate ?ctCreation ;
    ct:publishedBy <http://icddservice.org/> ; # the service
    ct:creator ?creator ;
    ct:description ?projectDescription ;
    ct:versionID "1" .
  ?dist a ct:ExternalDocument ;
    ct:creationDate ?creationDate ;
    ct:name ?label ;
    ct:description ?description ;
    ct:format ?format ;
    ct:filename ?filename .
} WHERE {
  BIND("2023-07-12T16:10:55.671+02:00"^^xml:dateTime as ?ctCreation)
  BIND(IRI(CONCAT("urn:uuid", STRUUID())) as ?index)
  BIND(replace(str(?mt), str("https://www.iana.org/assignments/media-types
     /"), str("")) as ?format)
  ?ds dct:creator ?creator ;
    dct:creationDate ?creationDate ;
    rdfs:comment ?description ;
    rdfs:label ?label ;
    dcat:distribution ?dist .
  ?dist dcat:mediaType ?mt ;
    dcat:accessURL ?dUrl .
}
```

Listing 6.4: SPARQL query (partial) to generate index.rdf for the ICDD container. Project ID and container creation date are known.

First, the *index.rdf* graph is to be constructed. This can be done by executing the query in Listing 6.4. To keep the SSoI intact, this query references all documents as being of type `ct:ExternalDocument`, referring to their location using `ct:url`. A similar query can be created if a project dump is needed that effectively contains all the documents, using respectively `ct:InternalDocument` and `ct:filename`. In that case, the middleware service will fetch the documents and store them in the *Payload documents* folder before sending the ICDD container as a ZIP archive to the client. To easily merge the results from the different endpoints, an identifier for the container and the index document is predefined and injected in the query, just like the creation time of the container. A shape can be created that allows creation of the ICDD index, similar to the shape that guarantees OpenCDE compatibility in Chapter 5.

```
CONSTRUCT {
    ?collection1 a ls:Linkset ;
        ls:hasLinkElement ?le .
    ?le ct:creationDate ?creationDate ;
        ls:hasDocument ?source ;
        ls:hasIdentifier ?selector .
    ?selector ls:identifier ?identifier .
}
WHERE {
  # subquery to filter all aliases as duplicates of the same collection
  {SELECT ?collection1
    WHERE {
      { # the collection has aliases, which should not be considered here as they would result in duplicate linksets
        ?collection1 consolid:aggregates+ ?collection2 .
        ?collection2 consolid:aggregates+ ?collection1 .
        FILTER(str(?collection1) < str(?collection2))
      } UNION {
        # the collection has no aliases
        ?collection1 consolid:aggregates/oa:hasSelector ?selector .
        FILTER NOT EXISTS {?collection1 consolid:aggregates/a consolid:ReferenceCollection}
}}}

  # propagate to find selectors, sources and identifiers for each collection
  ?collection1 a consolid:ReferenceCollection ;
    consolid:aggregates+ ?le . # aggregation of arbitrary depth
  ?le oa:hasSelector ?selector ;
    oa:hasSource ?source ;
    dct:created ?creationDate .
  ?selector rdf:value ?identifier .
}
```

Listing 6.5: SPARQL query (partial) to generate links.rdf for the ICDD container.

Next, the linksets are created. In this example, the Reference Aggregations are mapped to generic `ls:Link`s. A corresponding query is available in Listing 6.5. To avoid duplicates, only one of the aliases that aggregate each other is considered. Due to the property-chain and recursive structure of federated Reference Aggregations, the RDF Aggregator has a significant advantage compared to querying each Reference Registry individually (see Section 4.4.4).

6.7 Case Study: BCF API

A regular Issue Management Workflow contains of a chain of communications between different participants in a project. The open BIM format BCF is often used for this kind of processes, since stakeholders use different BIM software in many projects. Our example workflow will be based on the setup of the iGent project. In this workflow a request is submitted by the owner of the HVAC model, Arcadis, to inform the owner of the architectural model, B-B, that an opening needs to be created in a certain area of the building, so that technical equipment (for example, a pipe) can be routed through it – a procedure often denoted as 'Openings and Recesses' (Figure 6.2).

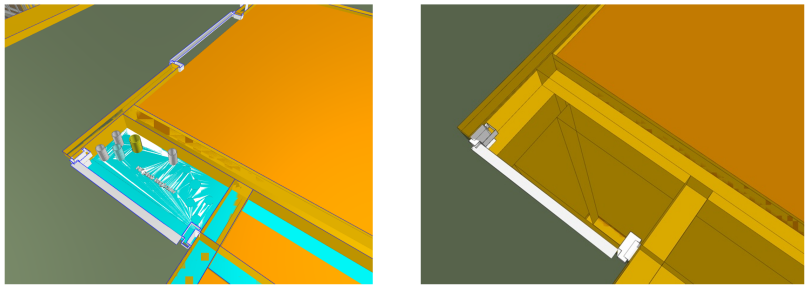

Figure 6.2: Left-hand side: geometric clash related to 'Openings and Recesses'. Right-hand side: geometry after the issue is resolved.

6.7.1 Workflow

In our scenario, we consider the use of a BCF API which allows communication with the ConSolid ecosystem. It is not the aim of the dissertation to provide a full BCF API implementation, but rather demonstrate the information flow from a federated ecosystem to the client via an existing BIM standard. The generic framework described in Section 6.2 will be used. Therefore, the first layer (the client) and the middleware layer, which combines the role of Aggregator and Adaptor, use the buildingSMART communication standard BCF API. The

client uses BIM authoring software with BCF communication capabilities. The identity associated with the client's vault is used for communication with the middleware and data layers.

In this example, the resources contain Issue Management data structured using the bcfOWL ontology [154], in line with the 4th and 5th star of structured data [13]. In the following paragraphs, the specific interaction patterns between the federated project and a micro-service that implements the BCF API specification are discussed, referring to the corresponding HTTP flows in Figure 6.3. Since the BCF middleware does not store any data, alternatively each stakeholder can initialise their own service without altering the results.

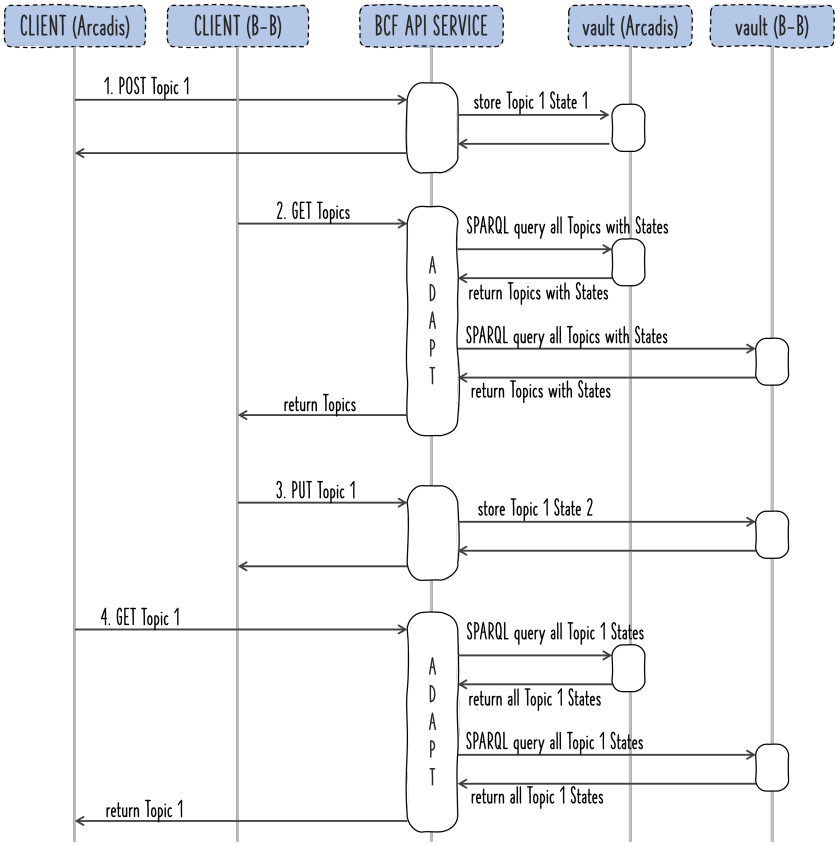

Figure 6.3: Sequence Diagram of the requests a client application (Application Layer) makes to a BCF API (Middleware Layer), which aggregates data from the project vaults (Storage Layer) and 'adapts' this data to match the requirements of the BCF specification. Based on [193].

Let us consider the scenario where Arcadis creates a new `Topic` for reviewing an opening in the building model (so that pipes can go through that wall), and assigns it to B-B. The newly created resource contains all the essential information for the approval request, such as the author, the creation date, the type, the corresponding building elements, and a camera perspective looking at the scenery, as well as a responsible partner. As discussed in Chapter 5, this can be validated with a shape, either defined in-project, or provided externally. For updating and describing the history of bcfOWL resources, the reader is referred to the approach of [125], which discusses the use of Topic States.

Listing 6.6 contains an example of how this `Topic` could be stored in the vault using RDF and bcfOWL, including an initial Topic State (Request #1). The BCF API interpretes the request, which is compatible with the BCF API specification, and takes care of the communication with the vaults. On the vault of Arcadis, a metadata record for this `Topic` resource is created to make it discoverable via the vault's SPARQL interface, and indicate whether the resource is structured using bcfOWL or regular BCF.

```
<https://arcadis.com/data/Topic1> a bcfowl:Topic ;
  dct:identifier "a5bee913-f912-494a-9cde-e5ebddd17226" .

<https://arcadis.com/data/Topic1State1> a bcfowl:TopicState ;
  bcfowl:hasTopic <https://arcadis.com/data/Topic1> ;
  bcfowl:hasTitle "Please check this opening" ;
  bcfowl:hasTopicType <https://dgfb.ugent.be/data/OpeningsAndRecesses> ;
  bcfowl:hasCreationDate "2022-12-12T13:45:30" ;
  bcfowl:hasCreationAuthor <https://arcadis.com/data/profile/card#me> ;
  bcfowl:hasTopicStatus <https://dgfb.ugent.be/data/ExtensionStatusOpen> ;
  bcfowl:hasLabel "Clash" ;
  bcfowl:hasAssignedTo <https://b-b.be/data/profile/card#me> .
```

Listing 6.6: The original bcfowl:Topic and its bcfowl:TopicState are stored on the vault of Arcadis.

At a later stage, B-B (i.e., the assigned party) checks via their BCF-compatible GUI whether there are any new Topics (Request #2), notices the new Topic, approves the issue assignment and updates the Topic (Request #3). Internally, a new Topic State is created on their vault. All updated properties are now added to this new `bcfowl:TopicState`, including a pointer back to the original `Topic` on the vault of Arcadis (i.e., the assigning party) (`bcfowl:hasTopic`). Listing 6.7 shows how this resource would now be stored in the vault of B-B. When the topic is queried (Request #4), the Adaptor merges the topic states on the different vaults into a single Topic that conforms to the BCF API specification. This process will be further documented in the next section.

```
<https://b-b.be/data/Topic1State2> a bcfowl:TopicState ;
  bcfowl:hasTopic <https://arcadis.com/data/Topic1> ;
  bcfowl:hasCreationDate "2022-12-13T12:14:15" ;
  bcfowl:hasCreationAuthor <https://b-b.be/data/profile/card#me> ;
  bcfowl:hasStatus <https://dgfb.ugent.be/data/ExtensionStatusClosed> ;
  bcfowl:hasAssignedTo <https://arcadis.com/data/profile/card#me> .
```

Listing 6.7: The updated bcfowl:TopicState is closed by B-B and reassigned to Arcadis. The new state is stored on the vault of B-B.

6.7.2 Adaptation of federated bcfOWL data to a BCF Topic

The BCF API specifies the following endpoint for retrieving a specific topic:

$$GET/bcf/\{version\}/projects/\{project\_id\}/topics/\{guid\}$$

This means that the project identifier must be known. In the case of ConSolid, this resolves to the identifier of the project access point (project_id), and the identifier of the Topic (guid). To construct its most recent view, the property `bcfowl:hasCreationDate` can be used. According to [28], the following properties are required:

1. guid
2. server_assigned_id: a project-unique identifier for the topic.
3. title
4. creation_date
5. creation_author

Additionally, the following optional parameters will be included:

1. assigned_to
2. modified_date
3. labels

Similar to the workflow discussed in Chapter 5 for creating DCAT datasets compatible with the OpenCDE containers, it is possible to include a shape in the project to validate whether all Topics have at least these required properties, provided a mapping to RDF properties exists.

To get a current view of the entire `Topic` via the BCF API, the middleware service (Adaptor and Aggregator) needs to query all `bcfowl:TopicStates` referring to the original `Topic`, and then reconstruct the topic based on the results. For example, the creation date, the labels and the author of the topic will be respectively the creation date, the labels and the author of the earliest topic state. The current status of the topic will be the status of the topic state which was created last.

```
SELECT ?topic ?cdate ?author ?topicId ?status ?assignedTo ?topicType ?label ?title WHERE {
    ?catalog a consolid:Project ;
        dct:identifier "46db9652-9d6b-4bd7-894b-462d3c807661" ; # the project ID is known
        dcat:dataset+ ?dataset .

    ?dataset dcat:distribution ?dist .
    ?dist dcat:accessURL ?topic ;
        dct:conformsTo <http://lbd.arch.rwth-aachen.de/bcfOWL#> .

    ?topic dct:hasIdentifier "a5bee913-f912-494a-9cde-e5ebddd17226". # the topic guid is known

    ?topicstate a bcfowl:TopicState ;
        bcfowl:hasTopic ?topic ;
        bcfowl:hasCreationDate ?cdate ;
        bcfowl:hasCreationAuthor ?author ;
        bcfowl:hasStatus ?status ;
        bcfowl:hasAssignedTo ?assignedTo ;
        bcfowl:hasTopicType ?topicType ;
        bcfowl:hasLabel ?label ;
        bcfowl:hasTitle ?title .
}
```

Listing 6.8: An exemplary query to retrieve all Topics of a project from the vaults.

The results of this query are included in Table 6.1 which shows the results from the different vaults after aggregation by the middleware service. Similarly, the property `bcfowl:hasCreationAuthor` can be used to find the provenance of data ownership.

?topic	?cdate	?author
<https://arcadis.com/data/Topic1>	"2022-12-12T13:45:30"	<https://arcadis.com/data/profile/card#me>
<https://arcadis.com/data/Topic1>	"2022-12-13T12:14:15"	<https://b-b.be/data/profile/card#me>

Table 6.1: Results of the query in Listing 6.8, based on the data in Listings 6.6 and 6.7. Rows in the table represent found Topic States.

6.7.3 Alternative approach using RDF Aggregators

Now, an alternative approach will be discussed, featuring an RDF Aggregator service as described in Section 6.4. Because this service will buffer datasets from the entire project, it can be queried as if it were centralised, i.e., using a single query. In addition, the usage of a CONSTRUCT query will be illustrated, resulting in an RDF graph of the topic (Listing 6.9). The latter step is only illustrative and does not depend on the use of an RDF Aggregator.

```
CONSTRUCT {
  ?topic dct:identifier ?topicId ;
    bcfowl:hasCreationAuthor ?author ;
    bcfowl:hasCreationDate ?cdate1 ;
    bcfowl:hasStatus ?status ;
    bcfowl:hasAssignedTo ?assignedTo ;
    bcfowl:hasTopicType ?topicType ;
    bcfowl:hasLabel ?label ;
    bcfowl:hasTitle ?title .
} WHERE {

  BIND("a5bee913-f912-494a-9cde-e5ebddd17226" as ?topicId)

  ?topic dct:identifier ?topicId .
  ?initialTopicState a bcfowl:TopicState ;
    bcfowl:hasTopic ?topic ;
    bcfowl:hasCreationDate ?cdate1 ;
    bcfowl:hasCreationAuthor ?author ;
    bcfowl:hasAssignedTo ?assignedTo ;
    bcfowl:hasTopicType ?topicType ;
    bcfowl:hasLabel ?label ;
    bcfowl:hasTitle ?title .

  # there are no older topic states, so this is the initial state
  FILTER NOT EXISTS {
    ?otherState1 a bcfowl:TopicState ;
      ex:hasCreationDate ?otherCDate1 .
    FILTER (?otherCDate1 < ?cDate)
  }

?finalTopicState a bcfowl:TopicState ;
    bcfowl:hasTopic ?topic ;
    bcfowl:hasCreationDate ?cdate2 ;
    bcfowl:hasStatus ?status .

# there are no more recent topic states, so this is the current state
FILTER NOT EXISTS {
    ?otherState2 a bcfowl:TopicState ;
        bcfowl:hasCreationDate ?otherCDate2 .
    FILTER (?otherCDate2 > ?cDate2)
}}
```

Listing 6.9: An exemplary query to retrieve all `Topics` of a project from the vaults.

As the BCF API relies on JSON (JavaScript Object Notation) for serving its information, this graph can be converted to the JSON-based RDF format JSON-LD – which attaches a context to a JSON object. If required, further alignment can rely on JSON-LD framing and the JSON-LD framing API[14]. As the BCF API does not include anything like Topic States, the CONSTRUCT query directly maps the retrieved information to the topic itself. Using filters, the right creation date, author and current status can be derived. Since the RDF Aggregator is project-bound, the topics are automatically part of the project - the dataset discovery pattern indicated in the query from Listing 6.8 can thus be omitted.

Applying the context indicated in Listing 6.10, the JSON response in Listing 6.11 will be obtained. Using JSON-LD compaction [164], the BCF API can send a response to the client containing the object related to the topic. The URI of the topic as retrieved from the original vault is used as a value of 'server_assigned_id'.

```
{"@context": {
    "bcfowl": "http://lbd.arch.rwth-aachen.de/bcfOWL#",
    "dcterms": "http://purl.org/dc/terms/",
    "guid": "dct:identifier",
    "title": "bcfowl:hasTitle",
    "labels": "bcfowl:hasLabel",
    "creation_date": "bcfowl:hasCreationDate",
    "topic_status": "bcfowl:hasTopicStatus",
    "server_assigned_id": "@id",
    "assigned_to": {
        "@id": "bcfowl:hasAssignedTo",
        "@type": "@id"
    },
    "creation_author": {
      "@id": "bcfowl:hasCreationAuthor",
      "@type": "@id"
    },
    "topic_type": {
      "@id": "bcfowl:hasTopicType",
      "@type": "@id"
}}}
```

Listing 6.10: Context to make the results compatible with BCF API response without losing semantics.

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

```
{
    {...context},
    "guid": "a5bee913-f912-494a-9cde-e5ebddd17226",
    "server_assigned_id": "https://arcadis.com/data/a5bee913-f912-494a-9
     cde-e5ebddd17226",
    "creation_author": "https://arcadis.com/data/profile/card#me",
    "creation_date": ""2022-12-13T12:14:15"",
    "topic_type": "https://dgfb.ugent.be/data/OpeningsAndRecesses",
    "topic_status": "https://dgfb.ugent.be/data/ExtensionStatusClosed",
    "title": "Please check this opening",
    "labels": [
        "Architecture",
        "Openings and Recesses"
    ]
}
```

Listing 6.11: JSON response generated by flattening the JSON-LD graph constructed by flattening the query in Listing 6.9 with the context provided in Listing 6.10

6.8 Conclusion

This Chapter introduced higher-level services which mediate between the raw data structures present on the vaults, and the end user applications, which mostly expect the data in a specific format (Adaptors), coming from a specific endpoint (Aggregators). Furthermore, the fact that a data vault does not have real functionality beyond storage and access control demands that higher-level interactions that require a particular workflow are executed through a dedicated service (or chain of services). When this service has some delegated access rights, it is recommended to not use one service for interacting with multiple vaults. Because of their close relationship to a data vault, in this dissertation these services are called *functional satellites*.

Furthermore, it was recognised that some scenarios benefit from a central access point for reading data, in contrast to sending queries to each of the participating vaults and then combining them locally. The RDF Aggregator that was introduced in this chapter uses an in-memory triple store to synchronise with the project's RDF resources. This includes at least the metadata resources (catalogues and datasets), but project resources that follow RDF syntax may be aggregated as well. In Chapter 4, two methods were conceptually described for retrieving Reference Collections, namely using federated queries or a cached union of the project's Reference Registries. The performance of the second scenario was significantly higher. A relevant application for the RDF Aggregator is thus to provide such union of Reference Registries, as an aid in discovering and aligning federated Reference Collections and their aliases.

Finally, it was shown how a subset of a ConSolid project can be the main input for other views on the project. When these views are based on international standards (e.g. ICDD or BCF), this means that the ecosystem can be made compatible with existing BIM workflows, without impacting the workflows themselves. In the example of ICDD, a container archive of the federated project was created, as discovered from a single project access point. The example of featuring the BCF API showed how not only ConSolid's metadata structures can be adapted to match existing industry practice, but also how highly structured *project data* can be stored in a federated way and nevertheless allow a central application to interact with project data – without any knowledge of the federated nature of the project.

Table 6.2 gives a summary of the characteristics of a federated CDE (as defined in this dissertation), focusing on this chapter's topic of aggregation and adaptation. The characteristics are related to the technologies that were identified as suitable for supporting them.

Characteristic	Technology	Data Patterns
C8 - Aggregators	HTTP(S), Web servers	n.a.
C9 - Adaptors	API mappings, Web servers	n.a.

Table 6.2: Characteristics for a federated CDE (aggregation and adaptation), corresponding technologies and data patterns as implemented in the ConSolid ecosystem.

6.9 Related Publications

This chapter contains edited fragments or concepts derived from the following publications:

- Jeroen Werbrouck, Pieter Pauwels, Jakob Beetz, and Erik Mannens. "Mapping Federated AEC projects to Industry Standards using dynamic Views". In: *10th Linked Data in Architecture and Construction Workshop*. CEUR-WS. org. 2022. URL: https://ceur-ws.org/Vol-3213/paper06.pdf (accessed 2024-3-18).
- Jeroen Werbrouck, Pieter Pauwels, Jakob Beetz, and Erik Mannens. "ConSolid: a Federated Ecosystem for Heterogeneous Multi-Stakeholder Projects". In: *Semantic Web Journal* (2023). Accepted. URL: https://biblio.ugent.be/publication/8633673/file/8633674.pdf (accessed 2024-3-18).

- Jeroen Werbrouck, Oliver Schulz, Jyrki Oraskari, Erik Mannens, Pieter Pauwels, and Jakob Beetz. “A generic framework for federated CDEs applied to Issue Management”. In: *Advanced Engineering Informatics* 58 (2023), p. 102136. URL: https://doi.org/10.1016/j.aei.2023.102136 (accessed 2024-3-18).

Chapter 7

Interfaces for Linking Federated Multi-Models

Some platforms already exist to interact with multi-models of limited scope, i.e., multi-models which will only contain specific media types during their lifetime (e.g., IFC, imagery, RDF) [201, 97]. However, the heterogeneity of federated multi-models as described in the previous chapters (i.e. with potentially *unlimited* scope) requires a more generic infrastructure. In this chapter, a modular framework is developed that allows the end user to link data in such federated multi-models, (largely) independent from their media types and in a user-friendly way. This framework will be called 'Mifesto' (Micro-Frontend Store). Conceptually, Mifesto functions independently from the data ecosystem described in previous chapters; it is not a part of the ConSolid ecosystem. Instead it forms a separate yet compatible ecosystem, with its own reasons to consider a federated setup, and its own reasons to consider catalogue-based data patterns. This means it will be applicable to multi-models in general, federated (ConSolid) and centralised (ICDD) alike. While the role of desktop BIM applications and, more recently, Web applications, will be to *create* project information, the role of Mifesto lies in *linking* such information with other sources on the Web via a GUI. Hence, it can co-exist in harmony with other approaches for creating project information. The framework will be based on domain-agnostic considerations, so its core patterns are not related to the topic of the built environment. This is done for the same reasons that steered the generic setup of ConSolid, namely to allow maximal extensibility to connect to other domains.

First, the characteristics of such framework will be defined (Section 7.1), similar to the earlier chapters of this dissertation. Then, the components for a federated 'application store' for micro-frontend modules will be introduced (Section 7.2). The RDF definitions that allow a semantic description of these components are described in Section 7.3, showing how these definitions can be combined to create the eventual ecosystem of federated store, module manifests and configurations. Section 7.4 discusses the eventual structure of the *AECOstore*, as an implementation of Mifesto; a store of applications related to the built environment. Finally, a reproduceable proof-of-concept is described

in Section 7.5. In this proof-of-concept, the user may select 'damage enrichment' from a set of potential interactions that are supported by modules in the store. Based on the already available resources in the active multi-model, a module can then be selected to provide a proxy for this semantic enrichment (e.g. geometric model).

Background technologies for this chapter are introduced in the following appendices:

- Appendix B (Semantic Web technologies (RDF, SPARQL, SHACL)) ;
- Appendix E (The DCAT vocabulary) ;
- Appendix G (Selectors for sub-document identifiers) ;
- Appendix J (the Mifesto Vocabulary).

7.1 Characteristics

From the perspective of linking heterogeneous datasets, the combinations of ontologies and file formats are virtually endless. Moreover, the order in which information is added to the multi-model is ideally defined based on the individual needs, tasks and already present datasets within a project, not by an external application developer. For example, if the activity involves the documentation of damage records, one intuitive way would be to do so by selecting a 3D element in a viewer and then inserting the details of the damage, possibly accompanied by a picture. However, acting along the lines of multi-models, which do not impose a hierarchy amongst their constituent resources, the availability of a 3D model should not be a prerequisite for a damage documentation activity. After all, the documentation process itself does not even need *any* proxy: a single spreadsheet, for example, would suffice in the first stage. To become part of a larger multi-model, however, the content of this dataset should then be linked to other project resources such as pictures or 3D geometry in a later stage – depending on the available project information at that later stage. The core goals of such framework were summarised in Section 2.4:

> "*[...] it should thus be possible to initiate and enrich a maximally structured multi-model from whatever domain is required by the current task (topology, damage, product information, user data ...), based on whichever auxiliary resource (3D geometry, CAD plans, cityGML, point clouds, imagery, textual documents, sensor data, RDF*

> *data ...) is already available or needs to be created or linked. Consequentially, an indefinite amount of semantic enrichment interfaces may exist. [...] With this in mind, an interface-oriented framework is needed that is as versatile as the multi-models it ought to interact with.*"

This yields a challenge for interface development and management: how to provide an industry specialist, mostly with little background in data models and schemas, with the means to interact with such heterogeneous multi-model? The concept of the 'citizen developer' [120, 106] and the upcoming of low-code [142] environments bring inspiration here. Low-code platforms lower the threshold for the configuration and use of organisation-specific IT solutions, which can now be easier configured by people with less coding experience, e.g., domain specialists. Often, they base upon visual programming interfaces which hide the internal complexity of modules that are chained together to achieve a particular outcome.

In this dissertation, the following (technology-agnostic) characteristics are identified for a flexible and future-proof ecosystem of federated frontends:

- **C 10:** ***Modularity****: The frontend ecosystem consists of independent modules rather than being monolithic.*
- **C 11:** ***Indirect Enrichment****: Concept enrichment and linking happens in an indirect way, to allow full functional independence of modules.*
- **C 12:** ***Common I/O Interface****: A minimal common I/O interface exists between the modules to communicate during runtime. The I/O patterns should be independent from the purpose of a module.*
- **C 13:** ***Decentral Publishing****: The ecosystem allows for the modules to be published in a decentral way to allow (1) third parties to publish targeted modules and (2) allow for on-the-fly module aggregations addressing new interaction scenarios.*
- **C 14:** ***Discoverability****: The decentrally published modules are discoverable on the Web. A rich, machine-readable metadata description of a module is necessary to allow identification and aggregation of the intended modules. The same holds for machine-readable descriptions of a multi-module configuration, which results in a GUI.*

A semantic enrichment paradigm based on plugins for a monolithic authoring tool is insufficient, as it has too great a dependency on one core model (IFC, RVT, ...), which might not be available at the time data needs to be added to the multi-model. It is therefore necessary that an enrichment interface is composed of *equivalent interaction modules* (C10). This multifaceted nature aligns with the vision of a 'general' CAD system as described by Ross and Rodriguez in 1964 [147], namely that it is necessary [...]:

> *"[...] to recognize once and for all that it is completely impossible to construct a system which will satisfy the requirements immediately and without modification. In fact to postulate the existence of a closed system for Computer-Aided Design as we mean it is completely and absolutely contradictory to the very sense of the concept. [...] The very nature of the system must be such that its area of application is continuously extended by its users to provide new capabilities as the need arises. [...] Thus we see that the basic thing to be provided in the initial organisation and structure of the [CAD] system is a capability for growth, expansion and modification. [...] If, in fact, the system can be so organised that it can naturally be moulded to suit the needs and interests of individual users, then the concept of a general Computer-Aided-Design System not only begins to seem possible, but practicable as well."*

This vision for CAD systems holds even more for an environment where semantics exist besides geometry (BIM, GIS ...). In an environment for multi-models, other resources such as imagery, point clouds, spreadsheets and external APIs also come into play. Practically, this means that one module (e.g. damage enrichment) cannot expect that another module (e.g. geometric viewer) will be part of the configuration as well. To allow these modules to nevertheless communicate with one another, a higher-level, 'abstract' concept is needed, of which the sole purpose is to aggregate all heterogeneous references to the same 'thing'. These references can be anything, for example, a geometric representation, an HTTP URI for RDF representations, a mentioning in a text or a pixel zone in an image. The overlaps with ConSolid's Reference Collections or the ICDD linksets, which were both introduced in Chapter 4, are immediately clear. From the perspective of CAD and BIM authoring tools, parallels with the 'connectors' in the Speckle ecosystem [1] can also be identified.

From a UI perspective, a Reference Collection or Linkset can thus be 'selected' via any of its representations, providing a proxy to all other representations as well as their metadata and original source. Modules in the ecosystem will be specialised in one or more types of selection, e.g., geometric selection or by query. As they only need a connection to a linkset, they can be unaware of all other modules in the configuration, and nevertheless enrich a concept that has been selected through a representation in another module. This principle of *indirect enrichment*, which allows a module to align with this abstract concept instead of its representations, identified by C11 is illustrated in Figure 7.1. To allow a module to broadcast and receive any linkset of interest and check if it is able to interact with one of its representations, a common I/O interface is necessary (C12).

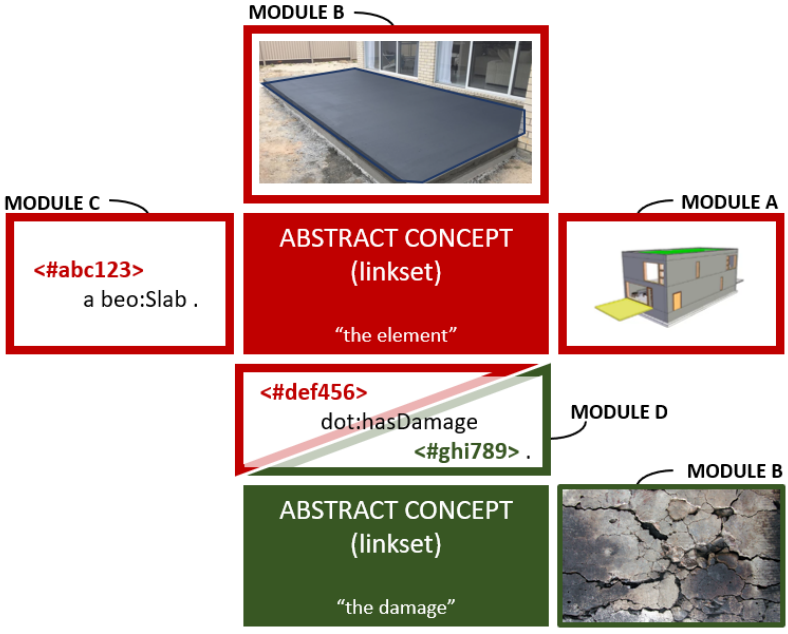

Figure 7.1: Digital representations of the same concept are connected, and can be related to representations of other concepts via compatible UI modules.

As there is no single instance governing what media types and ontologies may exist, let alone the interaction scenarios made possible by their combination, the development and publication of modules should also take place in a decentral way (C13). This decentral development practice also makes sense

bearing the notorious AAA-slogan of the Web (Anyone can say Anything about Anything) in mind: if anyone thinks their module is worth publishing (e.g. because they create a new ontology and they want it to be useful for semantic enrichment), they should be able to. Whether the module is actually *useful* will be determined by the end users. This does, however, not necessarily mean that perfidious applications can wreak havoc and go unchecked: modules can be aggregated in catalogues, and trustworthy catalogue providers may vouch for the reliability of the modules they aggregate. Catalogue providers may be, amongst others, academic institutions, CDE vendors or companies publishing BIM authoring tools. As catalogues are mere pointers to URLs on the Web, they can also be aggregated recursively in other catalogues. Likewise, earlier created multi-module configurations can be published decentrally and describe frequently occurring cases. This accounts for the discoverability mentioned in C14. The characteristics for queryability and discoverability are only fulfilled if a machine-readable description of modules and configurations is available, alongside loadable code.

The Mifesto ecosystem combines the characteristics for decentral publishing (C13 and C14) of independent modules (R10) that can indirectly communicate with one another by communicating abstract selection sets (11) through a common I/O interface (R12). The result is a decentral 'application store', which can be used to find existing GUI configurations, fit for a particular topic, or to create new ones from published modules. In such ecosystem, independently developed modules may be reused across multiple scenarios, dynamically combined in a tailor-made GUI that is only based on the available project data (e.g. images) and the current task (e.g. damage documentation). Consequently, the development effort to construct a GUI for linking multi-model data on the Web is much lower than with 'monolithic' SPAs.

Let us consider the amount of combinations that can be made from the following five modules:

1. 3D Viewer
2. Image Annotator
3. Point Cloud Annotator
4. Damage Data Enricher
5. Acoustic Data Enricher

Sometimes, the combination of two similar modules will be relevant, e.g. when two images or two geometric models need to be linked. On other occasions, this makes less sense, especially for semantic enrichment modules. However, generally, we can see that there will always be a lesser amount of modules than monolith interfaces for the same amount of scenarios. In the situation illustrated in Figure 7.2, there are only five reusable modules that need to be created to cover twelve relevant combinations - a monolithic environment would require twelve interfaces.

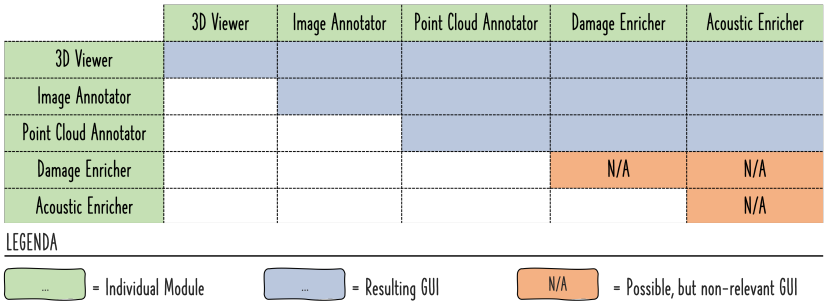

Figure 7.2: When an interface intends to address a linking scenario between various media types and/or ontologies, the development effort will be less in a modular environment than when developing 'monolithic' applications.

7.2 Components

The frontend ecosystem will include the following high-level components in order to comply to the characteristics discussed in Section 7.1. This includes both data-based components and service-based components. This section will discuss these components in a technology-agnostic way.

1. Module manifests (data component)
2. Loadable module code (data component)
3. Interface configurations (data component)
4. Federated (nested) catalogues (The 'store') (data component)
5. Store querying service (service component)
6. Bundler application (service component)

Module Manifests are the main access points for retrieving information about individual modules. A Manifest yields as much information as needed to be aggregated in larger configurations and be (made) compatible with other modules in a configuration and the data structures in a specific project. In Section 7.3, semantic definitions needed to create such Manifest will be discussed.

A Manifest describes the intention and functionalities of a module, but is separated from the actual loadable code. This code must be dereferenceable on the Web via a URL. As the focus in this context is on the application code as a specific resource, this component is labelled a 'data component'. However, after loading and parsing this resource in the Bundler application (see further), it can be seen as a service component as well.

Similar to individual module manifests, *Interface Configurations* are semantic descriptions of the combination of existing modules, which help in addressing a particular interaction scenario. If, so to speak, the Module Manifests would be the ingredients for creating a GUI, the Interface Configuration would be the recipe – providing instructions on how the bundler must organise the modules in pages, routes, components and widgets. They can be created dynamically, before loading them in the Bundler, or stored as already addressed data applications, allowing discovery and reuse of existing configurations.

Interface Configurations and Modules can be contained in larger catalogues by their URL. These catalogues can, in turn, be nested in multiple other catalogues, which allows the setup of a graph-like, decentral *Micro-Frontend Store*. Independent developers can create and publish modules and configurations without being aware of other branches in the tree, allowing new modules to be embedded in larger configurations just by creating an over-arching store catalogue and setting this catalogue as the main store in the UI. Using link-traversal technologies (see Section 3.2.4), the branches of this tree can be discovered on-the-fly by a store querying service (Figure 7.3).

The *Bundler* is a container application, a broker responsible for wiring the UIs dictated by a loaded Interface Configuration, checking the compatibility of modules and facilitating information exchange. Based on the URL of an Interface Configuration, the Bundler discovers which modules should be loaded, how they should be displayed, whether there are any nested modules and what global parameters they rely on. These definitions are structured using a dedicated ontology, which is discussed in Section 7.3. The Bundler ensures the retrieval of Reference Collections and broadcasting their selection to the loaded modules.

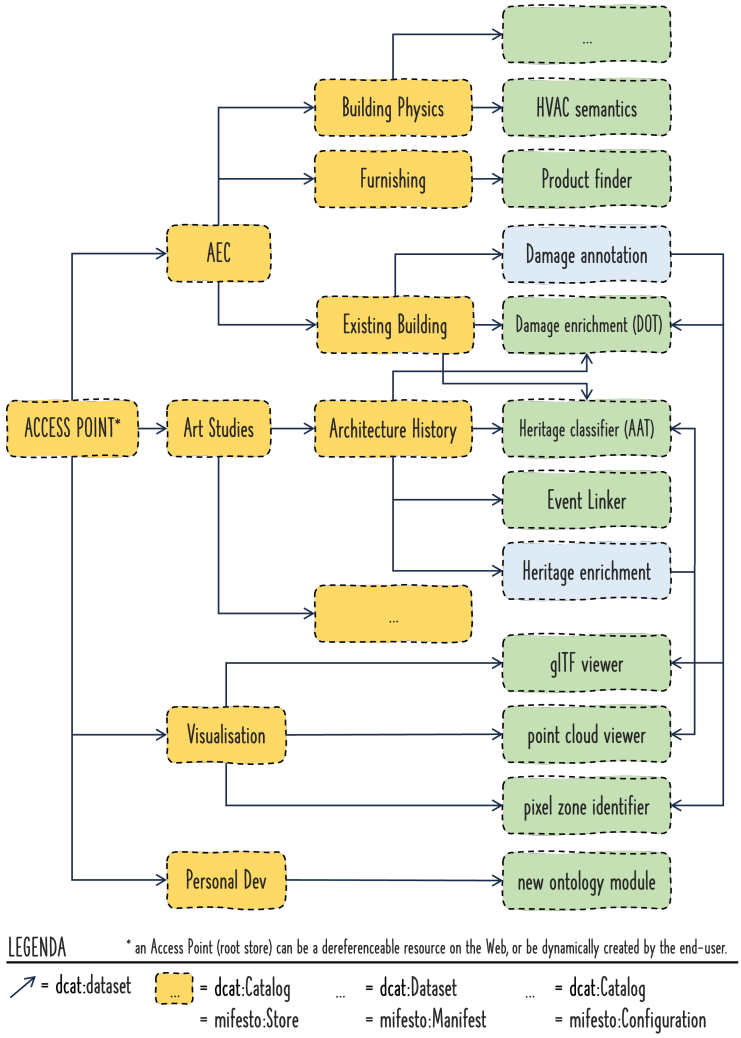

Figure 7.3: A property chain of federated stores (DCAT catalogues) starts from a main access point for the store and allows aggregation of all lower-level modules in the federated store.

7.3 The Mifesto vocabulary

The following Sections introduces the Micro-Frontend Store (Mifesto) vocabulary, which provides an ontological basis for federated frontend modules to be combined in larger configurations, allowing users to communicate with heterogeneous, federated data in a user-friendly way (compared to API calls).

7.3.1 Manifest Definitions

A manifest is a self-describing Web resource, containing the necessary information for the module to be loaded and made interoperable with other modules, as well as metadata required to discover the module. The following core classes and properties are identified:

- Class indicating that a resource is a manifest for declaring the module's properties (`mifesto:Manifest`) (subclass of `dcat:Dataset`).
- Object Property referencing the used distribution of the module (`dcat:distribution`).
- Datatype Property referencing the revision number of the module (`doap:revision`).
- Object Property referencing the URL where the injectable code of the micro-frontend is published (`mifesto:code`) (subclass of `dcat:accessURL`).
- Object Properties for indicating the compatibility of the module with a particular type of sub-document identifiers; both regarding what the module is able to interpret (`mifesto:readsIdentifier`) and what it is able to register in the multi-model (`mifesto:writesIdentifier`). This may be indicated with conformance (`dct:conformsTo`) to a particular standard, SHACL shape, GUID validator or other ways to programmatically validate an entry.
- Object Property indicating which mediatypes the module can interact with, such as geometry formats (e.g., glTF, COLLADA), BIM models (e.g., IFC, RVT...) imagery (e.g., JPEG, PNG), point clouds (e.g., E57, XYZ) or any RDF serialisation. When available, the mediatypes listed by iana.org [76] should be used (`mifesto:compatibleMedia`).
- Object property indicating which ontologies the module is compatible with, both in terms of interpreting and semantic enrichment of the project (`mifesto:usesVocabulary`). For example, a module for setting the topological structure of a project may list the BOT ontology; a damage enrichment module could reference the DOT ontology.

```
# setting the document as a mifesto:Manifest
<> a dcat:Dataset, mifesto:Manifest;
    dcat:distribution <#v1> ;
    rdfs:label "Geometry visualisation module" ;
    rdfs:comment "This module allows to visualise glTF models present in
     the federated multi-model" .

<#v1> a dcat:Distribution ;
    doap:revision "v1" ;
    mifesto:code <https://aecostore.github.io/viewer-module/index.js> ;
    mifesto:compatibleMedia <https://www.iana.org/assignments/media-types/
     model/gltf+json> ;
    mifesto:readsIdentifier <#identifierDefinition1> ;
    # conformance with ConSolid projects
    dct:conformsTo <https://consolidproject.be/shapes/module> .

<#identifierDefinition1> a mifesto:IdentifierDefinition ;
    dct:conformsTo <https://saref.etsi.org/saref4inma/UUID> .
```

Listing 7.1: An example of a Module Manifest, describing a module for geometric visualisation.

```
# setting the document as a mifesto:Manifest
<> a mifesto:Manifest;
    dcat:distribution <#v1> ;
    rdfs:label "Damage annotation module" ;
    rdfs:comment "This module allows to interact with damage records in a
     ConSolid project, including retrieval of damaged elements and
     creation of damage records." .

<#v1> a dcat:Distribution ;
    doap:revision "v1" ;
    mifesto:code <https://aecostore.github.io/damage-module/index.js> ;

    # the module allows uploading and viewing imagery
    mifesto:compatibleMedia <https://www.iana.org/assignments/media-types/
     image/png>,
         <https://www.iana.org/assignments/media-types/image/jpeg> ;
    mifesto:readsIdentifier <#identifierDefinition1> ;
    mifesto:writesIdentifier <#identifierDefinition1> ;

    # indicates conformance with the ConSolid project definition of
     Reference Registries, sub-document identifiers and Dataset
     Collections
    dct:conformsTo <https://consolidproject.be/shapes/module> ;

    # indicates usage of the Damage Topology Ontology (DOT)
    mifesto:usesVocabulary <https://w3id.org/dot#> .
```

Listing 7.2: An example of a Module Manifest, describing a module for damage annotation.

7.3.2 Interface Configurations

An Interface Configuration contains the instructions for a Bundler application to generate a UI. It references the manifests of the modules that should be loaded and describes how modules are organised (standalone or nested in pages or other modules) and their dimensions. When possible, definitions from frequently occurring ontologies are reused. The following core classes and properties for Interface Configurations are identified:

- Class indicating that a resource is an Interface Configuration (`mifesto:Configuration`) (subclass of `dcat:Catalog`).
- Class indicating that a resource is a separate page in the interface (`mifesto:Page`) (subclass of `dcat:Catalog`).
- Class indicating that a resource is a component in the interface (`mifesto:Component`) (subclass of `dcat:Catalog`).
- Subclass of `mifesto:Component` indicating that a resource is an organisational component, i.e it does not have any content of its own, but requires child components (`mifesto:OrganisationalComponent`).
- Object Property referencing the Module Manifest to be loaded by the Component (`mifesto:hasModule`) (subclass of `dcat:dataset`).
- Object Property referencing the child modules of an organisational component (`mifesto:hosts`) (subclass of `dcat:dataset`).
- Datatype Property indicating the route by which the Page will be accessible (`mifesto:hasRoute`).

The following classess and properties can be seen as auxiliary, and are mainly oriented towards generating the lay-out of the application:

- Class indicating that a resource is a dimension setting in the interface (`mifesto:DimensionSetting`).
- Object Property referencing the Dimension Setting for a Component or Page (`mifesto:hasDimensionSetting`).
- Subclass of `mifesto:DimensionSetting` indicating that a resource is a grid dimension setting, i.e. there must be a row-column layout for this Component or Page (`mifesto:GridDimensionSetting`).

- Object Property referencing the actual dimensions for a Component or Page (`mifesto:hasDimensions`).
- Datatype Properties referencing the actual dimensions for a Component or Page (`mifesto:initialRows` and `mifesto:initialColumns`).

Listing 7.3 shows an example of how these properties are used in an Interface Configuration.

```
# setting the document as a mifesto:Configuration
<> a mifesto:Configuration, dcat:Catalog ;
    mifesto:hosts <#p1>, <#p2> .

# The project interaction page, hosting the interaction modules
<#p1> a mifesto:Page, mifesto:OrganisationalComponent, dcat:Catalog ;
    rdfs:comment "This is the project interaction page" ;
    mifesto:hasDimensionSetting <#dimset1> ;
    mifesto:hasModule <https://raw.githubusercontent.com/AECOstore/demo-page/main/public/manifest.ttl> ;     # the hosting (UI-less) module
    mifesto:hosts <#m1>, <#m2>, <#m3> ; # the hosted modules
    mifesto:hasRoute "/demo" . # the route associated with this page

# the authorisation page
<#p2> a mifesto:Page, mifesto:Component, dcat:Catalog ;
    rdfs:comment "This is a Solid-compatible authentication page" ;
    mifesto:hasRoute "/auth" ;
    mifesto:hasModule <https://raw.githubusercontent.com/AECOstore/auth-page/main/public/manifest.ttl> .

# a hosting (UI-less) module for tab-based hosting of modules
<#m1> a mifesto:Component, mifesto:OrganisationalComponent, dcat:Catalog ;
    mifesto:hasModule <https://raw.githubusercontent.com/AECOstore/tabs-module/main/public/manifest.ttl> ;
    mifesto:hasDimensions
    [ mifesto:initialColumns 4 ; mifesto:initialRows 8 . ];
    mifesto:hosts <#m1_1>, <#m1_2> .

# module related to damage assessment.
<#m1_1> a mifesto:Component, dcat:Catalog ;
    mifesto:hasModule <https://raw.githubusercontent.com/AECOstore/damage-module/main/public/manifest.ttl> .

# module for querying the project (e.g. SPARQL, GraphQL)
<#m1_2> a mifesto:Component, dcat:Catalog ;
    mifesto:hasModule <https://raw.githubusercontent.com/AECOstore/query-module/main/public/manifest.ttl> .

# module for displaying 3D geometry and visual selection aid
<#m2> a mifesto:Component, dcat:Catalog ;
    mifesto:hasModule <https://raw.githubusercontent.com/AECOstore/viewer-module/main/public/manifest.ttl> ;
    mifesto:hasDimensions
       [ mifesto:initialColumns 8 ; mifesto:initialRows 8 .] .

<#dimset1> a mifesto:GridDimensionSetting . # the dimension settings
```

Listing 7.3: An example of an Interface Configuration.

7.4 Mifesto Stores

The DCAT vocabulary is considered a fit foundation for defining stores and referencing their content. The Mifesto Store class (`mifesto:Store`) is then a subclass of `dcat:Catalog`. Stores can reference either Modules, Interface Configurations or other `dcat:Catalogs`. As `dcat:Catalogs` can be chained (a `dcat:Catalog` is a subclass of `dcat:Dataset` and references its constituent sub-datasets with `dcat:dataset`), it is possible to create a union of two or more existing resources to form a single point of access and create a federated catalogue of configurations and modules, to address specific interaction scenarios (see Figure 7.3). These configurations and modules may be created independently by different organisations. A generic query for all modules and configurations in a store is given in Listing 7.4. More refined queries may be necessary to discover the modules needed for a specific interaction with the multi-model. A `dcat:Catalog` that is also a Mifesto Store will be classified as `mifesto:Store`. In analogy with *Dataset Collections* and *Reference Collections* in earlier chapters, another name for a `mifesto:Store` could be *Interface Collection*.

```
SELECT * WHERE {
    # a property chain of dcat:dataset links
    ?store dcat:dataset+ ?module ;
        a dcat:Catalog, mifesto:Store .

    ?module a mifesto:Manifest ;
        dcat:distribution ?dist .
    ?dist mifesto:code ?code .

    # optional further filters on ?module or ?dist
    # ?module mifesto:usesVocabulary <https://w3id.org/dot#> .
}
```

Listing 7.4: SPARQL query to identify all modules in a federated application store (DCAT). A query engine supporting link traversal will be necessary.

A store can be dynamically queried using LTQP (see Section 3.2.4), or maintain a local triple store which synchronises with the individual configurations and manifests. As the data patterns of the store are based on the DCAT vocabulary, the RDF Aggregator described in Chapter 6 can be used to aggregate the federated modules of a Mifesto store, similar to the aggregation of ConSolid projects: the URL of a catalogue is used as an access point to discover its sub-catalogues and their datasets to an arbitrary depth and load them into a local triple store.

7.5 Proof-of-Concept

This dissertation provides a prototype of the Bundler application and some modules based on the Piral [160] framework. The default Piral infrastructure is adapted to aggregate and interpret federated micro-frontends described with the Mifesto vocabulary, and facilitate the interactions between the modules, based on Solid authentication, ConSolid Projects and Consolid References and Annotations. The micro-frontend modules are coded using ReactJS [113]. However, most micro-frontend infrastructures (including Piral) allow to combine modules created in different Javascript frontend frameworks such as Angular [61] and Vue [202], giving more flexibility to developers and making the framework more future-proof. The Bundler[15] and all modules are published on the Github organisation page with URL https://github.com/aecostore.

7.5.1 Bundler-module interfaces

It is important that the Bundler and the modules share a common vocabulary and internal data structure. The Bundler exposes this as a set of functions to the modules, and it is up to the modules to indicate in their manifest what functionality they rely upon. This way, compatibility issues can be checked before loading the configuration. The terminology from the ConSolid project will be reused. For example, to indicate the scale-independence of a 'project', the term 'catalogue' will be used. Conform with the DCAT vocabulary, 'datasets' provide the metadata records, 'distributions' contain the eventual asset information.

This is reflected in the data handling mechanisms as well. Information management by the Bundler happens via an (in-memory) triple store. The overarching data structure is a DCAT catalogue, which resolves to datasets, distributions and lower level catalogues. Similar to the data storage logic in the ConSolid ecoystem, this allows to load multi-project configurations ('all libraries in Flanders') in exactly the same way as single-project configurations. The loaded multi-model can be a direct mirror of a ConSolid project, or a reconfiguration of a centralised multi-model. In the case of a federated multi-model, the in-memory triple store takes the function of an RDF Aggregator (Chapter 6). This application logic also allows to interact with a centralised model, for example, an ICDD project. In this case, a SPARQL CONSTRUCT query can be used that inverts the query in Listings 6.4 and 6.5 (Chapter 6). This will result in an in-memory catalogue that is equivalent to a one-vault ConSolid project, i.e.

[15] Bundler prototype, https://github.com/AECOstore/bundler. Accessed 2023-12-08.

directly referring to its constituent datasets instead of intermediary catalogues for individual stakeholders (partial projects).

To allow interaction between the Bundler and the loaded modules, the following minimal set of functions is considered:

- getData (*parameter*): get information shared with other modules via the Bundler I/O (e.g. catalogue, current selection, ...), either via a SPARQL query or via predefined URLs that resolve to a SPARQL query ;
- setData (*parameter*, *value*): set information to share with other modules via the Bundler I/O ;
- queryProject (*query*, *sources*): query the (cached) project catalogue entirely or partially ;
- findCollectionBySelector (*identifier*, *source*): find the associated Reference Collection and its representations after a selection event ;
- (authenticated) fetch(*url*, *options*): fetch function which authenticates with a valid token (OIDC or OAUTH 2.0).

7.5.2 Modules, Configurations and Store Catalogue

The following micro-frontend modules were published in context of this demo, as pages:

1. Welcome Page[16], containing information about the loaded multi-model.
2. Store Page[17], allows to query a given catalogue and discover and load specific configurations.
3. Authorisation Page[18]: allows the configuration to use a Solid-authenticated access token for discovery and querying of Pod-based project data.
4. Project Page[19]: landing page for ConSolid projects, allows the creation and retrieval of ConSolid Projects and gives an overview of pending project invitations to join existing ConSolid projects.

[16]Welcome Page, https://github.com/AECOstore/welcome-page/blob/main/public/manifest.ttl. Accessed 2023-11-01.

[17]Store Page, https://github.com/AECOstore/store-page/blob/main/public/manifest.ttl. Accessed 2023-11-01.

[18]Authorisation Page, https://github.com/AECOstore/auth-page/blob/main/public/manifest.ttl. Accessed 2023-11-01.

[19]Project Page, https://github.com/AECOstore/project-page/blob/main/public/manifest.ttl. Accessed 2023-11-01.

5. Interaction Page[20]: aggregates the modules for interacting with project data.

In the demo configurations, the Interaction Page will host a combination of the following modules:

1. SPARQL Query Module[21]: queries RDF resources (data and metadata) in the federated project. Allows to interact with concepts via query results. Query results are matched with their abstract concepts, the module communicates these to the Bundler, which redistributes the selection set to the other modules.

2. 3D Viewer Module[22]: discovery and visualisation of glTF geometric models in the project. Allows to interact with concepts via selection of 3D geometry. Selected geometries are matched with their abstract concepts, the module communicates these to the Bundler, which redistributes the selection set to the other modules. The module receives concepts selected by other modules - if these concepts have a glTF-based representation, the viewer will highlight these representations for the end-user.

3. Damage Enrichment Module[23]: allows the end user to semantically enrich concepts selected via one of the aforementioned GUIs (i.e., by geometry or query). Essentially, all modules for semantic enrichment will follow the same procedure, i.e. create a representation of the selected concept in an RDF document on the Pod and then allow enrichment of this representation with the selected ontology, via a user-friendly interface. In this example, the module bases upon the DOT ontology [68] in the background.

4. Pixel Selector Module[24]: discovery and visualisation of imagery in the project. Allows to identify pixel regions and link them to existing or new Reference Collections.

[20]Demo Page, https://github.com/AECOstore/demo-page/blob/main/public/manifest.ttl. Accessed 2023-11-01.

[21]Query Module, https://github.com/AECOstore/query-module/blob/main/public/manifest.ttl. Accessed 2023-11-01.

[22]3D Viewer Module, https://github.com/AECOstore/viewer-module/blob/main/public/manifest.ttl. Accessed 2023-11-01.

[23]Damage Enrichment Module, https://github.com/AECOstore/damage-module/blob/main/public/manifest.ttl. Accessed 2023-11-01.

[24]Pixel Selector Module, https://github.com/AECOstore/image-module/blob/main/public/manifest.ttl. Accessed 2023-11-01.

5. Tabs Module[25]: auxiliary GUI module to host other modules in a tab-based organisation.

While these modules are specifically developed to be compatible with ConSolid projects, similar modules can be developed for centralised multi-models. Configurations[26] and their modules[27] are discoverable via a public catalogue. The pages are published separately[28], as they can be considered more infrastructural and less related to facilitating the interaction with project data. The module catalogue and the page catalogue are, in turn, aggregated into the main store catalogue that will be queried by the Bundler[29]. The content of the access point of the AECOstore is given in Listing 7.5.

```
<> a dcat:Catalog, mifesto:Store ;
    dcat:dataset <https://raw.githubusercontent.com/AECOstore/RESOURCES/
     main/stores/configstore.ttl> ,
    <https://raw.githubusercontent.com/AECOstore/RESOURCES/main/stores/
     modulestore.ttl> ,
    <https://raw.githubusercontent.com/AECOstore/RESOURCES/main/stores/
     infrastructure.ttl> .
```

Listing 7.5: Content of the AECOstore catalogue.

In the next sections, the Mifesto framework will be demonstrated with a case study related to damage enrichment.

7.5.3 Demonstration

The default landing page of the Bundler implementation is itself a Mifesto Configuration[30], aggregating a Welcome module and a Store module. The Store module allows to indicate a Store Catalogue, which can be used as a starting point to initiate discovery and loading of existing configurations. In the store used in this demonstration, a query visualisation configuration and a damage enrichment configuration are aggregated (Figure 7.4).

[25]Tabs Module, https://github.com/AECOstore/tabs-module/blob/main/public/manifest.ttl. Accessed 2023-11-01.

[26]Configuration Store, https://github.com/AECOstore/RESOURCES/blob/main/stores/configstore.ttl

[27]Module Store, https://github.com/AECOstore/RESOURCES/blob/main/stores/modulestore.ttl

[28]Pages Store, https://github.com/AECOstore/RESOURCES/blob/main/stores/infrastructure.ttl

[29]AECOstore, https://github.com/AECOstore/RESOURCES/blob/main/stores/root.ttl

[30]Default Configuration, https://github.com/AECOstore/RESOURCES/blob/main/configurations/welcome.ttl. Accessed 15/11/2023.

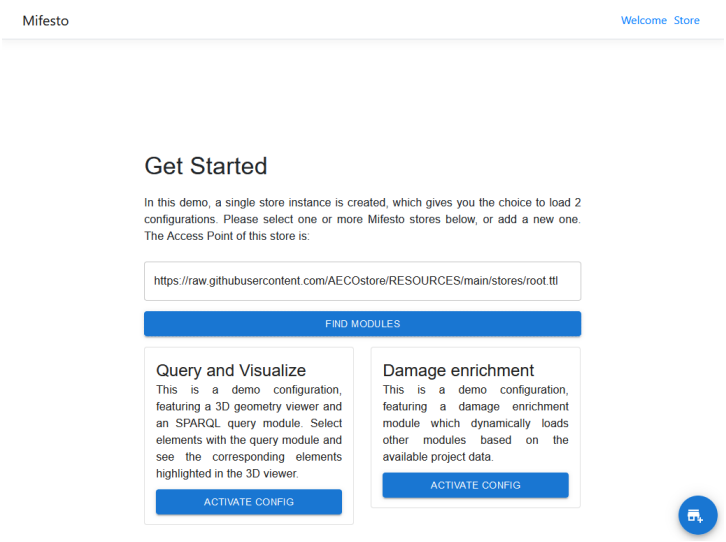

Figure 7.4: The default configuration of the bundler prototype allows to select an existing Mifesto configuration on the "Store" page.

Both configurations consist of the same basic infrastructure, namely the Auth module (as a page, '/auth'), the Project module (as a page, '/') and the Interaction module (as a page, '/project'). The Auth module allows to authenticate to a Solid Server. The project module gives an overview of the projects a user participates in. The Interaction module will host the child modules that are necessary for each specific interaction scenario. The initial sequence of requests to load this configuration is illustrated in Figure 7.5.

The first configuration is intended for querying and visualisation. In this configuration, the SPARQL query module is hosted as a child module by an organisational component, namely the Tabs Module. The query module allows to query RDF-based project resources, and propagate the results corresponding with certain variables to find their associated Reference Collections. The bundler then broadcasts the selection set to the other modules, in this case the 3D viewer. If the selected Reference Collections have a representation in the currently displayed 3D model, this representation is highlighted in the viewer. The semantic description of this configuration is available at the Mifesto Github Organisation[31]. The resulting interface is shown in Figure 7.6.

[31] Query-Visualisation Configuration, https://github.com/AECOstore/RESOURCES/blob/main/configurations/viz_query.ttl. Accessed 16/11/2023.

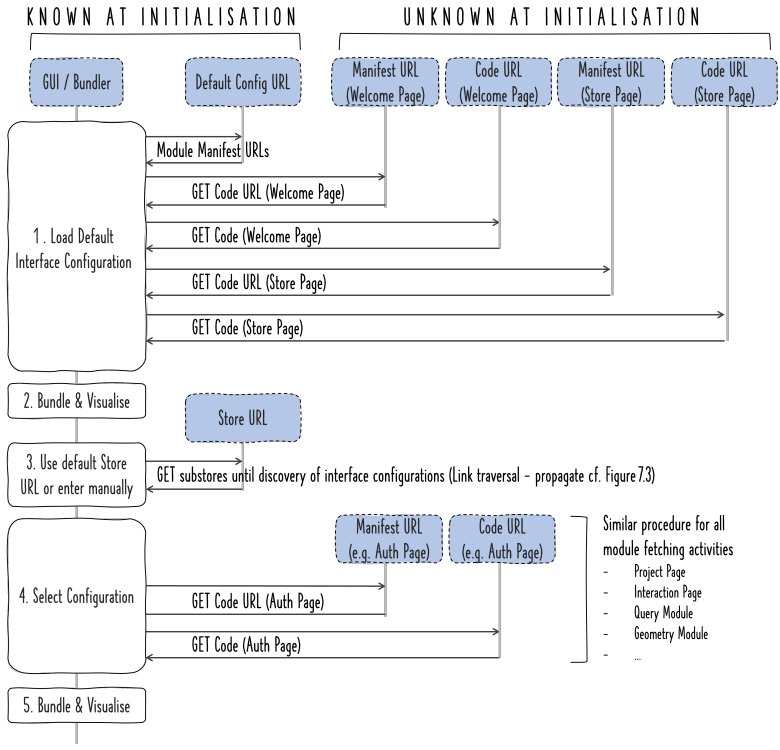

Figure 7.5: Steps for the Bundler to fetch the initial interface, containing a Welcome Page and a Store Page.

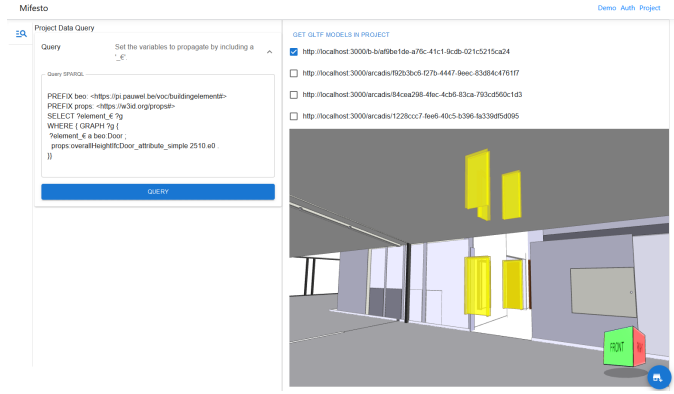

Figure 7.6: The first configuration loads a SPARQL query module combined with a 3D visualiser.

The second configuration considers the case of damage enrichment via heterogeneous resources (Figure 7.7). The main module of this configuration is the Damage Enrichment Module, which bases upon the DOT ontology [68]. The other interaction modules (e.g. imagery or geometry visualisation) can either be an explicit part of the configuration, or dynamically loaded based on the available project resources, using an auxiliary module that first queries the active project for the used media types and then queries the store for modules that are capable of interacting with these media types (e.g. glTF). If such a module is found, it is loaded, allowing the user to create damage records and immediately link them to building elements in the multi-model, through the geometry. This sequence is illustrated in Figure 7.8

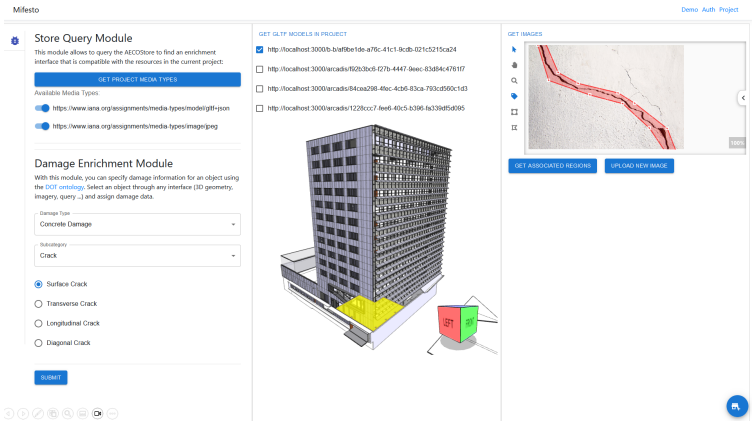

Figure 7.7: Dynamic interface prototype for registering damage records in a federated project. The interface includes a Damage Registration module, and dynamically loads an interface for visual interaction via auxiliary resources, depending on the project content.

Similarly, if the project would only contain IFC files, an IFC visualiser can be loaded instead of a glTF module. When no auxiliaries are available in the existing project, the Damage Enrichment Module may just add the damage records to the multi-model without linking them to other resources – something that can still happen in a later phase. The semantic configuration for this GUI is available at the Mifesto Github Organisation[32].

[32]Damage Configuration, https://github.com/AECOstore/RESOURCES/blob/main/configurations/damage-enrichment.ttl. Accessed 16/11/2023.

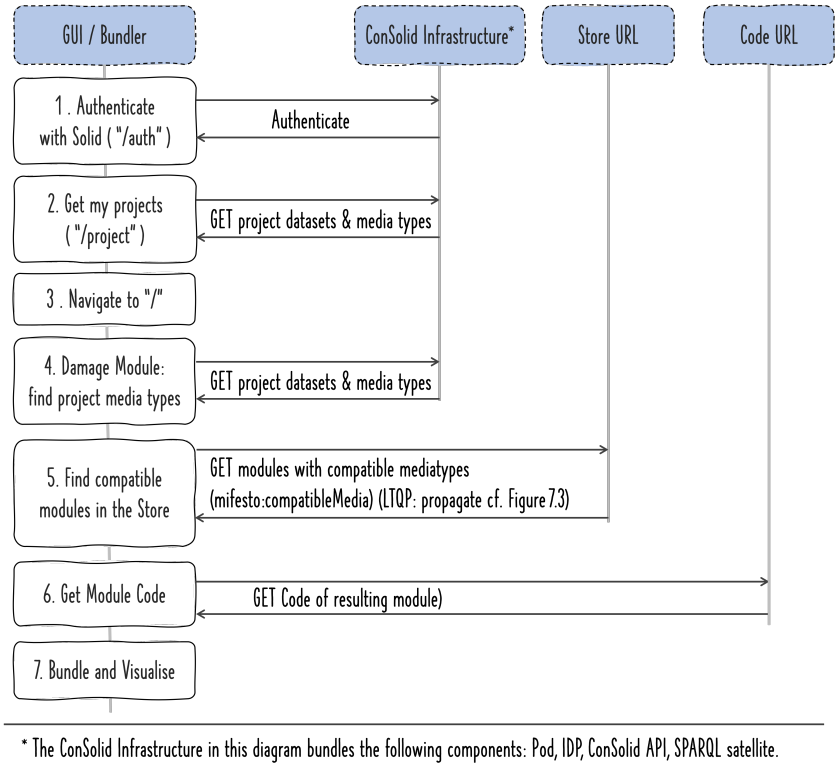

Figure 7.8: Steps for dynamically retrieving modules capable of interacting with project data.

7.6 Conclusion

In this chapter, a novel framework was defined for linking heterogeneous datasets in federated multi-models, called Mifesto. True to the FAIR principles and the reuse potential of data structured in open formats, federated multi-models hold the promise of usability in multiple usage scenarios, some of which are unknown at the time the project is initiated. From a UI perspective, this genericness may seem too big to tackle. However, by applying a minimal set of core design principles, the Mifesto framework allows to link new information to the already existing stack of datasets in a multi-model, using federated GUI modules. Of course, before interaction with a specific media type or semantic enrichment with a certain ontology is possible, a compatible UI module must be present. It was argued logically (but not quantitatively proven) that the effort needed to create a federated module that can be reused again and again,

in combination with other modules, is smaller than repeatedly hard-coding the same interactions in 'monolithic' GUIs (Figure 7.2).

In this conceptual phase of Mifesto, the creation of configurations and manifests happens by manually creating the semantic (Turtle) descriptions. Future implementation work may focus on the creation of GUIs for configuring these in a more user-friendly way.

The framework is compatible with the ConSolid ecosystem, but can be used using other multi-model specifications such as ICDD as well, as these can be reconfigured to a single-vault ConSolid project by the Bundler application. Mifesto is discipline-agnostic, but a proof-of-concept implementation (AECOstore) was created for the built environment. As the AECOstore is just a catalogue, it can be nested into larger catalogues; the same holds for its aggregated sub-catalogues and the eventual modules they refer to.

Mifesto complements existing authoring tools for digitising information about the built environment. Authoring tools are thereby responsible for *creating* the data, which may vary in granularity and scope. For example, one authoring tool may allow creation of a BIM model which already includes both semantics and geometry, or two others can be specialised in creating respectively geometry and semantics. Mifesto allows to *link* diverse datasets in a visual way. In a context of BIM maturity level 3, creation and linking activities may also happen simultaneously within the same configuration. This was illustrated with the case study of damage enrichment.

Table 7.1 gives a summary of the characteristics of a federated micro-frontend ecosystem (as defined in this dissertation). The characteristics are related to the technologies that were identified as suitable for supporting them. When a technology is a 'data technology' (e.g., RDF), more details are provided regarding the data patterns that were developed or adopted.

Characteristic	Technology	Data Patterns
C10 - Modularity	Micro-frontends (Piral.io)	n.a.
C11 - Decentral Publishing	RDF (MIFESTO)	Mifesto Manifests
C12 - Discoverability	RDF (DCAT)	Mifesto Store
C13 - Indirect Enrichment	RDF (ConSolid)	Reference Collections / WADM Selectors
C14 - Common I/O interface	ReactJS (Piral.io Bundler)	Bundler-module interfaces

Table 7.1: Characteristics for a federated ecosystem of micro-frontends, corresponding technologies and data patterns as implemented in Mifesto.

Chapter 8
Evaluation

In the previous chapters, two conceptual, modular frameworks were devised for handling different aspects of a Web-wide CDE. The first ecosystem, ConSolid, deals with the data storage part: how can we aggregate and link knowledge in heterogeneous resources on the Web? The second ecosystem, Mifesto, allows flexible linking and enrichment of heterogeneous, federated multi-models with a wide usage scope.

In this chapter, both ecosystems will be evaluated against multiple evaluation sets. First, the research questions that were formulated in Chapter 1 will be revisited in Section 8.1, evaluating whether this thesis has addressed them to a sufficient degree. Section 8.2 then weighs both ConSolid and Mifesto against the FAIR principles, giving an indication of how well the ecosystems succeed in a providing an infrastructure for general 'federated data management'. Finally, limitations of this research and its outcomes are identified and discussed in Section 8.3.

8.1 Research Results

This section will evaluate whether this dissertation has sufficiently addressed the following research questions, which were introduced in Chapter 1:

- RQ1 What are the technology-agnostic characteristics for a scalable CDE with high potential for discovery of related information, and integration and reuse of existing data sources?
- RQ2 How can these requirements be addressed using current-day standards, Web engineering concepts and technology specifications?
- RQ3 Using the technologies mentioned in RQ2, what are data patterns for structuring, discovering and querying information in a federated environment?
- RQ4 What are data patterns for mediatype-agnostic, cross-resource linking and annotation in a federated environment?
- RQ5 Is such environment compatible with current-day information management practice in the AECO sector?

- RQ6 How can a domain-specialist without extensive IT knowledge link new datasets and their content to an existing federated project catalogue, independent from the media type or present topics of both the new datasets and the existing multi-model?

The order of the research questions indicates a resolution that goes from very abstract considerations into high-level characteristics that were addressed using specific technological solutions, which were extended when necessary, and the data patterns developed throughout the different chapters of this thesis. Regarding RQ1, RQ2, RQ3 and RQ4, separate characteristics for the ecosystem were introduced in each chapter dedicated to data modelling (Chapters 3 and 4), after which technologies were identified that are capable of addressing them. However, these data modelling patterns alone are not sufficient to speak of a CDE. Therefore, additional layers were added in Chapter 5: *Data Validation* and Chapter 6: *Middleware Services*. These chapters make use of specific examples to develop the setup of a *generic* ecosystem, which provides the basis for addressing more CDE-specific challenges in further research projects. Chapter 6 also argues that a federated multi-model is not necessarily incompatible with existing information management practices in the AECO sector, which often relies on data centralisation. This relates to RQ5.

The evaluation of the data ecosystem (i.e., ConSolid) can be done by checking whether the abstract requirements identified in this dissertation were addressed by corresponding technologies. In the conclusion section of every chapter, an overview was given of the identified characteristics, the technologies used to address them in this dissertation and, if more information was required, their corresponding data patterns, which were either developed in context of this dissertation or adopted from other work. Table 8.1 provides an overview of all characteristics related to the data ecosystem and their interpretation for the ConSolid ecosystem.

Characteristic	Technology	Data Patterns
C1 - Decentral, Secure Storage	HTTP(S), Web servers	n.a.
C2 - Decentral Authentication	Solid, WebID-OIDC	n.a.
C3 - Guaranteed Data Heterogeneity	Decoupling ecosystem and project data	dcat:Distribution / accessURL
C4 - Uniform Metadata Descriptions	RDF (DCAT)	dcat:Dataset / dcat:Distribution
C5 - Uniform Query Language	SPARQL	consolid:hasSPARQLsatellite
C6 - Cross-document Annotation	RDF (WADM)	WADM Annotations
C7 - Cross-document Linking	RDF (ConSolid)	Reference Collections / WADM Selectors
C8 - Aggregators	HTTP(S), Web servers	n.a.
C9 - Adaptors	API mappings, Web servers	n.a.

Table 8.1: Requirements, corresponding technologies and data patterns in ConSolid.

Additionally, a conceptual framework to make interaction with such federated multi-models of wide usage scope more convenient for domain specialists (RQ6) has been introduced in Chapter 7 as Mifesto. This framework bases upon modular GUIs consisting of federated micro-frontend aggregations. Table 8.2 revisits abstract requirements, technologies and data patterns, this time applied to Mifesto.

Characteristic	Technology	Data Patterns
C10 - Modularity	Micro-frontends (Piral.io)	n.a.
C11 - Decentral Publishing	RDF (MIFESTO)	Mifesto Manifests
C12 - Discoverability	RDF (DCAT)	Mifesto Store
C13 - Indirect Enrichment	RDF (ConSolid)	Reference Collections / WADM Selectors
C14 - Common I/O interface	ReactJS (Piral.io Bundler)	Bundler-module interfaces

Table 8.2: Characteristics, corresponding technologies and data patterns in Mifesto.

The above-mentioned characteristics make an explicit distinction between data storage, metadata patterns, resource data types and interaction patterns. Since most of the technologies applied in this dissertation are still relatively novel at the time of writing, such separation will render the over-arching ecosystem more resilient. Although the combination of these different components of the ecosystem will render the most powerful environment in terms of compatibility, subsequent research may thus well base upon one single aspect without the need to adopt the entire ecosystem. Thus, neither ConSolid or Mifesto is to be seen as an 'all-or-nothing' environment.

8.2 FAIR evaluation

The FAIR principles are considered apt evaluation criteria, given the intended general applicability of the ecosystem. The FAIR principles are subdivided into sub-requirements for each principle [60].

8.2.1 ConSolid

This section lists the FAIR sub-requirements and explains them from the perspective of the ConSolid ecosystem:

- **F1. (Meta)data are assigned a globally unique and persistent identifier**: Every resource is identified with a URL, which is made as stable as possible by eliminating implicit semantics. URLs may change in a data handover process, but in this case, sub-document links can be maintained.

- **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**: Metadata records indicate the data they describe via DCAT distributions and access URLs.
- **F4. (Meta)data are registered or indexed in a searchable resource**: The ecosystem is based on recursive catalogues 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. An RDF Aggregator can offer a searchable and access-controlled graph of metadata and RDF-based project resources.
- **A1. (Meta)data are retrievable by their identifier using a standardised communications protocol**: All resources on a data vault can be retrieved via an HTTP URL.
  - **A1.1. The protocol is open, free, and universally implementable**: The HTTP(S) protocol adheres to this requirement.
  - **A1.2. The protocol allows for an authentication and authorisation procedure, where necessary**: authentication and authorisation is possible with HTTP(S).
- **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 may enforce this.
- **I1. (Meta)data use a formal, accessible, shared, and broadly applicable language for knowledge representation**: Metadata are recorded using RDF, which is the main W3C framework for knowledge representation.
- **I2. (Meta)data use vocabularies that follow FAIR principles**: In the FAIR data points specification [119], the DCAT vocabulary is indicated as the basis for metadata content. Dataset Collections in ConSolid are based on DCAT data patterns.

- **I3. (Meta)data include *qualified references*[33] to other (meta)data**: In the ConSolid ecosystem, metadata is formed by a union of the data catalogues on a vault and the Reference Registry. This union can be queried via the SPARQL satellite, to retrieve dataset aggregations and complementary information for project resources. Indication the reliance of (meta)data on other (meta)data is possible, but cannot be enforced.
- **R1. (Meta)data are richly described with a plurality of accurate and relevant attributes**: According to [60], 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. As DCAT metadata is graph-based, it can be combined with domain-specific vocabularies in a straightforward way, in order to allow interdisciplinary purposes.
    1. **R1.1. (Meta)data are released with a clear and accessible data usage license**: Although not explicitly discussed in this dissertation, a license document can be linked to datasets and distributions, e.g., using `cc:license` [46]. Machine-readable license information (RDF) is provided by Creative Commons[34].
    2. **R1.2. (Meta)data are associated with detailed provenance**: Although not explicitly discussed in this dissertation, provenance information on can be linked to RDF-based metadata, e.g., using the PROV-O ontology [107].
    3. **R1.3. (Meta)data meet domain-relevant community standards**: In Chapter 5, it was shown how metadata conformity to specific shapes can be checked. This means that relevant metadata statements can be agreed upon on a project-specific (and thus domain-relevant) basis. In the context of construction data, the CDC ontology was mentioned. Note that the same procedure can be used to validate whether license (R1.1) and provenance (R1.2) data is present. However, the application of such shapes is not *enforced* in this dissertation's implementation of the characteristics, i.e., ConSolid.

[33] In [60], qualified references indicate 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.

[34] CC License RDF: https://wiki.creativecommons.org/wiki/License_RDF. Accessed 2024-03-21.

8.2.2 Mifesto

This section lists the FAIR sub-requirements and explains them from the perspective of Mifesto:

- **F1. (Meta)data are assigned a globally unique and persistent identifier**: Every Mifesto manifest, code, configuration and store is identified with a URL.
- **F2. Data are described with rich metadata (defined by R1 below)**: All resources (manifests, configurations, stores) are described in metadata records. Because these records are RDF-based, they can be used for fine-grained filtering and discovery of these resources.
- **F3. Metadata clearly and explicitly include the identifier of the data they describe**: Metadata records indicate the data they describe via DCAT distributions and access URLs.
- **F4. (Meta)data are registered or indexed in a searchable resource**: The Mifesto ecosystem is based on recursive catalogues to allow clients to index relevant (meta)data on-the-go and cache this index for future usage. Link-traversal-based queries can search a federated store catalogue on-the-go. An RDF Aggregator can be used to provide a faster querying service.
- **A1. (Meta)data are retrievable by their identifier using a standardised communications protocol**: Both the metadata records (the manifests) and their distributions (the code) can be retrieved via a HTTP URL.
  - **A1.1. The protocol is open, free, and universally implementable**: The HTTP(S) protocol adheres to this requirement.
  - **A1.2. The protocol allows for an authentication and authorisation procedure, where necessary**: authentication and authorisation is possible with HTTP(S).
- **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 by the publishers of Mifesto modules, the archival of (meta)data and retention policies can not (yet) be controlled in a technical way.

- **I1. (Meta)data use a formal, accessible, shared, and broadly applicable language for knowledge representation**: Metadata are recorded using RDF, which is the main W3C framework for knowledge representation.
- **I2. (Meta)data use vocabularies that follow FAIR principles**: in the FAIR data points specification [119], the DCAT vocabulary is indicated as the basis for metadata content. The structure of a Mifesto Store is based on DCAT data patterns.
- **I3. (Meta)data include qualified references to other (meta)data**: In Mifesto, a store is formed by the aggregation of its constituent subcatalogues and datasets. As for the metadata descriptions, we can say that aggregation can only take place when there are qualified references to such other catalogues. Regarding the *code* of the individual modules (`mifesto:code`), we can say that their functional decoupling from other modules is what gives them their flexibility.
- **R1. (Meta)data are richly described with a plurality of accurate and relevant attributes**: the Mifesto vocabulary includes attributes that allow fine-grained filtering of Mifesto manifests. Other vocabularies can be used to provide further semantic details, such as licensing (R1.1), provenance (R1.2) and domain-specific information (R1.3).

8.3 Limitations

In this section, limitations of this research will be identified. A differentiation will be made between conceptual limitations, technological limitations and limitations of the overall research.

8.3.1 Conceptual Limitations

One may wonder whether the concept of a 'pure' SSoI is compatible with the notion of multi-models and FAIR data stewardship. In an 'only IFC' or 'only Revit' environment, ambiguities and clashes can be discovered relatively easily, making automatic maintenance of an SSoI in the form of a coordination model possible [151]. However, as argued in this thesis, when only a single domain-specific data model is to be used, the potential for interdisciplinary data reuse is quite limited. This holds for both proprietary and open data models. Multi-model containers were introduced because not all relevant information can be expressed using the same data formats. In this sense, *a supervised co-existence*

of several data formats will be accepted as a basis for an SSoI, but information may still be duplicated in resources that follow different schema's. Hence, the risk for ambiguities persists. However, the metadata-based registration of sub-document links allows to identify similarities and formulate annotations to clarify the relationship. In this sense, the gist of an SSoI (especially a federated one) is less about not having any ambiguities than about being able to discover them, and deal with them in a way that is optimal for the project.

8.3.2 Technological limitations

A first technological limitation is the expected decrease in performance of the ecosystem in larger configurations, with more vaults and stakeholders. Although performance was explicitly not a part of this dissertation, it is an important criterion for valorisation. The iGent building, which served as the input for this dissertation's case study, is a medium-to-large building, resulting in a basic input set of around 350 000 triples, and around 70 000 more for the Reference Collections. When more disciplines and actors become involved, this may increase to 1-10 million or even more. However, present-day triple stores are capable of efficiently handling these amounts of triples, so no major problems are expected regarding this aspect.

Another factor is the granularity of project data. In ConSolid, resources are still largely considered as 'permeable silo's, but on the other hand everything that can be fetched by an HTTP request may be considered as a dedicated resource (LDP). The granularity of project data can theoretically increase indefinitely, but this comes with a significant overload in metadata statements. For example, one could consider a 3D model where all elements are stored in separate resources with separate access rights. While this enables powerful data management tools, every resource will have its own metadata record and ACL listing. The metadata record will also have its ACL listing. This means that there will always be a pay-off between granularity, storage capacity and performance of the ecosystem. The same holds for the reliance on Reference Collections: the more 'abstract concepts' (elements, damages, persons...), the more Reference Collections, accompanied by a significant amount of additional triples per representation and per sub-document selector. In brief: the approach in ConSolid is not the most efficient one in terms of storage space. In terms of access control, the queryable union graph presented by the SPARQL satellite was evaluated for a few hundreds of resources in Chapter 3, and some strategies were devised for decreasing query time by more precise querying, and better integration of the partial services involved.

Finally, the degree of federation presents another limitation. While from a data storage perspective, the ecosystem can be scaled indefinitely, in practice, performance issues are likely to play a significant role in multi-building configurations, which were only theoretically described in this dissertation. The same limitation can be identified for single-building catalogues with a large number of data vaults – for example, when the vaults of inhabitants (e.g., their preferences and (physical) access control rights) become part of the catalogue. The setup of dedicated RDF aggregators may offer solace here, but further research is required to prove their viability in such context.

8.3.3 Research Limitations

This dissertation has the intention to give a broad overview of the topic of data federation in AEC, while offering concrete solutions for achieving a generally applicable approach. Inevitably, because of the limited scope of a PhD research project, some of the topics were only discussed theoretically (e.g., timeseries satellites), or implemented only until a certain level of complexity (PBAC and Mifesto Ontology), or only touched for illustrative purposes without developing a ready-to-use solution (e.g. ICDD and BCF). These limitations are largely influenced by time constraints, and can hence certainly be further developed and evaluated in future research.

In terms of dissemination, the genericness of the topics described in this research may make it difficult to imagine what can be done with the devised frameworks, inhibiting its adoption. The limited scope of this research did not allow to at the same time devise the abstract requirements and take a deep dive regarding specific applications. However, the question 'what can we do with such generic ecosystem' is actually similar to the question 'what can we do with FAIR (or open) data': its main value lies in the ability to connect formerly unconnected, federated and access-controlled datasets, in order to gain new insights to use in professional practice. Practical applications of the frameworks, and their valorisation potential, will be discussed in Chapter 9.

Chapter 9
Conclusion

This dissertation has made a case for a Web environment where resources of diverse media types and schemas may be aggregated in a federated manner and linked to one another on a sub-document level. The different aspects of such environment were described in 7 chapters, which subsequently addressed a general rationale (Chapter 2) data storage (Chapter 3), cross-resource linking (Chapter 4), validation and access control (Chapter 5), aggregation and adaptation (Chapter 6), visual linking and enrichment by end-users (Chapter 7) and an evaluation of the initial goals and research questions (Chapter 8).

Section 9.1 lists the contributions of this dissertation to the field and the industry. The research has led to a number of findings, which can be found in Section 9.2. In Section 9.3, key limitations are outlined that can be considered for future work. Finally, Section 9.4 offers a non-exhaustive overview of valorisation scenarios based on the findings of this dissertation. Sections 9.3 and 9.4 will be structured referencing the Technology Readiness Levels (TRL) [172], a measurement scale for the maturity and viability of new technologies.

9.1 Contributions

The contributions of this thesis can be grouped under a number of higher-level categories:

1. Multi-model federation:
    (a) Outline of the characteristics for a CDE for federated, heterogeneous multi-models;
    (b) Identification of existing technologies capable of addressing the characteristics;
    (c) Outline of *design choices* and *data patterns* compatible with the identified technologies;
    (d) The conceptual combination of micro-frontends with semantic, federated catalogues, and the identification of the benefits this offers;

2. The Solid ecosystem (research towards multi-pod environments):
    (a) Access-controlled union graph of a Pod (SPARQL satellite);
    (b) Pattern-based access control (PBAC);
    (c) Access-controlled RDF aggregators;

This resulted in numerous external deliverables beyond this dissertation text, amongst which a proof-of-concept implementation. These deliverables (i.e., vocabularies and codebases) were published online with an open license.

In the following sections, the contributions listed above will be further elaborated, including an additional section on dissemination.

9.1.1 Multi-model Federation and Enrichment

The idea to organise as much information as possible using RDF has long prevailed in the field of Linked Building Data. The result is a hierarchical approach to data structures: RDF representations are made almost mandatory in order to link other heterogeneous datasets. Moreover, in these RDF graphs, metadata and actual project semantics are often intertwined. The concept of multi-models and the corresponding standards, which were established in other research projects, allows to take a more neutral stance: by separating metadata and project resources, building data can (but should not necessarily) be RDF, but other datasets (imagery, geometry) can be part of the overall project catalogue as well. In this dissertation, this core idea of multi-models, which are currently largely applied as 'information containers', was extended to a federated, dynamic approach. Applying a federation logic from the beginning, it was shown that by imposing a recursive structure on simple data patterns, a powerful infrastructure enables query-based discovery and filtering of relevant project datasets, which may be hosted anywhere on the internet. To create these data patterns, standardised, domain-agnostic data patterns such as DCAT were used whenever possible (Chapter 3).

Centralised solutions for interacting with heterogeneous multi-models will be able to cover most interactions that stay within the disciplinary boundaries of the AECO industries. In a common BIM authoring tool, assigning properties to a building element is typically done by selecting a 3D representation and then entering the values of the property – semantic enrichment happens almost exclusively using 3D geometry as a proxy. However, for multi-disciplinary tasks that do not occur very frequently or involve specific data formats or novel ontologies, the chances decrease to have a 3D model at your disposal.

This dissertation identified the presence of linksets in common multi-model approaches as a key enabler for a modular GUI framework that allows *many-to-many* enrichment. Namely, through the linkset (or Reference Collection), several digital representations can be connected to one another with zero knowledge about the other representations. The freedom of granularity of documents in a multi-model allows project topics to be described in separate data pools without impacting the connections with other project data. As a consequence, the most fit data types can be chosen for specific topics. In this sense, the concept of 'enrichment' applies to almost any activity where information is added to the overall catalogue.

By applying a semantic layer to the existing concept of micro-frontends, several benefits were achieved compared to 'monolithic' solutions. Firstly, a catalogue of interfaces based on Semantic Web technologies allows to publish these interfaces in a federated way, allowing 'permissionless innovation' [176], and query them to discover the modules necessary for a specific interaction (e.g., damage documentation, image linking, ...). The reasons for federating interface modules are thus similar to data federation, although they are driven by different concerns. Secondly, with a given 'module configuration GUI', end-users are able to address their case by querying a micro-frontend store, configuring new interfaces configurations or loading existing ones, without extensive programming knowledge ('low-code'). Each module thus only needs to be specialised in one data format or ontology (Chapter 7). An activity of damage documentation may be carried out either independently from other modules, or linked to available pictures, a 3D model, a combination of both or any other available project dataset. Furthermore, on-the-fly configurations may be generated based on the available data types, thus shaping the user interface according to the needs of the project.

9.1.2 The Solid Ecosystem

This dissertation did not actively contribute to further development of the Solid specification. However, it became clear that, in order for it to be useful as the basis for a federated CDE, conceptual extensions and a higher-level data management practice were necessary. After identifying these extensions, potential technological approaches were proposed to address them. For example, the extension of Solid Pods for query-based resource discovery, the usage of rich metadata structures and the handling of these metadata records as 'normal' resources on a Pod (being a metadata record or not just depends on whether the resource describes itself as such) (Chapter 3). As Solid Servers by design

do not consider the content of data, higher-level interfaces ('satellites') were used to facilitate more complex interactions with data vaults. In concreto, a SPARQL satellite was designed and implemented as a prototype. The SPARQL satellite allows to query a permitted union graph of a data vault, which is a novelty in the Solid ecosystem.

With the use of the Solid ecosystem comes the advantage of a single authentication system in an entire network of CDEs, using the WebID-OIDC [37] protocol or alternatives such as an OAuth 2.0 token [69]. It was mentioned that mere compatibility with this protocol would already mean an important step towards cross-CDE document exchange and interoperability. The default WebID-based authentication mechanisms in Solid also provide the basis for federated authorisation using ACL [32] or ACP [23] access control rules. This dissertation extended these mechanisms using a 'Pattern-based Access Control' approach, so both resources and visitors can be checked against specific requirements, before access is granted (Chapter 5, Section 5.3). Using encryption public keys, it can be verified whether a statement was issued by a trusted authority on a specific topic. This way, the expressivity of RDF ontologies can be fully used for access control purposes. This directly contributes to the granularity and flexibility of access control in the ecosystem. The compatibility of PBAC with the ACP protocol was not validated in this dissertation and remains to be done.

Furthermore, a methodology was devised for aggregated SPARQL endpoints of project data, using signed tokens which include the allowed resources (Chapter 6, Section 6.4). In this way, a well-performing (read-only) mirror of project data can be used to interact with project information.

9.1.3 Dissemination: Federation in the Built Environment

A final contribution of this research is less quantifiable, and relates to stimulating the discussion on the topic of data federation in the built environment. Throughout the years, the findings of this research have sparked the debate on centralisation versus federation, more specifically in context of federated CDEs. Research on Linked Data in architecture and construction has been carried out for almost two decades, but in most cases this data remained centralised in a single triple store, or required an ad-hoc setup of numerous accounts on various platforms. This dissertation supported the case to no longer consider centralisation the only option for a CDE, but instead consider centralised environments as nodes within a larger, federated environment. The starting point, namely the high degree of decentralisation in the industry (the 'double

patchwork'), lead to an extensive rationale in Chapter 2. The core arguments of this rationale were disseminated on many an international occasion, and have been well-received by the community.

9.2 Findings

9.2.1 Federation: the Web as a CDE?

One of the core ideas behind this research was to define technology-agnostic requirements to 'use the Web as a CDE', and then devise a technological solution for addressing those requirements. Whether ConSolid describes an infrastructure that succeeds in this goal, depends on the interpretation of what a CDE is. In this dissertation, the term 'CDE' has been used in a very broad way, as a collaborative environment for interacting with heterogeneous data, primarily about the built environment. The federated aspect implies that some combinations of services will lie closer to providing a CDE than others. In this sense, ConSolid itself cannot be considered 'a CDE'. Rather, it is a general set of data patterns and concepts that can be applied in a layered way: vaults and satellites provide data storage and validation, (chains of) middleware services facilitate views on this data that allow standardised (BCF, ICDD, ISO 19650) or customised interactions with project data. Specific combinations of services and vaults can then be created and labelled as a CDE.

The same holds for the interpretation of the term 'multi-model', which has been stretched beyond the typical AECO-related examples. From a data storage perspective, the framework effectively allows to aggregate data on the Web into a project catalogue, which becomes a synonym for a federated multi-model that contains both sensitive data and open data. In the first case, i.e., sensitive data, a metadata description and a resource are both stored on an access-restricted vault. In the second case, i.e., openly accessible data, only the metadata is stored on a vault. This metadata describes the role of the external resource in the project, and allows to aggregate it in the bigger catalogue. The inherent federated and domain-agnostic structure of the data catalogues used in this dissertation, and their extensibility across the Web, allows them to be labelled as multi-models with potentially unlimited size and scope.

On a larger scale, catalogues can be aggregated in higher-level 'matryoshka' catalogues, creating multi-asset aggregations. Such flexible upscaling of the size of a 'catalogue of interest' is only possible in a federated Web environment. Hence, this is an example of using the Web as a CDE. In contrast with aggregations managed by a dedicated consortium (where project access points

recursively refer to one another), such high-level catalogues will be largely unidirectional and read-only. Furthermore, this dissertation only discussed the data patterns that *allow* such higher-level catalogues, and does not dwell on who should *create* them. Their main application is expected to be in the fields of data analysis, for instance related to typology or geographic location. As these topics relate to actual datasets and not to the ecosystem's data patterns, they were not covered in this dissertation. However, they are expected to be useful for individual offices (e.g., learning from previous projects), research purposes (e.g., urban analysis), multi-dwelling residencies or governmental institutions (e.g., facility management and operations). The performance of data gathering for multi-project catalogues was not tested.

9.2.2 Heterogeneity of Data and Documents

Because ConSolid relies on the FAIR principles applied to *metadata*, resources can use any (domain-specific) schema or RDF ontology, aligning with the concept of heterogeneous multi-models. The ability to connect identifiers on a sub-document level is thereby essential. When resources use open standards, sub-document linking is rather straightforward, as identifiers are often embedded in a publicly documented JSON or XML structure, or use a standardised 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 that owns the data format offers an API that is able to parse the resource. For example, the Autodesk Platform Services (APS) (formerly Autodesk Forge) [5] 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 a registration of a damage record, based on semantic statements, imagery and geometry. A similar approach may be identified for other unstructured dataset types such as point clouds. Safeguarding heterogeneity in the ecosystem by separating metadata and project data is essential to ensure the ecosystem's future-proofness. In this way, the whole spectrum between 'full Linked Data' configurations and 'non-Linked Data' configurations can be covered – it is not influenced by the architecture of the platform but can be a project-specific choice. This is reinforced by the Mifesto front-end ecosystem, which allows to gradually broaden the scope of a multi-model by looking at the already available resources and the intended 'enrichment' task.

The BIM maturity level 3 envisages a data-oriented practice, as opposed to the other maturity levels, which are labelled as 'document-oriented'. The interpretation of this dissertation, however, allows documents and databases to co-exist. In other words, documents are less thought of as 'data silos' and more as permeable collections that can reference each other's contents and can likewise be referenced in other datasets living on the Web. Reference Collections (or Linksets in centralised multi-models) thereby allow an independence of the data storage strategy. In turn, this allows to arbitrarily subdivide individual resources into smaller subsets, i.e., increasing the granularity of the multi-model without altering the interaction options. From the over-arching perspective of a multi-model, there are no differences between storing a 3D building model as one single file or storing it as hundreds of files (e.g., one per object), with possibly each one having its own specific access control rules. Similarly, whether semantic properties are stored together with the geometry in a single BIM model, or separated in multiple documents (e.g., an RDF graph or a spreadsheet) has no real impact on the fact that data in these documents can be nevertheless interlinked. Note that this only considers the data patterns – data discovery and retrieval will logically take longer in situations with higher granularity. Furthermore, the possibilities to efficiently interpret the interlinked resources will be higher when common formats are used.

9.2.3 Collections

One of the recurring principles in this dissertation is the principle of collections, applying recursivity and aggregation chains. This makes it possible to find the same results, independent from the original access point. These principles could be seen in action in almost all chapters of this dissertation. The Dataset Collections of Chapter 3 make use of a recursive pattern to identify partial projects, and an aggregation chain to traverse the catalogue until the project data is found. Both patterns base on the property `dcat:dataset`. The Reference Collections of Chapter 4 recursively aggregate other Reference Collections ('aliases') to inform the user which representations can be used as a proxy to enrich a specific concept with new sources or knowledge. Shape Collections (Chapter 5) are a subtype of Dataset Collections. Therefore, they can make use of the same aggregation chains. Finally, the Micro-Frontend Store can be federated and queried due to nested aggregations of UI modules, and hence functions as a 'Micro-Frontend Collection' (Chapter 7).

9.2.4 Resource 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 filters instead of the fixed hierarchy (as used in Solid's implementation of LDP containers), eliminates the need for implicit semantics in the URL and allows a more flexible aggregation of project resources, depending on the task at hand. To make such filtering sufficiently well-performing, however, the setup of an access-controlled SPARQL satellite to a Solid Pod is required.

Changes in document statuses or other metadata tags will not break the resource's URL, guaranteeing stability during the resource's publication life cycle. After all, aggregations of resources can happen flexibly based on queries on metadata and existing `dcat:Catalog`s. However, when the *ownership* of data changes, the domain name and hence the root of a URL is likely to change as well - breaking potential semantic links in project resources. One scenario is the shift from a project's construction phase to the operational phase, which is often accompanied by a data handover between stakeholders in a so-called 'soft landing' [137]. The existence of concept aliases and the separation of heterogeneous identifiers and their allocation (or validity) in a specific resource, discussed in Chapter 4, allow such data handover: when a project document is allocated on a new Pod, on a resource level its sub-document identifiers will remain valid within the resource; and on a metadata level Reference Collections can be created on the new Pod. Asynchronously, they can then be re-aligned with existing aliases in Reference Registries by other stakeholders of the project(s) it is part of. When this has happened, the references to this document in the 'old' Reference Registry can be safely removed. A prerequisite is, of course, that this change of ownership propagates through the network, so other stakeholders can add the new aliases of the concepts referenced in the document that changed owner. While this dissertation describes the logical steps to achieve such a handover, this was not tested in the case study. There are several options to do this, which can be combined to make the process more robust (see Section 9.3.

9.2.5 Compatibility with Existing Industry Practice

The full separation of metadata and resources in ConSolid has as a side effect that the content of a resource is not affected by the membership of one or more ConSolid projects. For example, a BIM model can still be opened in the authoring tool that was used to create it. In this sense, the ecosystem

can be present as an invisible layer, opening up new possibilities for data sharing and linking, without disrupting existing industry practices. In the same way, structured metadata information can be aggregated and adapted to fit international standards such as ICDD (Section 6.6). Project data, if present in a highly structured form (cf. the 5 stars of data structure), can also be reconfigured by middleware services to allow information exchange between the federated ecosystem and existing tools. This was illustrated with an example related to the BCF API (Section 6.7).

9.3 Future Research

The infrastructure devised in this research significantly differs from existing BIM and CDE platforms. Although multiple steps were taken to allow integration of the federated infrastructure with existing standards and practices, a rocky road still lies ahead before the infrastructure can effectively be used in a real-world setting. The Technology Readiness Levels (TRL) [110] define 9 levels for technological maturity. It is estimated that both ConSolid and Mifesto have reached TRL3-4. TRL3 corresponds with *'Experimental proof of concept'* [172]. TRL4 corresponds with *'Technology validated in lab'*. Although the concept of 'laboratory experiments' does not directly map to the technologies discussed in this dissertation, it can be argued that a modular approach was devised from the start of the research, the critical function of the modules of the ecosystems was tested and their mutual interactions were illustrated with an example using BIM models from an existing medium-sized building. However, neither ConSolid nor Mifesto were tested in a real-life environment with AEC stakeholders. Instead, for ConSolid, an exemplary case of damage enrichment was used for demonstrative purposes, as this case embodies multiple of the core goals of the ecosystem, namely interdisciplinarity beyond the typical AEC domains, interaction between multiple stakeholders and management of heterogeneous data sources. For Mifesto, conceptual examples were developed to illustrate the design patterns of the ecosystem.

TRL5 and TRL6 respectively validate and demonstrate a technology in an industrially relevant environment. For these (and further) TRLs to become within reach, and thus for the ecosystems to become effectively useful for the AECO (and other) industries, numerous topics are still to be researched as future work.

9.3.1 Data Redundancy

A first consideration relates to data redundancy. When a data vault can be maintained by project partners instead of a specialised data storage provider, a certain degree of data redundancy needs to be present in order for the consortium to keep functioning when the data of one of the partners becomes unavailable. An in-depth discussion of such infrastructure was not considered in this dissertation and remains to be done in future work. However, a few key points can be made. As the LDP interface to a vault directly maps the availability of a resource to a particular domain, the uptime of the servers related to this domain determine whether the resource can be fetched. In the case of RDF Aggregators, the (LDP) URL of a resource is used as a named graph, so even if the URL is not available, an RDF Aggregator can still be queried. In the case of non-RDF data, the option exists for the metadata (RDF) to point to a secondary access URL in case the first is not available.

9.3.2 Data Handover

In Section 9.2.4, an approach for data handover was briefly sketched. However, this was not explicitly validated. Also, this dissertation lacks a discussion on the meaning of a handover activity in a federated environment. Existing standards such as ICDD can facilitate transmission of numeric data (no geometry) for the purpose of operations. However, a main idea in a federated environment is that information remains available throughout the asset life cycle (following the FAIR principles), so a data transmission to another Pod triggers multiple questions. For example, who is the owner of the data, does data extraction from a BIM model for operations result in information duplication, and is data extraction even necessary when a middleware service exists that offers an 'operations-oriented' view on the – already existing – generic project data? Another question is whether the URLs of project data must be related to the actors that created the data, which would reduce data handover to a transfer of domains, or even of just access rules.

9.3.3 Versioning

Versioning of project data was largely omitted in the scope of this dissertation. From the perspective of the ecosystem, project resources are heterogeneous. Hence, it cannot enforce a specific versioning mechanism, as different formats will have different versioning mechanisms. However, in the case of metadata, which is uniformly structured as RDF, a versioning mechanism can be used to indicate both the most recent version of the metadata record and the

most recent distribution. The DCAT 3 specification introduces a versioning mechanism which can be used to trace the history of a particular dataset or distribution. In order for this to be part of the ConSolid ecosystem, the query templates used for resource discovery should include these patterns.

9.3.4 Data Access

The PBAC framework introduced in Chapter 5 set the lines for a flexible, pattern-based access control system on Solid Pods. However, some complexities were not addressed in this dissertation, such as how to deal with conflicting rules or how the framework scales with an increasing amount of rules. Compatibility with the official ACP protocol needs to be checked. The example of access delegation was briefly covered in the use case, based on the statements of an explicit and an implicit authority. However, more complex cases should be investigated, both regarding the chaining of certificates and the validation procedure. Finally, the current procedure is based upon JWT tokens [92]. Allowing the PBAC certificates to be created and signed conform upcoming Web recommendations such as the Verifiable Credentials (VC) specification [163], will make the framework more interoperable and future-proof. This being said, such replacement will only affect the extraction of the statements and validation of the signature, and does not influence the procedure for checking the joint graph of valid certificates against the established PBAC rules.

9.3.5 Authority and Responsibility

Currently, the ecosystem fully supports the AAA slogan of the Web: Anyone can say Anything about Anything. The data patterns in the project allow anyone in the consortium to contribute to the project, but there are no validation mechanisms to check whether an actor has the authority to make specific contributions. For example, no one but the structural engineer should commit datasets related to the structural integrity of the building. Although a resource topic can be indicated in its metadata record, the resource may of course still contain other data. The complexity of checking this increases because of the ensured heterogeneity of project documents in the ecosystem. In this regard, the use of RDF-based semantics may offer certain advantages compared with other solutions. In this dissertation, a workflow was outlined where authorities can be defined and allowed to make trustworthy statements on particular topics. This workflow was primarily applied to access control. Future research should determine whether the same processes and data patterns can be used regarding the trustworthiness of (RDF-based) project data as well.

9.3.6 Automating the Ecosystem

In Chapter 6, a service layer was described, based on the concepts of Aggregators and Adaptors. Theoretically, larger (headless) service networks could be build. This is not a hard requirement to achieve TRL5-6, but may be seen as a suggestion for more thorough research regarding the service layer of the ecosystem. This dissertation did not include experimental validation of the performance and interoperability challenges that arise with the setup of complex networks of chained services.

Extended automation scenarios become possible with a machine-readable project schedule, which can be hosted on one of the project vaults. Apart from an automated initialisation of the project infrastructure (catalogues, reference registries ...), a schedule can describe different tasks in the project, which can in turn be linked to known services that follow a certain standard or require an application-specific data structure. When the services document their requirements on project data or metadata, these requirements can be used to structure project data from the early beginnings of the project. The Shape Collections devised in Chapter 5 can provide a foundation for this, although currently these only consider validation of RDF data using SHACL.

9.4 Valorisation

Once the above-mentioned topics (Section 9.3) have been addressed or further refined, the feasibility of the ecosystems in an industry environment has been proved. However, to reach TRL7, an actual *'system prototype must be demonstrated in an operational environment'*. This means the codebases developed in this dissertation as prototypes must be integrated and professionally implemented following best practices. A TRL8 and TRL9 means that the technology is complete and qualified, opening possibilities for further valorisation of this research.

This dissertation discussed the requirements for a federated CDE of heterogeneous multi-models, in the context of the many-headed monster called 'the built environment'. Only an abstracted, discipline-agnostic approach to all discussed topics allowed to address them in an integrated way. Consequently, the diverging nature of the discussed topics allows to apply its findings to entirely different domains than the built environment, giving the research a strong long-term valorisation potential. The domain-independence of Consolid and Mifesto was set as an explicit research goal, but since the AECO industries have provided the main context for this dissertation, the following

(non-exhaustive) list of valorisation scenarios will be limited to topics relevant for the built environment:

1. Relating building data and GIS datasets from different data owners:
    (a) Infrastructure and asset information (e.g. energy grids);
    (b) As-built models and user preferences;
2. Relating building data and historical datasets:
    (a) Heritage objects and historical events ('virtual museums');
    (b) Renovation of heritage objects;
    (c) Catalogue setup of archived asset sources;
3. Relating building data and external product datasets:
    (a) HVAC maintenance;
    (b) Interior planning;
4. Building data and governmental regulations:
    (a) Accessibility regulation;
    (b) Fire safety regulation;
5. Multi-asset aggregations for urban or typological analysis:
    (a) Social analysis (asset data and personal data on a vault) ;
    (b) Typological design commonalities (e.g., school design) ;
6. Learning from earlier projects by the same office:
    (a) Project planning helper ;
    (b) Ecological impact estimation ;

Whether these cases can be actually further developed depends, of course, on multiple factors, which can only be marginally influenced by the ecosystem, independent from their TRL. A first factor is privacy. Depending on who performs the action, data may or may not be available. For example, relating asset data and personal data will be a sensitive topic. On the other hand, these cases only become possible in a privacy-sensitive way *because* the Solid protocol, on which ConSolid relies, allows fine-grained access control by the owners of data. A user can give explicit consent to use specific information snippets.

The implications of this privacy-enabled data sharing and further technical challenges are an active field of research within the Solid Community. A second factor is interoperability between datasets, which will depend on their respective formats. Due to the principle of heterogeneity, on an infrastructural level ConSolid cannot really influence how ecosystem-internal data is organised, let alone external datasets. However, the benefits of highly structured (5 star) data were emphasised numerous times throughout the dissertation. With the FAIR principles becoming more widely adopted, as well as the rising usage of Semantic Web technologies for documenting public datasets, we can assume that more and more of the above-mentioned valorisation cases will come within reach.

The valorisation potential of the Mifesto framework is more difficult to estimate. The generic applicability of the framework will allow continuous development of new modules, which gives rise to novel combinations of media types and ontologies, thereby supporting novel cases of graphically supported 'many-to-many' enrichment. Therefore, the valorisation scenarios that were listed from the *data storage perspective* of ConSolid can be considered equally relevant from the *user interaction perspective* of Mifesto. After all, Mifesto just provides the means to graphically interact with multi-models. Therefore, any real valorisation scenario of ConSolid *can* always make use of Mifesto, although this is not a necessity when the project's default media types are set, e.g. using (inter)national standards (cf. 'Little Open BIM' [134]). Thus, many practical implementation scenarios can just be based on common development practices for GUIs. Micro-frontends can be thought of as an extension of these practices.

In this sense, neither Mifesto and ConSolid are 'all-or-nothing' solutions: they can be gradually adopted and extended when the need rises – facilitating tailor-made data management on the World Wide Web.

References

[1] Aec Systems Ltd. *Speckle: The Platform For 3D Data*. 2021. URL: https://speckle.guide/ (accessed 2023-10-24).

[2] Rajat Agarwal, Shankar Chandrasekaran, and Mukund Sridhar. *Imagining construction's digital future*. Tech. rep. 2016. URL: https://www.mckinsey.com/capabilities/operations/our-insights/imagining-constructions-digital-future (accessed 2023-11-30).

[3] Arghavan Akbarieh, FN Teferle, and J O'Donnell. "Semantic Material Bank: A web-based linked data approach for building decommissioning and material reuse". In: *ECPPM 2022-eWork and eBusiness in Architecture, Engineering and Construction 2022*. CRC Press, 2023, pp. 69–76. ISBN: 9781003354222. URL: https://doi.org/10.1201/9781003354222-9.

[4] Riccardo Albertoni, David Browning, Simon Cox, Alejandra Gonzales Beltran, Andrea Perego, and Peter Winstanley. *Data Catalog Vocabulary (DCAT) - Version 3*. W3C Recommendation. W3C, 2023. URL: https://www.w3.org/TR/vocab-dcat-3 (accessed 2023-11-30).

[5] Autodesk Inc. *Autodesk Platform Services*. 2023. URL: https://aps.autodesk.com/ (accessed 2023-11-30).

[6] M.J. Barbosa, P. Pauwels, V. Ferreira, and L. Mateus. "Towards Increased BIM Usage for Existing Building Interventions". In: *Structural Survey* 34 (2016), pp. 168–190. URL: https://doi.org/10.1108/SS-01-2015-0002.

[7] David Becket, Tim Berners-Lee, Prud'hommeaux Erix, and Gavin Carothers. *RDF 1.1 Turtle - Terse RDF Triple Language*. W3C Recommendation. W3C, 2014. URL: https://www.w3.org/TR/turtle/ (accessed 2023-11-30).

[8] J. Beetz, J.P. Leeuwen, van, and B. Vries, de. "An Ontology Web Language Notation of the Industry Foundation Classes". In: *Proceedings of the 22nd CIB W78 Conference on Information Technology in Construction*. Technische Universität Dresden, 2005, pp. 193–198. URL: https://alexandria.tue.nl/openaccess/Metis209539.pdf (accessed 2023-11-30).

[9] Jakob Beetz. "Facilitating distributed collaboration in the AEC/FM sector using Semantic Web Technologies". PhD thesis. TU Eindhoven, 2009. URL: https://pure.tue.nl/ws/files/2966330/200911977.pdf (accessed 2023-11-30).

[10] Jakob Beetz, Léon van Berlo, Ruben de Laat, and Pim van den Helm. "BIMserver.org–An open source IFC model server". In: *Proceedings of the 27th CIB W78 conference on Information Technology in Construction*. Cairo, Egypt, 2010. URL: https://d1wqtxts1xzle7.cloudfront.net/5351671/beetz_berlo-cib-w78_cairo-libre.pdf (accessed 2023-11-30).

[11] Jakob Beetz, Jos Van Leeuwen, and Bauke De Vries. "IfcOWL: A case of transforming EXPRESS schemas into ontologies". In: *Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AI EDAM* 23 (2009), p. 89. URL: https://doi.org/10.1017/S0890060409000122.

[12] Tim Berners-Lee. *Cool URIs don't change*. W3C Style Guide. W3C, 1998. URL: https://www.w3.org/Provider/Style/URI (accessed 2024-2-12).

[13] Tim Berners-Lee. *Design Issues: Linked Data*. 2006. URL: https://www.w3.org/DesignIssues/LinkedData.html (accessed 2022-11-17).

[14] Tim Berners-Lee, James Hendler, Ora Lassila, et al. "The Semantic Web". In: *Scientific American* 284 (2001), pp. 28–37. URL: https://doi.org/10.1038/scientificamerican0501-34.

[15] Mark Bew and Richards M. "Bew-Richards BIM maturity model". In: *Construct IT Autumn Members Meeting*. Brighton, UK, 2008.

[16] Bart Bogaerts, Bas Ketsman, Younes Zeboudj, Heba Aamer, Ruben Taelman, and Ruben Verborgh. "Link Traversal with Distributed Subweb Specifications". In: *Proceedings of the 5th International Joint Conference on Rules and Reasoning*. Vol. 12851. Lecture Notes in Computer Science. Springer, 2021, pp. 62–79. URL: https://doi.org/10.1007/978-3-030-91167-6_5.

[17] Calin Boje, Annie Guerriero, Sylvain Kubicki, and Yacine Rezgui. "Towards a semantic Construction Digital Twin: Directions for future research". In: *Automation in construction* 114 (2020), p. 103179. URL: https://doi.org/10.1016/j.autcon.2020.103179.

[18] Mathias Bonduel. *Construction Dataset Context Ontology*. 2020. URL: https://mathib.github.io/cdc-ontology/ (accessed 2023-11-30).

[19] Mathias Bonduel. "A Framework for a Linked Data-based Heritage BIM". eng. PhD thesis. KU Leuven, 2021. URL: https://lirias.kuleuven.be/retrieve/618662 (accessed 2023-11-30).

[20] Mathias Bonduel, Jyrki Oraskari, Pieter Pauwels, Maarten Vergauwen, and Ralf Klein. “The IFC to Linked Building Data Converter - Current Status”. In: *Proceedings of the 6th Linked Data in Architecture and Construction Workshop (LDAC)*. Vol. 2159. London, United Kingdom, 2018, pp. 34–43. URL: http://ceur-ws.org/Vol-2159/04paper.pdf (accessed 2023-11-30).

[21] Mathias Bonduel, Anna Wagner, Pieter Pauwels, Maarten Vergauwen, and Ralf Klein. “Including Widespread Geometry Formats in Semantic Graphs using RDF Literals”. In: *Proceedings of the European Conference on Computing in Construction (EC3 2019)*. Chania, Greece, 2019, pp. 341–350. URL: https://doi.org/10.35490/EC3.2019.166.

[22] André Borrmann, Markus König, Christian Koch, and Jakob Beetz. “Building Information Modeling: Why? What? How?” In: *Building Information Modeling: Technology Foundations and Industry Practice*. Springer International Publishing, 2018, pp. 1–24. ISBN: 978-3-319-92862-3.

[23] Matthieu Bosquet. *Access Control Policy*. Protocol. W3C Solid Community Group, 2022. URL: https://solid.github.io/authorization-panel/acp-specification/ (accessed 2023-11-30).

[24] Pierre Bourreau, Nathalie Charbel, Jeroen Werbrouck, Madhumitha Senthilvel, Pieter Pauwels, and Jakob Beetz. “Multiple inheritance for a modular BIM”. In: *Le BIM et l’évolution des pratiques: Ingénierie et architecture, enseignement et recherche* (2020), pp. 63–82. URL: https://community.osarch.org/uploads/editor/a0/1se97k6z8n3v.pdf (accessed 2024-3-18).

[25] Box, Inc. *The Information Economy: A Study of Five Industries*. Tech. rep. Box, Inc., 2014. URL: https://img.forconstructionpros.com/files/base/acbm/fcp/document/2014/06/box-cloud-study_11535206.pdf (accessed 2023-11-30).

[26] Dan Brickley and R. V. Guha. *RDF Schema 1.1*. W3C Recommendation. W3C, 2014. URL: https://www.w3.org/TR/rdf-schema/ (accessed 2023-11-30).

[27] buildingSMART International. *BCF API*. 2022. URL: https://github.com/buildingSMART/BCF-API (accessed 2023-5-17).

[28] buildingSMART International. *BCF API - topic GET.json*. 2022. URL: https://raw.githubusercontent.com/buildingSMART/BCF-API/release_3_0/Schemas/Collaboration/Topic/topic_GET.json (accessed 2023-11-30).

[29] buildingSMART International. *BIM Collaboration Format (BCF)*. 2022. URL: https://github.com/BuildingSMART/BCF-XML/tree/release_3_0/Documentation (accessed 2022-10-27).

[30] buildingSMART International. *Documents API*. 2022. URL: https://github.com/buildingSMART/documents-API (accessed 2023-11-30).

[31] buildingSMART International Ltd. *Technical Roadmap buildingSMART - Getting ready for the future*. Tech. rep. buildingSMART International Ltd., 2020. URL: https://f3h3w7a5.rocketcdn.me/wp-content/uploads/2020/09/20200430_buildingSMART_Technical_Roadmap.pdf (accessed 2023-11-30).

[32] Sarven Capadisli and Tim Berners-Lee. *Web Access Control*. Protocol. W3C Solid Community Group, 2022. URL: https://solid.github.io/web-access-control-spec/ (accessed 2023-11-30).

[33] Sarven Capadisli, Tim Berners-Lee, Ruben Verborgh, and Kjetil Kjernsmo. *Solid Protocol*. Protocol. W3C Solid Community Group, 2022. URL: https://solidproject.org/TR/protocol (accessed 2023-11-30).

[34] Sarven Capadisli and Amy Guy. *Linked Data Platform 1.0*. W3C Recommendation. W3C, 2017. URL: https://www.w3.org/TR/ldn/ (accessed 2023-11-30).

[35] Bo Carlsson. "The Digital Economy: what is new and what is not?" In: *Structural change and economic dynamics* 15.3 (2004), pp. 245–264. URL: https://doi.org/10.1016/j.strueco.2004.02.001.

[36] David Churcher, Sarah Davidson, and Anne Kemp. *UK BIM Framework Guidance*. 2022. URL: https://ukbimframeworkguidance.notion.site/UK-BIM-Framework-Guidance-20a045d01cfb42fea2fef35a7b988dbc (accessed 2023-11-24).

[37] Aaron Coburn, Elf Pavlik, and Dmitri Zagidulin. *Solid-OIDC*. Specification. Solid, 2022. URL: https://solidproject.org/TR/oidc (accessed 2023-11-30).

[38] Cocycles Ltd. *Bit - Build anything in components*. 2019. URL: https://bit.dev/ (accessed 2023-1-11).

[39] Sandra Collins, Francoise Genova, Natalie Harrower, Simon Hodson, Sarah Jones, Leif Laaksonen, Daniel Mietchen, Rūta Petrauskaitė, and Peter Wittenburg. "Turning FAIR into reality: Final report and action plan from the European Commission expert group on FAIR data". In: (2018). URL: https://data.europa.eu/doi/10.2777/1524.

[40] Pieter Colpaert. “Publishing transport data for maximum reuse”. PhD thesis. Ghent University, 2017. URL: https://phd.pietercolpaert.be (accessed 2023-11-30).

[41] Aaron Costin, Jeffrey W. Ouellette, and Jakob Beetz. “Building product models, terminologies, and object type libraries”. In: *Buildings and Semantics*. CRC Press, 2022, pp. 3–24.

[42] Laura Daniele, Raul Garcia-Castro, Maxime Lefrançois, and Maria Poveda-Villalon. *Smart Applications REFerence Ontology*. 2020. URL: https://saref.etsi.org/core (accessed 2023-11-30).

[43] Laura Daniele, Frank den Hartog, and Jasper Roes. “Created in close interaction with the industry: the smart appliances reference (SAREF) ontology”. In: *International Workshop Formal Ontologies Meet Industries*. Springer, 2015, pp. 100–112. URL: https://link.springer.com/chapter/10.1007/978-3-319-21545-7_9 (accessed 2023-11-30).

[44] Ruben Dedecker, Wout Slabbinck, Jesse Wright, Patrick Hochstenbach, Pieter Colpaert, and Ruben 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. URL: https://solidlabresearch.github.io/WhatsInAPod/ (accessed 2023-11-30).

[45] Makx Dekkers, Pavlina Fragkou, Natasa Sofou, and Bert Van Nuffelen. *DCAT-AP 3.0*. EU standard. SEMIC EU, 2023. URL: https://semiceu.github.io/DCAT-AP/releases/3.0.0/ (accessed 2023-11-30).

[46] CC Developers. *Creative Commons Rights Expression Language*. 2006. URL: https://creativecommons.org/ns# (accessed 2024-3-21).

[47] Digitaal Vlaanderen. *DCAT-AP Vlaanderen profiel en validator*. 2022. URL: https://www.vlaanderen.be/digitaal-vlaanderen/onze-oplossingen/open-data/dcat-ap-vlaanderen-profiel-en-validator (accessed 2023-8-22).

[48] *DIN SPEC 91391-1:2019 Common Data Environments (CDE) for BIM projects - Function sets and open data exchange between platforms of different vendors - Part 1: Components and function sets of a CDE*. DIN SPEC. Deutsches Institut für Normung (DIN), 2019.

[49] *DIN SPEC 91391-2:2019 Common Data Environments (CDE) for BIM projects –Function sets and open data exchange between platforms Part 2: Open data exchange with Common Data Environments*. DIN SPEC. Deutsches Institut für Normung (DIN), 2019.

[50] Manish K Dixit, Varusha Venkatraj, Mohammadreza Ostadalimakhmalbaf, Fatemeh Pariafsai, and Sarel Lavy. “Integration of facility management and building information modeling (BIM)”. In: *Facilities* Vol. 37 (2019), pp. 455–483. URL: https://doi.org/10.1108/F-03-2018-0043.

[51] Charles Eastman. “The use of computers instead of drawings in building design”. In: *AIA journal* 63 (1975), pp. 46–50. URL: https://www.researchgate.net/profile/Charles-Eastman/publication/234643558_The_Use_of_Computers_Instead_of_Drawings_in_Building_Design/links/54aff5690cf2431d3531c7a7/The-Use-of-Computers-Instead-of-Drawings-in-Building-Design.pdf (accessed 2023-11-30).

[52] Chuck Eastman, Paul Teicholz, Rafael Sacks, and Kathleen Liston. *BIM handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers and Contractors*. 2nd. Hoboken, New York, United States: John Wiley & Sons, 2011, p. 650. ISBN: 978-0-470-54137-1.

[53] The Economist. *Break down these walls*. 2008. URL: https://www.economist.com/leaders/2008/03/19/break-down-these-walls (accessed 2023-11-30).

[54] Mohamed Elagiry, Valentina Marino, Natalia Lasarte, Peru Elguezabal, and Thomas Messervey. “BIM4Ren: Barriers to BIM implementation in renovation processes in the Italian market”. In: *Buildings* 9 (2019), pp. 200–217. URL: https://doi.org/10.3390/buildings9090200.

[55] Nuyts Emma, Jeroen Werbrouck, Ruben Verstraeten, and Louise Deprez. “Validation of Building Models against Legislation using SHACL”. In: *LDAC2023, the 11th Linked Data in Architecture and Construction Workshop*. 2023. URL: https://linkedbuildingdata.net/ldac2023/files/papers/papers/LDAC2023_paper_8284.pdf (accessed 2024-3-18).

[56] European Commission - Joinup. *DCAT-AP SHACL shapes*. 2019. URL: https://github.com/SEMICeu/dcat-ap_shacl (accessed 2023-10-13).

[57] Stephen Eyre. *Mott Macdonald Ltd v Trant Engineering Ltd [2021] EWHC 754 (TCC)*. Mar. 30, 2021. URL: https://www.bailii.org/ew/cases/EWHC/TCC/2021/754.html (accessed 2022-11-16).

[58] Getty Research Institute. *Art & Architecture Thesaurus*. 2014. URL: http://vocab.getty.edu/aat/ (accessed 2023-11-30).

[59] Ghent University – imec. *comunica-feature-link-traversal*. 2021. URL: https://github.com/comunica/comunica-feature-link-traversal (accessed 2023-10-30).

[60] GO FAIR. *FAIR principles*. 2016. URL: https://www.go-fair.org/fair-principles/ (accessed 2023-11-30).

[61] Google. *Angular - Deliver Web Apps with Confidence*. 2015. URL: https://angular.io/ (accessed 2023-1-11).

[62] Khronos 3D Formats Working Group. *glTF 2.0 Specification*. 2021. URL: https://registry.khronos.org/glTF/specs/2.0/glTF-2.0.html (accessed 2023-11-30).

[63] Mario Gürtler, Ken Baumgärtel, and Raimar J Scherer. "Towards a workflow-driven multi-model bim collaboration platform". In: *Working Conference on Virtual Enterprises*. Springer, 2015, pp. 235–242. URL: https://doi.org/10.1007/978-3-319-24141-8_21.

[64] Philipp Hagedorn, Liu Liu, Markus König, Rade Hajdin, Tim Blumenfeld, Markus Stöckner, Maximilian Billmaier, Karl Grossauer, and Kenneth Gavin. "BIM-Enabled Infrastructure Asset Management Using Information Containers and Semantic Web". In: *Journal of Computing in Civil Engineering* 37.1 (2023), p. 04022041. URL: https://doi.org/10.1061/(ASCE)CP.1943-5487.0001051.

[65] Philipp Hagedorn, Madhumitha Senthilvel, Hans Schevers, and Lucas Verhelst. "Towards usable ICDD containers for ontology-driven data linking and link validation". In: *Proceedings of the 11th Linked Data in Architecture and Construction Workshop (LDAC)*. Matera, Italy, 2023. URL: https://linkedbuildingdata.net/ldac2023/files/papers/papers/LDAC2023_paper_2079.pdf (accessed 2023-11-30).

[66] Armin Haller, Krzysztof Janowicz, Simon JD Cox, Maxime Lefrançois, Kerry Taylor, Danh Le Phuoc, Joshua Lieberman, Raúl Garcıéa-Castro, Rob Atkinson, and Claus Stadler. "The modular SSN ontology: A joint W3C and OGC standard specifying the semantics of sensors, observations, sampling, and actuation". In: *Semantic Web* 10 (2019), pp. 9–32. URL: https://hal.science/hal-01885335/document (accessed 2023-11-30).

[67] Al-Hakam Hamdan and Mathias Bonduel. *Damage Topology Ontology*. 2019. URL: https://w3id.org/dot# (accessed 2023-11-30).

[68] Al-Hakam Hamdan, Mathias Bonduel, and Raimar J. Scherer. "An ontological model for the representation of damage to constructions". In: *Proceedings of the 7th Linked Data in Architecture and Construction Workshop (LDAC)*. Lisbon, Portugal, 2019, pp. 64–77. URL: http://ceur-ws.org/Vol-2389/05paper.pdf (accessed 2019-12-5).

[69] Dick Hardt. *The OAuth 2.0 authorization framework*. Tech. rep. 2012. URL: https://www.rfc-editor.org/rfc/rfc6749 (accessed 2023-11-30).

[70] Steve Harris, Andy Seaborne, and Eric Prudhéommeaux. *SPARQL 1.1 Query Language*. W3C Recommendation. W3C, 2013. URL: https://www.w3.org/TR/sparql11-query/ (accessed 2023-11-30).

[71] Olaf Hartig. "An overview on execution strategies for Linked Data queries". In: *Datenbank-Spektrum* 13 (2013), pp. 89–99. URL: https://link.springer.com/article/10.1007/s13222-013-0122-1.

[72] Olaf Hartig. "SQUIN: a traversal based query execution system for the web of linked data". In: *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*. 2013, pp. 1081–1084. URL: https://dl.acm.org/doi/pdf/10.1145/2463676.2465231 (accessed 2023-11-30).

[73] James Hendler, Fabien Gandon, and Dean Allemang. *Semantic Web for the Working Ontologist: Effective Modeling for Linked Data, RDFS, and OWL*. Morgan & Claypool, 2020. ISBN: 978-1-4503-7617-4.

[74] N. Vu Hoang and Seppo Törmä. "Drumbeat platform–a web of building data implementation with backlinking". In: *eWork and eBusiness in Architecture, Engineering and Construction*. CRC Press, 2017, pp. 155–163.

[75] Tim-Jonathan Huyeng, Christian-Dominik Thiele, Anna Wagner, Meiling Shi, André Hoffmann, Wendelin Sprenger, and Uwe Rüppel. "An approach to process geometric and semantic information as open graph-based description using a microservice architecture on the example of structural data". In: *Proceedings of the EG-ICE 2020 Workshop on Intelligent Computing in Engineering*. Berlin, Germany: Universitätsverlag der TU Berlin, 2020. (accessed 2023-11-30).

[76] iana.org. *Internet Assigned Numbers Authority*. 1997. URL: https://www.iana.org/ (accessed 2023-11-3).

[77] InfluxData, Inc. *Get started with InfluxDB v2*. 2023. URL: https://docs.influxdata.com/influxdb/v2/ (accessed 2023-10-20).

[78] Inc. Inrupt. *Enterprise Solid Server (ESS)*. 2020. URL: https://www.inrupt.com/products/enterprise-solid-server (accessed 2023-2-23).

[79] buildingSMART International. *OpenCDE API standards*. 2018. URL: https://github.com/buildingSMART/OpenCDE-API (accessed 2023-4-12).

[80] Dama International. *DAMA-DMBOK: data management body of knowledge*. Technics Publications, LLC, 2017.

[81] International Organization for Standardization (ISO). *The ISO 21597 ICDD Part 1 Container ontology*. 2020. URL: https://standards.iso.org/iso/21597/-1/ed-1/en/Container.rdf (accessed 2023-11-30).

[82] International Organization for Standardization (ISO). *The ISO 21597 ICDD Part 1 Linkset ontology*. 2020. URL: https://standards.iso.org/iso/21597/-1/ed-1/en/Linkset.rdf (accessed 2023-11-30).

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

[84] *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*. ISO Standard. International Organization for Standardization (ISO), 2018.

[85] *ISO 19650-2:2018 Organization and digitization of information about buildings and civil engineering works, including building information modelling (BIM) – Information management using building information modelling – Part 2: Delivery phase of the assets*. ISO Standard. International Organization for Standardization (ISO), 2018.

[86] *ISO 19650-3:2020 Organization and digitization of information about buildings and civil engineering works, including building information modelling (BIM) - Information management using building information modelling - Part 3: Operational phase of the assets*. ISO Standard. International Organization for Standardization (ISO), 2020.

[87] *ISO 19650-4:2022 Organization and digitization of information about buildings and civil engineering works, including building information modelling (BIM) - Information management using building information modelling - Part 4: Information exchange*. ISO Standard. International Organization for Standardization (ISO), 2022.

[88] *ISO 19650-5:2020 Organization and digitization of information about buildings and civil engineering works, including building information modelling (BIM) - Information management using building information modelling - Part 5: Security-minded approach to information management*. ISO Standard. International Organization for Standardization (ISO), 2020.

[89] *Information container for linked document delivery - Exchange specification - Part 1: Container (ISO 21597-1:2020)*. ISO Standard. International Organization for Standardization (ISO), 2020.

[90] *Information container for linked document delivery - Exchange specification - Part 2: Link types (ISO 21597-2:2020)*. ISO Standard. International Organization for Standardization (ISO), 2020.

[91] Laurent Joblot, Thomas Paviot, Dominique Deneux, and Samir Lamouri. "Literature review of Building Information Modeling (BIM) intended for the purpose of renovation projects". In: *IFAC-PapersOnLine* 50 (2017), pp. 10518–10525. URL: https://doi.org/10.1016/j.ifacol.2017.08.1298.

[92] M. Jones, J. Bradley, and N. Sakimura. *JSON Web Token (JWT)*. 2015. URL: https://doi.org/10.17487/rfc7519.

[93] Michael B. Jones, Anthony Nadalin, Brian Campbell, John Bradley, and Chuck Mortimore. *OAuth 2.0 Token Exchange*. 2020. URL: https://doi.org/10.17487/RFC8693.

[94] Ebrahim P Karan, Javier Irizarry, and John Haymaker. "BIM and GIS integration and interoperability based on semantic web technology". In: *Journal of Computing in Civil Engineering* 30 (2016). URL: https://doi.org/10.1061/(ASCE)CP.1943-5487.0000519.

[95] Janakiram Karlapudi, Prathap Valluru, and Karsten Menzel. "An explanatory use case for the implementation of Information Container for linked Document Delivery in Common Data Environments". In: *Proceedings of the EG-ICE 2021 Workshop on Intelligent Computing in Engineering*. Berlin, Germany: Universitätsverlag der TU Berlin, 2021, pp. 76–86.

[96] Sabrina Kirrane, Alessandra Mileo, and Stefan Decker. "Access control and the resource description framework: A survey". In: *Semantic Web* 8.2 (2017), pp. 311–352. URL: https://doi.org/10.3233/SW-160236.

[97] M. Kirschstein, L. Liu, and L Höltgen. *ICDD Platform Documentation*. 2022. URL: https://icdd.vm.rub.de/ui/Page/Documentation (accessed 2023-11-30).

[98] Fabian Kirstein, Benjamin Dittwald, Simon Dutkowski, Yury Glikman, Sonja Schimmler, and Manfred Hauswirth. "Linked data in the european data portal: A comprehensive platform for applying dcat-ap". In: *Electronic Government: 18th IFIP WG 8.5 International Conference, EGOV 2019*. San Benedette Del Tronto, Italy: Springer, 2019, pp. 192–204. URL: https://doi.org/10.1007/978-3-030-27325-5_15.

[99] Pavel Klinov. *FROM vs FROM NAMED in SPARQL*. 2021. URL: https://www.stardog.com/labs/blog/from-vs-from-named-in-sparql/ (accessed 2023-11-30).

[100] Graham Klyne, Jeremy J. Carroll, and Brian McBride. *RDF 1.1 Concepts and Abstract Syntax*. W3C Recommendation. W3C, 2014. URL: https://www.w3.org/TR/rdf11-concepts/ (accessed 2023-11-30).

[101] Dimitris Kontokostas and Holger Knublauch. *Shapes Constraint Language (SHACL)*. W3C Recommendation. W3C, 2017. URL: https://www.w3.org/TR/shacl/ (accessed 2023-11-30).

[102] Tobias Kuhn, Paolo Emilio Barbano, Mate Levente Nagy, and Michael Krauthammer. "Broadening the scope of nanopublications". In: *Extended Semantic Web Conference*. Springer, 2013, pp. 487–501. URL: https://arxiv.org/pdf/1303.2446.pdf (accessed 2023-11-30).

[103] Tobias Kuhn and Michel Dumontier. "Trusty URIs: Verifiable, immutable, and permanent digital artifacts for linked data". In: *The Semantic Web: Trends and Challenges: 11th International Conference, ESWC 2014, Anissaras, Crete, Greece, May 25-29, 2014. Proceedings 11*. Springer, 2014, pp. 395–410. URL: https://arxiv.org/pdf/1401.5775.pdf (accessed 2023-11-30).

[104] Birgit van Laar. "Vernieuwde inspectieverslagen van Monumentenwacht helpen eigenaars om hun gebouwd erfgoed beter te onderhouden en te beheren". In: *Preventieve Conservatie van Klimaat- en Schademonitoring naar een Geïntegreerde Systeembenadering*. Vol. 35. Leuven, Belgium: WTA-NL-VL, 2019, pp. 1–9. URL: https://www.wta-international.org/fileadmin/user_upload/Nederland-Vlaanderen/syllabi/2019-04-05_Preventieve_Conservatie.pdf (accessed 2023-11-30).

[105] Jose Emilio Labra Gayo, Eric Prud'hommeaux, Iovka Boneva, and Dimitris Kontokostas. *Validating RDF Data*. Vol. 7. 1. Morgan & Claypool Publishers LLC, 2017, pp. 1–328. ISBN: 978-3-031-79478-0. URL: https://doi.org/10.2200/s00786ed1v01y201707wbe016.

[106] Mary Lebens, Roger J Finnegan, Steven C Sorsen, and Jinal Shah. "Rise of the Citizen Developer". In: *Muma Business Review* 5 (2022), pp. 101–111. URL: https://doi.org/10.28945/4885.

[107] Timothy Lebo, Satya Sahoo, Deborah McGuinness, Khalid Belhajjame, James Cheney, David Corsar, Daniel Garijo, Stian Soiland-Reyes, Stephan Zednik, and Jun Zhao. *Prov-o: The prov ontology*. 2013. URL: https://www.w3.org/TR/prov-o/ (accessed 2024-3-21).

[108] Zhiliang Ma and Yuan Ren. "Integrated application of BIM and GIS: an overview". In: *Procedia Engineering* 196 (2017), pp. 1072–1079. URL: https://doi.org/10.1016/j.proeng.2017.08.064.

[109] Andrew Malcolm, Jeroen Werbrouck, and Pieter Pauwels. "LBD server: Visualising Building Graphs in web-based environments using semantic graphs and glTF-models". In: *Formal Methods in Architecture: Proceedings of the 5th International Symposium on Formal Methods in Architecture (5FMA), Lisbon 2020*. Springer. 2021. URL: https://doi.org/10.1007/978-3-030-57509-0_26 (accessed 2024-3-18).

[110] John C Mankins et al. "Technology readiness levels". In: *White Paper, April* 6 (1995). URL: https://aiaa.kavi.com/apps/group_public/download.php/2212/TRLs_MankinsPaper_1995.pdf (accessed 2023-11-30).

[111] Essam Mansour, Andrei Vlad Sambra, Sandro Hawke, Maged Zereba, Sarven Capadisli, Abdurrahman Ghanem, Ashraf Aboulnaga, and Tim Berners-Lee. "A demonstration of the solid platform for social web applications". In: *Proceedings of the 25th International Conference Companion on World Wide Web*. 2016, pp. 223–226. URL: https://doi.org/10.1145/2872518.2890529.

[112] Carol C Menassa. "From BIM to digital twins: A systematic review of the evolution of intelligent building representations in the AEC-FM industry". In: *Journal of Information Technology in Construction (ITcon)* 26.5 (2021), pp. 58–83. URL: https://doi.org/10.36680/j.itcon.2021.005.

[113] Meta Platforms. *React - A JavaScript library for building user interfaces*. 2013. URL: https://reactjs.org/ (accessed 2023-1-11).

[114] Microsoft. *What is Power BI?* 2015. URL: https://powerbi.microsoft.com/en-au/what-is-power-bi/ (accessed 2023-10-29).

[115] Michael Mikowski and Josh Powell. *Single page web applications: JavaScript end-to-end*. Simon and Schuster, 2013. ISBN: 9781617290756.

[116] Barend Mons, Cameron Neylon, Jan Velterop, Michel Dumontier, Luiz Olavo Bonino da Silva Santos, and Mark D Wilkinson. "Cloudy, increasingly FAIR; revisiting the FAIR Data guiding principles for the European Open Science Cloud". In: *Information services & use* 37.1 (2017), pp. 49–56. URL: http://doi.org/10.3233/ISU-170824.

[117] Lina Morkunaite, Fayes Haitham Al-Naber, Ekaterina Petrova, and Kjeld Svidt. "An Open Data Platform for Early-Stage Building Circularity Assessment". In: *Proceedings of the 38th CIB W78 Conference on Information Technology in Construction*. Luxembourg: University of Ljubljana, 2021, pp. 813–822. URL: https://vbn.aau.dk/ws/files/466232044/An_Open_Data_Platform_for_Early_Stage_Building_Circularity_Assessment.pdf.

[118] van Nederveen. "Building Information Modelling in the Netherlands: A Status Report". In: *18th CIB World Building Congress*. Vol. 361. Salford Quays, United Kingdom: CIB, 2010, p. 13. URL: https://site.cibworld.nl/dl/publications/w078_pub361.pdf (accessed 2023-11-30).

[119] Luiz Olavo Bonino, Kees Burger, and Rajaram Kaliyaperumal. *FAIR Data Point - working draft*. 2022. URL: https://specs.fairdatapoint.org/ (accessed 2023-11-30).

[120] Marten Oltrogge, Erik Derr, Christian Stransky, Yasemin Acar, Sascha Fahl, Christian Rossow, Giancarlo Pellegrino, Sven Bugiel, and Michael Backes. "The rise of the citizen developer: Assessing the security impact of online app generators". In: *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2018, pp. 634–647. URL: https://doi.org/10.1109/SP.2018.00005 (accessed 2023-11-30).

[121] Femke Ongenae and Pieter Bonte. *Aggregators to realize Scalable Querying across Decentralized Solid pods*. 2023. URL: https://knows.idlab.ugent.be/education/master-thesis/2023/aggregators-to-realize-scalable-querying-across-decentralized-solid-pods/ (accessed 2023-11-30).

[122] Open Knowledge Foundation. *The Open Definition*. 2007. URL: https://opendefinition.org/ (accessed 2023-1-11).

[123] OpenSource BIM. *BIMserver Javascript API*. 2020. URL: https://github.com/opensourceBIM/BIMserver-JavaScript-API (accessed 2023-11-30).

[124] J Oraskari and S Törmä. "Access control for web of building data: Challenges and directions". In: *eWork and eBusiness in Architecture, Engineering and Construction* (2017), pp. 45–53.

[125] Jyrki Oraskari, Oliver Schulz, and Jakob Beetz. "Towards describing version history of BCF data in the Semantic Web". In: *Proceedings of the 10th Linked Data in Architecture and Construction Workshop (LDAC)*. Hersonissos, Greece, 2022.

[126] Jyrki Oraskari, Oliver Schulz, Jeroen Werbrouck, and Jakob Beetz. "Enabling Interoperable Issue Management in a Federated Building and Construction Sector". In: *EG-ICE 2022 Workshop on Intelligent Computing in Engineering*. 2022. URL: https://api.semanticscholar.org/CorpusID:250108125 (accessed 2024-3-18).

[127] Daniela Pasini. "Connecting BIM and IoT for addressing user awareness toward energy savings". In: *Journal of Structural Integrity and Maintenance* 3.4 (2018), pp. 243–253.

[128] P. Pauwels and W. Terkaj. "EXPRESS to OWL for Construction Industry: Towards a Recommendable and Usable ifcOWL Ontology". In: *Automation in Construction* 63 (2016), pp. 100–133. URL: https://doi.org/10.1016/j.autcon.2015.12.003.

[129] P. Pauwels, S. Zhang, and Y.-C. Lee. "Semantic Web Technologies in AEC industry: A Literature Overview". In: *Automation in Construction* 73 (2017), pp. 145–165. URL: https://doi.org/10.1016/j.autcon.2016.10.003.

[130] Pieter Pauwels. *BuildingElement Ontology*. 2018. URL: https://pi.pauwel.be/voc/buildingelement/index-en.html (accessed 2023-11-30).

[131] Pieter Pauwels. *DistributionElement Ontology*. 2018. URL: https://pi.pauwel.be/voc/distributionelement/index-en.html (accessed 2023-11-30).

[132] Pieter Pauwels, Thomas Krijnen, Walter Terkaj, and Jakob Beetz. "Enhancing the ifcOWL ontology with an alternative representation for geometric data". In: *Automation in Construction* 80 (2017), pp. 77–94. URL: https://doi.org/10.1016/j.autcon.2017.03.001.

[133] Pieter Pauwels and Kris McGlinn. *Buildings and Semantics: Data Models and Web Technologies for the Built Environment*. CRC Press, 2022. ISBN: 9781032023120.

[134] Pieter Pauwels and Ekaterina Petrova. *Information in Construction*. 2018. URL: https://vbn.aau.dk/ws/files/525394301/InformationInConstruction_PauwelsPetrova.pdf (accessed 2023-11-30).

[135] Severi Peltonen, Luca Mezzalira, and Davide Taibi. "Motivations, benefits, and issues for adopting Micro-Frontends: A Multivocal Literature Review". In: *Information and Software Technology* 136 (2021), p. 106571. URL: https://doi.org/10.1016/j.infsof.2021.106571.

[136] M Perry and J Herring. *GeoSPARQL-A Geographic Query Language for RDF Data OGC*. 2011. URL: https://www.opengeospatial.org/standards/geosparql (accessed 2019-12-4).

[137] David Philp, David Churcher, and Sarah Davidson. “Government Soft Landings”. In: (2019). URL: https://doi.org/10.17863/CAM.45315.

[138] Pardis Pishdad-Bozorgi, Xinghua Gao, Charles Eastman, and Alonzo Patrick Self. “Planning and developing facility management-enabled building information model (FM-enabled BIM)”. In: *Automation in Construction* 87 (2018), pp. 22–38. URL: https://doi.org/10.1016/j.autcon.2017.12.004.

[139] Maria Poveda-Villalón and Raúl Garcia-Castro. “Extending the SAREF ontology for building devices and topology”. In: *Proceedings of the 6th Linked Data in Architecture and Construction Workshop (LDAC 2018)*. Vol. 2159. 2018, pp. 16–23. URL: https://ceur-ws.org/Vol-2159/02paper.pdf (accessed 2023-11-30).

[140] C Preidel, A Borrmann, C Oberender, and M Tretheway. “Seamless integration of common data environment access into BIM authoring applications: The BIM integration framework”. In: *eWork and eBusiness in Architecture, Engineering and Construction*. CRC Press, 2017, pp. 119–128. URL: https://mediatum.ub.tum.de/doc/1306961/654ocv4bit32uh73pjj1z1vjs.pdf (accessed 2023-11-30).

[141] Cornelius Preidel, André Borrmann, Hannah Mattern, Markus König, and Sven-Eric Schapke. “Common Data Environment”. In: Springer, 2018, pp. 279–291. ISBN: 978-3-319-92862-3.

[142] Niculin Prinz, Christopher Rentrop, and Melanie Huber. “Low-Code Development Platforms-A Literature Review.” In: *AMCIS*. 2021. URL: https://web.archive.org/web/20220801212143id_/https://aisel.aisnet.org/cgi/viewcontent.cgi?article=1079&context=amcis2021 (accessed 2023-11-30).

[143] Eric Prud’hommeaux, Iovka Boneva, Jose Emilio Labra Gayo, and Greg Kellogg. *Shape Expressions Language 2.1*. Report. Shape Expressions Community Group, 2019. URL: https://shex.io/shex-semantics/index.html (accessed 2023-11-30).

[144] Mads Holten Rasmussen, Maxime Lefrançois, Georg Ferdinand Schneider, and Pieter Pauwels. “BOT: the building topology ontology of the W3C linked building data group”. In: *Semantic Web* 12.1 (2021), pp. 143–161. URL: https://doi.org/10.3233/SW-200385.

[145] Mads Holten Rasmussen, Pieter Pauwels, Maxime Lefrançois, and Georg Ferdinand Schneider. *Building Topology Ontology*. 2021. URL: https://w3c-lbd-cg.github.io/bot/ (accessed 2023-11-30).

[146] Mads Holten Rasmussen, Pieter Pauwels, Maxime Lefrançois, Georg Ferdinand Schneider, Christian Anker Hviid, and Jan Karlshøj. "Recent changes in the Building Topology Ontology". In: *Proceedings of the 5th Linked Data in Architecture and Construction Workshop (LDAC)*. Dijon, France, 2017. URL: https://doi.org/10.13140/RG.2.2.32365.28647.

[147] Douglas T Ross and Jorge E Rodriguez. "Theoretical foundations for the computer-aided design system". In: *Proceedings of the May 21-23, 1963, spring joint computer conference*. 1963, pp. 305–322. URL: https://doi.org/10.1145/1461551.1461589.

[148] Rafael Sacks, Zijian Wang, Boyuan Ouyang, Duygu Utkucu, and Siyu Chen. "Toward artificially intelligent cloud-based building information modelling for collaborative multidisciplinary design". In: *Advanced Engineering Informatics* 53 (2022), p. 101711. URL: https://doi.org/10.1016/j.aei.2022.101711.

[149] Robert Sanderson, Paolo Ciccarese, and Benjamin Young. *Web Annotation Data Model*. W3C Recommendation. W3C, 2017. URL: https://www.w3.org/TR/annotation-model/ (accessed 2023-11-30).

[150] Robert Sanderson, Paolo Ciccarese, and Benjamin Young. *Web Annotation Vocabulary*. W3C Recommendation. W3C, 2017. URL: https://www.w3.org/TR/annotation-vocab (accessed 2023-11-30).

[151] Sven-Eric Schapke, Jakob Beetz, Markus König, Christian Koch, and André Borrmann. "Collaborative data management". In: *Building Information Modeling: Technology Foundations and Industry Practice*. Springer, 2018, pp. 251–277. ISBN: 978-3-319-92862-3.

[152] Raimar J Scherer and S-E Schapke. "A distributed multi-model-based management information system for simulation and decision-making on construction projects". In: *Advanced Engineering Informatics* 25 (2011), pp. 582–599. URL: https://doi.org/10.1016/j.aei.2011.08.007.

[153] Raimar J Scherer, Sven-Eric Schapke, and Helga Tauscher. *Mefisto: Management - Leadership - Information - Simulation in Construction*. Technische Universitaet Dresden, 2010. ISBN: 978-3-86780-187-4.

[154] Oliver Schulz, Jyrki Oraskari, and Jakob Beetz. "bcfOWL: A BIM collaboration ontology". In: *Proceedings of the 9th Linked Data in Architecture and Construction Workshop*. Luxembourg, 2021, pp. 1–12. URL: https://ceur-ws.org/Vol-3081/12paper.pdf (accessed 2023-11-30).

[155] Oliver Schulz, Jeroen Werbrouck, and Jakob Beetz. "Towards Scene Graph Descriptions for Spatial Representations in the Built Environment". In: *30th International Workshop on Intelligent Computing in Engineering, EG-ICE 2023*. 2023. URL: https://www.ucl.ac.uk/bartlett/construction/sites/bartlett_construction/files/towards_scene_graph_descriptions_for_spatial_representations_in_the_built_environment.pdf (accessed 2024-3-18).

[156] Mikki Seidenschnur, Ali Kücükavci, Esben Visby Fjerbæk, Kevin Michael Smith, Pieter Pauwels, and Christian Anker Hviid. "A common data environment for HVAC design and engineering". In: *Automation in Construction* 142 (2022), p. 104500. URL: https://doi.org/10.1016/j.autcon.2022.104500.

[157] Madhumitha Senthilvel, Jyrki Oraskari, and Jakob Beetz. "Common Data Environments for the Information Container for linked Document Delivery". In: *Proceedings of the 8th Linked Data in Architecture and Construction Workshop*. 2020. URL: https://ceur-ws.org/Vol-2636/10paper.pdf (accessed 2023-11-30).

[158] Gustavus J Simmons. "Symmetric and asymmetric encryption". In: *ACM Computing Surveys (CSUR)* 11.4 (1979), pp. 305–330. URL: https://doi.org/10.1145/356789.356793.

[159] Single-spa. *Single-spa - A javascript router for front-end microservices*. 2018. URL: https://single-spa.js.org/ (accessed 2023-1-11).

[160] Smapiot GmbH. *Piral - Breaking the Frontend Monolith!* 2019. URL: https://piral.io/ (accessed 2023-1-11).

[161] Stuart Snydman, Robert Sanderson, and Tom Cramer. "The International Image Interoperability Framework (IIIF): A community & technology approach for web-based images". In: *Archiving conference*. Vol. 2015. 1. Society for Imaging Science and Technology, 2015, pp. 16–21. URL: https://stacks.stanford.edu/file/druid:df650pk4327/2015ARCHIVING_IIIF.pdf (accessed 2023-11-30).

[162] Wawan Solihin, Charles Eastman, and Yong Cheol Lee. "A framework for fully integrated building information models in a federated environment". In: *Advanced Engineering Informatics* 30.2 (2016), pp. 168–189. URL: https://doi.org/10.1016/j.aei.2016.02.007.

[163] Manu Sporny, Dave Longley, and David Chadwick. *Verifiable Credentials Data Model v1.1*. W3C Recommendation. W3C, 2022. URL: https://www.w3.org/TR/vc-data-model/ (accessed 2023-11-30).

[164] Manu Sporny, Dave Longley, Gregg Kellogg, Markus Lanthaler, Pierre-Antoine Champin, and Niklas Lindström. *JSON-LD: JSON for Linking Data*. 2020. URL: https://www.w3.org/TR/json-ld/ (accessed 2023-10-16).

[165] Manu Sporny, Dave Longley, Markus Sabadello, Reed Drummond, Orie Steele, and Christopher Allen. *Decentralized Identifiers (DIDs) v1.0*. W3C Recommendation. W3C, 2022. URL: https://www.w3.org/TR/did-core/ (accessed 2023-11-30).

[166] Ruben Taelman, Joachim Van Herwegen, Miel Vander Sande, and Ruben Verborgh. "Comunica: a modular SPARQL query engine for the web". In: *International Semantic Web Conference*. Springer, 2018, pp. 239–255. URL: https://doi.org/10.1007/978-3-030-00668-6_15.

[167] Ruben Taelman and Ruben Verborgh. "Evaluation of Link Traversal Query Execution over Decentralized Environments with Structural Assumptions". In: (2023). URL: https://arxiv.org/ftp/arxiv/papers/2302/2302.06933.pdf (accessed 2023-11-30).

[168] Davide Taibi and Luca Mezzalira. "Micro-Frontends". In: *ACM SIGSOFT Software Engineering Notes* 47.4 (2022), pp. 25–29. URL: https://doi.org/10.1145/3561846.3561853.

[169] Alon Talmor and Jonathan Berant. "The web as a knowledge-base for answering complex questions". In: *CoRR* abs/1803.06643 (2018). URL: https://doi.org/10.48550/arXiv.1803.06643.

[170] Seppo Törmä. "Semantic Linking of Building Information Models". In: *2013 IEEE Seventh International Conference on Semantic Computing*. 2013, pp. 412–419. URL: https://doi.org/10.1109/ICSC.2013.80.

[171] Trimble Inc. *Trimble Connect API*. 2021. URL: https://connect.trimble.com/ (accessed 2023-11-30).

[172] Irene Tzinis. *Technology Readiness Level*. 2012. URL: https://www.nasa.gov/directorates/heo/scan/engineering/technology/technology_readiness_level (accessed 2023-11-30).

[173] Joachim Van Herreweghe, Ruben Verborgh, Ruben Taelman, and et al. *Community Solid Server*. 2023. URL: https://github.com/CommunitySolidServer/CommunitySolidServer (accessed 2023-11-30).

[174] Theo Van Veen. "Wikidata: From 'an' identifier to 'the' identifier". In: *Information technology and libraries* 38.2 (2019), pp. 72–81. URL: https://doi.org/10.6017/ital.v38i2.10886.

[175] Maarten Vandenbrande. “Aggregators to realize Scalable Querying across Decentralized Data Sources”. In: *Doctoral Consortium at ISWC (ISWC-DC 2023), 08.11.2023, Athens, Greece*. Athens, Greece, 2023, p. 4. URL: https://biblio.ugent.be/publication/01HFXXKQ51BYBEMG4Z01EK4CH1/file/01HFXXN3ZQNVKX0M4J5MFHP2RH (accessed 2023-11-30).

[176] Ruben Verborgh. *Solid: innovation through personal data control*. 2021. URL: https://rubenverborgh.github.io/ECA-2021/#title (accessed 2021-3-10).

[177] Sofie Verbrugge, Frederic Vannieuwenborg, Marlies Van der Wee, Didier Colle, Ruben Taelman, and Ruben Verborgh. “Towards a personal data vault society: an interplay between technological and business perspectives”. In: *2021 60th FITCE Communication Days Congress for ICT Professionals: Industrial Data – Cloud, Low Latency and Privacy (FITCE)*. 2021, pp. 1–6. URL: https://doi.org/10.1109/FITCE53297.2021.9588540.

[178] S Vilgertshofer et al. “TwinGen: Advanced technologies to automatically generate digital twins for operation and maintenance of existing bridges”. In: *ECPPM 2022*. 2022. URL: https://mediatum.ub.tum.de/doc/1687936/document.pdf (accessed 2023-11-30).

[179] R. Volk, J. Stengel, and F. Schultmann. “Building Information Modeling (BIM) for Existing Buildings - Literature Review and Future Needs”. In: *Automation in Construction* 38 (2014), pp. 109–127. URL: https://doi.org/10.1016/j.autcon.2013.10.023.

[180] W3C OWL Working Group. *OWL 2 Web Ontology Language Document Overview (Second Edition)*. W3C Recommendation. W3C, 2012. URL: https://www.w3.org/TR/2012/REC-owl2-overview-20121211/ (accessed 2023-11-30).

[181] W3C/OGC Spatial Data on the Web Working Group. *Semantic Sensor Network Ontology*. 2017. URL: http://www.w3.org/ns/ssn/ (accessed 2023-11-30).

[182] Anna Wagner, Mathias Bonduel, Pieter Pauwels, and Uwe Rüppel. “Representing construction-related geometry in a semantic web context: A review of approaches”. In: *Automation in Construction* 115 (2020). URL: https://doi.org/10.1016/j.autcon.2020.103130.

[183] Anna Wagner, Mathias Bonduel, Jeroen Werbrouck, and Kris McGlinn. “Geometry and geospatial data on the web”. In: *Buildings and Semantics*. CRC Press, 2022, pp. 69–99.

[184] Anna Wagner, Laura Kristina Moeller, Christian Leifgen, and Christian Eller. *Building Product Ontology*. 2019. URL: https://w3id.org/bpo# (accessed 2023-11-30).

[185] Anna Wagner and Uwe Rüppel. "BPO: The Building Product Ontology for Assembled Products". In: *Proceedings of the 7th Linked Data in Architecture and Construction Workshop (LDAC)*. Lisbon, Portugal, 2019, pp. 106–119. URL: http://ceur-ws.org/Vol-2389/08paper.pdf (accessed 2019-12-5).

[186] Webpack. *Module Federation*. 2020. URL: https://webpack.js.org/concepts/module-federation/ (accessed 2023-1-11).

[187] Jeroen Werbrouck. *Pattern-based Access Control (Vocabulary)*. URL: https://w3id.org/pbac# (accessed 2024-3-18).

[188] Jeroen Werbrouck, Pieter Pauwels, Jakob Beetz, and Léon 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. URL: https://biblio.ugent.be/publication/8633673 (accessed 2024-3-18).

[189] Jeroen Werbrouck, Pieter Pauwels, Jakob Beetz, and Erik 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. URL: https://biblio.ugent.be/publication/8724183/file/8750812.pdf (accessed 2024-3-18).

[190] Jeroen Werbrouck, Pieter Pauwels, Jakob Beetz, and Erik Mannens. "Mapping Federated AEC projects to Industry Standards using dynamic Views". In: *10th Linked Data in Architecture and Construction Workshop*. CEUR-WS. org. 2022. URL: https://ceur-ws.org/Vol-3213/paper06.pdf (accessed 2024-3-18).

[191] Jeroen Werbrouck, Pieter Pauwels, Jakob Beetz, and Erik Mannens. "ConSolid: a Federated Ecosystem for Heterogeneous Multi-Stakeholder Projects". In: *Semantic Web Journal* (2023). Accepted. URL: https://biblio.ugent.be/publication/8633673/file/8633674.pdf (accessed 2024-3-18).

[192] Jeroen Werbrouck, Pieter Pauwels, Mathias Bonduel, Jakob Beetz, and Willem Bekers. "Scan-to-graph: Semantic enrichment of existing building geometry". In: *Automation in Construction* 119 (2020), p. 103286. URL: https://doi.org/10.1016/j.autcon.2020.103286.

[193] Jeroen Werbrouck, Oliver Schulz, Jyrki Oraskari, Erik Mannens, Pieter Pauwels, and Jakob Beetz. "A generic framework for federated CDEs applied to Issue Management". In: *Advanced Engineering Informatics* 58 (2023), p. 102136. URL: https://doi.org/10.1016/j.aei.2023.102136 (accessed 2024-3-18).

[194] Jeroen Werbrouck, Madhumitha Senthilvel, Jakob Beetz, Pierre Bourreau, and Léon Van Berlo. "Semantic query languages for knowledge-based web services in a construction context". In: *26th International Workshop on Intelligent Computing in Engineering, EG-ICE 2019*. Vol. 2394. 2019. URL: https://ceur-ws.org/Vol-2394/paper03.pdf (accessed 2024-3-18).

[195] Jeroen Werbrouck, Madhumitha Senthilvel, Jakob Beetz, and Pieter Pauwels. "A checking approach for distributed building data". In: *31st forum bauinformatik, Berlin: Universitätsverlag der TU Berlin*. 2019, pp. 173–81. URL: https://biblio.ugent.be/publication/8667508/file/8667516.pdf (accessed 2024-3-18).

[196] Jeroen Werbrouck, Madhumitha Senthilvel, Jakob Beetz, and Pieter Pauwels. "Querying heterogeneous linked building datasets with context-expanded graphql queries". In: *7th Linked Data in Architecture and Construction Workshop*. Vol. 2389. 2019, pp. 21–34. URL: https://biblio.ugent.be/publication/8623179/file/8623180.pdf (accessed 2024-3-18).

[197] Jeroen Werbrouck, Madhumitha Senthilvel, and Mads Holten Rasmussen. "Federated data storage for the AEC industry". In: *Buildings and Semantics*. CRC Press, 2022, pp. 139–164.

[198] Jeroen Werbrouck, Ruben Taelman, Ruben Verborgh, Pieter Pauwels, Jakob Beetz, and Erik 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. URL: https://ceur-ws.org/Vol-2636/09paper.pdf (accessed 2024-3-18).

[199] Wikimedia Foundation. *Wikidata*. 2011. URL: https://www.wikidata.org (accessed 2023-11-30).

[200] Mark D Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan-Willem Boiten, Luiz Bonino da Silva Santos, Philip E Bourne, et al. "The FAIR Guiding Principles for scientific data management and stewardship". In: *Scientific data* 3.1 (2016), pp. 1–9. URL: https://doi.org/10.1038/sdata.2016.18.

[201] Wistor. *Data-centric working made easy*. 2021. URL: https://wistor.nl (accessed 2023-3-13).

[202] Evan Yue. *Vue - The Progressive JavaScript Framework*. 2013. URL: https://vuejs.org/ (accessed 2023-1-11).

[203] Dmitri Zagidulin. *Proposal: Support Decentralized Identifiers (DIDs) in addition to Web IDs*. 2019. URL: https://github.com/solid/specification/issues/217 (accessed 2023-11-30).

Appendix A

Prefixes and Namespaces

```
# vocabularies devised in this dissertation
consolid: <https://w3id.org/consolid#>
pbac: <https://w3id.org/pbac#>
mifesto: <https://w3id.org/mifesto#>

# data vaults used in the case-study
arcadis: <https://arcadis.com/data/>
bb: <https://b-b.be/data/>
dgfb: <https://dgfb.ugent.be/data/>

# third-party vocabularies used in this dissertation
acl: <http://www.w3.org/ns/auth/acl#>
as: <https://www.w3.org/ns/activitystreams#>
bcfowl: <http://lbd.arch.rwth-aachen.de/bcfOWL#>
bot: <https://w3id.org/bot#>
cdo: <https://w3id.org/cdo#>
ct:<https://standards.iso.org/iso/21597/-1/ed-1/en/Container.rdf>
dcat: <http://www.w3.org/ns/dcat#>
dct: <http://purl.org/dc/terms/>
doap: <http://usefulinc.com/ns/doap#>
dot: <https://w3id.org/dot#>
ex: <http://example.org/>
foaf: <http://xmlns.com/foaf/0.1/>
ldp: <http://www.w3.org/ns/ldp#>
ls: <https://standards.iso.org/iso/21597/-1/ed-1/en/Linkset.rdf>
oa: <http://www.w3.org/ns/oa#>
pim: <http://www.w3.org/ns/pim/space#>
rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
rdfs: <http://www.w3.org/2000/01/rdf-schema#>
schema: <https://schema.org/>
sh: <http://www.w3.org/ns/shacl#>
solid: <http://www.w3.org/ns/solid/terms#>
vcard: <http://www.w3.org/2006/vcard/ns#>
xsd: <http://www.w3.org/2001/XMLSchema#>
```

Listing A.1: Namespaces used in this thesis and their corresponding prefixes.

Appendix B
The Semantic Web

B.1 The Semantic Web and Linked Data

The cornerstone of the Semantic Web is the Resource Description Framework (RDF) standard [100]. In RDF, individual data snippets ('Resources') are identified via Uniform Resource Identifiers (URI), which can be connected to one another using a *subject-predicate-object* format, thereby forming basic sentences (so-called 'triples'). Because a resource can be part of many triples, either as subject or object, a set of overlapping triples will result in a directed graph, where the predicate labels the edge between two nodes (Figure B.1).

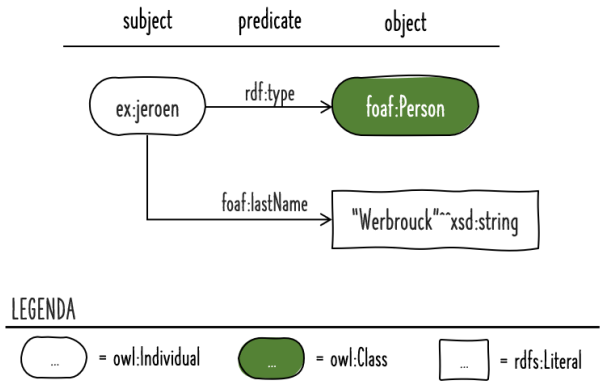

Figure B.1: RDF statements visualised as a directed graph.

Full URIs can be notated in a more concise way using *prefixed names* [7], consisting of a prefix and a local part, which are separated by a colon. The original URI can be easily reconstructed by concatenating both parts of the prefixed name. Prefixes used in this dissertation are listed in Appendix A. When a URI is not only an *identifier* for a data snippet, but also allows to *retrieve* the corresponding data, it is called a Uniform Resource Locator (URL). Using URLs, it is thus possible to unambiguously identify a concept, enrich it and retrieve additional information about this resource on the Web. Essentially, the use of HTTP URLs allows to treat the Web as a unified database and create federated knowledge graphs that integrate data from multiple independent disciplines.

When several resources together form a conceptual schema related to a specific domain, an 'ontology' or 'vocabulary' is defined. This can be done using RDF Schema (RDFS) [26] or the Web Ontology Language (OWL) [180]. Ontologies do not only organise concepts on a specific domain using formal logics, but also aid in negotiating between contradictory information on the web and making implicit information explicit through analysis of existing relationships and definitions in a graph, a process called 'inferencing'. A distinction can be made between resources that refer to concepts and are defined in ontologies, and resources that represent specific instances or data. The terms TBox and ABox refer to this dichotomy: TBox ('Terminological') refers to the group of triples defining classes and properties, while ABox ('Assertion') refers to the group of triples containing nodes that are instances of domain specific classes. The definitions in an ontology are identified with unique URIs and typically share a common namespace. For example, all the definitions in the Building Topology Ontology [146] are defined in the BOT namespace `https://w3id.org/bot#`, which allows easy notation of an ontology's concepts into prefixed names.

Listing B.1 lists an RDF graph based on the Terse RDF Triple Language [7], one of the most known serialisations of RDF. Turtle is used throughout this dissertation as the preferred serialisation for RDF examples. Prefixes for frequently occurring namespaces are omitted but can be consulted in Appendix A.

```
@prefix inst: <https://b-b.be/data/f9deed78> .

bb:DTU116 rdf:type  bot:Building .
bb:DTU116 bot:hasSpace bb:Auditorium81 .
bb:DTU116 bot:hasSpace bb:Auditorium82 .

bb:Auditorium81 rdf:type bot:Space .
bb:Auditorium82 rdf:type bot:Space .

bb:Auditorium81 bot:containsElement bb:Chair1 .
bb:Chair1 rdf:type bot:Element .
```

Listing B.1: Example RDF graph (Turtle)

Querying RDF graph patterns happens using the SPARQL Protocol And RDF Query Language (SPARQL) standard [70]. Although SPARQL can be used for querying graphs locally, it can also be used to query federated RDF resources, e.g., using the concept of link-traversal [71, 16]. Listing B.2 illustrates the basic syntax of a SPARQL query.

```
SELECT DISTINCT ?prop
WHERE {
    ?sp a bot:Space ;
        ?prop ?val .
}
```

Listing B.2: Example SPARQL query

For a more in-depth introduction and practical guide to Semantic Web technologies such as RDF and SPARQL, the reader is referred to Hendler et al. [73].

B.2 Semantic Web Technologies for the Built Environment

The application of Semantic Web technologies for the built environment effectively allows to kill two birds with one stone. Firstly, it enables the integration of modular, domain-specific data models ('ontologies') in a domain-agnostic fashion: all disciplines are considered equal. Ontologies may contain complex data patterns and allow to 'infer' explicit knowledge from implicit statements. The structured, domain-agnostic nature of a knowledge graph also allows data validation, rule checking and proof [129]. Secondly, since the building blocks of RDF graphs are URIs (or URLs), this data can be decentrally maintained on the Web, a basic requirement for the envisaged ecosystem. This offers a very high scalability compared to the assumption that all data should be present in a single CDE framework.

Research in the field of Linked Data for architecture and construction has maintained a direct link with international standards such as IFC, eventually resulting in a BuildingSMART-approved IFC schema 'ifcOWL' [11, 128]. However, because IfcOWL is designed to be as compatible with 'standard' IFC as possible, the complexity commonly associated with IFC persists with ifcOWL [20]. Therefore, focus for developing domain ontologies has recently shifted towards small, modular ontologies, as advocated by the Linked Building Data Community Group (LBD CG). The cornerstone of this modular approach is the Building Topology Ontology (BOT) [144, 145], containing only the definitions to describe topological relationship of a building: a `bot:Site` contains (`bot:hasBuilding`) one or more `bot:Building`s, which can be further destructured into `bot:Storey`s and `bot:Space`s, eventually containing `bot:Element`s. With the topological description as the spine of the model, other vocabularies can then be used to further enrich it. For example, regarding element classification there are the *BuildingElement Ontology* (BEO) [130] and

the DistributionElement Ontology (DEO) [131], respectively based upon the IfcBuildingElement and IfcDistributionElement subtrees in the IFC specification. Other classification systems can be used in parallel, such as the Getty Art and Architecture Thesaurus (AAT) [58] or Wikidata [174, 199]. Building component description can be done using the *Building Product Ontology* [185, 184], while damage classification is covered by the *Damage Topology Ontology* (DOT) [68, 67] and sensordata can be linked to the project using SSN [66, 181] and SAREF [43, 139, 42].

It is clear that this approach of combining modular ontologies is much less bound to the limits of a single 'monolithic' data model - because the data model can be extended when required by the situation. As the number of modular vocabularies grows, it becomes possible to combine their defined classes and properties in a meaningful way.

Research towards the use of Semantic Web and Linked Data technologies in the AEC domain has been carried out for several years. By now, it has resulted in several ready to use domain models of varying levels of complexity. However, with the abundance of stakeholders and their tools in the AEC market, as well as the high number of changes in design models and the diversity of classification systems and data sets used, the domain struggles with scaling up this linked data paradigm and leveraging the acknowledged potential of Linked Building Data outside academic contexts. Furthermore, as discussed in section 2.3, unstructured datasets (e.g. imagery, point clouds) and semi-structured datasets will continue to play a role. This means that even when all semantic data would be organised according to 5-star Linked Data, data structures for extendible, federated multi-models still need to support a heterogeneous (i.e., not only RDF) set of information.

For a more in-depth introduction to Semantic Web technologies applied to the built environment, the reader is referred to [133].

B.3 Validating Linked Data

In a Web-wide, data-driven environment such as ConSolid, a tug-of-war takes place between applications and data: data, expressed in open formats, exists independently from applications - however, applications are needed to make sense of the data and present it to end-users in a meaningful way. Applications are considered windows on data, but to take that role, they will demand a certain data quality.

RDF is an extremely expressive language for representing and connecting data on the Web. However, for ensuring the quality and consistency of this data, a higher-level layer is necessary. Throughout the years, various specifications have been devised for validating RDF. Two notable approaches are ShEx (Shape Expressions) [143] and SHACL (SHApes Constraint Language) [101]. While these two existed long in parallel, in 2017 SHACL became an official W3C recommendation. With SHACL, RDF data can be validated against a set of predefined constraints or *shapes* – which are expressed in RDF themselves.

For example, a tool giving information about a project's topology exposes a compatibility shape stating that every `bot:Site` instance must have at least one `bot:Building`, which in turn needs at least one bot:Storey that contains a minimum of one bot:Space. Also, the service can only work when a bot:Site is linked to a geolocation, using the bot:hasZeroPoint predicate. An example for such shape is given in Listing B.3.

For a more in-depth introduction and practical guide to Semantic Web technologies SHACL and ShEx, the reader is referred to Gayo et al. [105].

```
@prefix app: <https://www.my-app.com/shapes#> .

app:SiteShape a sh:NodeShape ;
    sh:targetClass bot:Site ;
    sh:property [
    sh:path bot:hasBuilding ;
    sh:class bot:Building ;
    sh:minCount 1 ;
    sh:message "A Site must have at least one Building" ;
  ], [
    sh:path bot:hasZeroPoint ;
    sh:minCount 1 ;
    sh:maxCount 1 ;
    sh:message "A Site must have exactly one zero Point" ;
] .

app:BuildingShape
    a sh:NodeShape ;
    sh:targetClass bot:Building ;
    sh:property [
    sh:path bot:hasStorey ;
    sh:class bot:Storey ;
    sh:minCount 1 ;
    sh:message "A Building must have at least one Storey" ;
] .

app:StoreyShape
    a sh:NodeShape ;
    sh:targetClass bot:Storey ;
    sh:property [
    sh:path bot:hasSpace ;
    sh:class bot:Space ;
    sh:minCount 1 ;
    sh:message "A Storey must have at least one Space";
] .
```

Listing B.3: SHACL shape for checking the topological structure of a built asset through the BOT ontology.

Appendix C
Solid

When datasets are not openly accessible on the Web, but only available to selected agents, an authentication mechanism needs to be in place. 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 (e.g. via tokens (2-way handshake)). Different is the Solid project for Web decentralisation [111, 176]. The Solid Protocol [33] defines how a decentralised authentication layer can be added on top of a heterogeneous LDP-compliant resource server.

Protocols for decentral identity verification and authentication are quite common already in many Web APIs with varying scope and functionality. For instance, the OpenID Connect (OIDC) [37] standard allows for user authentication and identity checking without the need to create a local account. An external Identity Provider (IDP) can confirm the identity of a client to other applications on the Web, provided that the client has an account on the servers of the identity provider . This is the background mechanism behind the well-known procedure to 'log in with X', which allows you to identify yourself without creating an account. Often, CDE solutions base upon this protocol to allow platform services to be deployed externally and communicate with the central database.

While OIDC implementations are mostly provided by large IT companies, Solid eliminates the need to rely on those large enterprises for hosting and sharing data. The authentication layer of Solid 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 C.1). Solid's combines the decentral flexibility of the WebID concept with the field-proven security of OIDC in the WebID-OIDC specification [37].

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

card:me
    foaf:name "Jeroen Werbrouck" ;
    rdf:type schema:Person, foaf:Person ;
    solid:oidcIssuer <https://pod.werbrouck.me/> .

card:
    foaf:primaryTopic card:me ;
    foaf:maker card:me ;
    rdf:type foaf:PersonalProfileDocument .
```

Listing C.1: Example solid card (Turtle) enriching the WebID.

WebIDs are often accompanied by a dedicated personal data vault, or in Solid terms, a 'Pod'. Using implementations of the Solid Specifications like the Community Solid Server (CSS) [173], anyone can thus set up a Solid-based IDP, which means that WebIDs can be self-hosted. Figure C.1 shows the relationship between the different components in Solid. In the context of this dissertation, this means that enterprises can maintain their own WebIDs and Pods, allowing them to be in control of their data. When participating in multi-stakeholder collaborative projects, an office server that implements WebID-OIDC can thus be set up to allow specific external accounts to have access to a specific project dataset. A company (or its employees) can authenticate in a standardised way to the vaults of other project participants to access their (filtered) project contributions.

Figure C.1: Relationship between a Solid Pod, a WebID and an Identity Provider

Appendix D

FAIR Data Principles

The FAIR principles are a set of guidelines to enhance the findability, accessibility, interoperability and reusability of digital resources. The main focus lies on improving machine-readability, so information exchange can take place with minimal human intervention.

Although the FAIR principles are technology-agnostic, the Semantic Web technology stack has been identified as one of the few that is able to address them to their full extents [116]. In context of this dissertation, this mainly applies to the metadata layer of the ecosystem, while leaving room for heterogeneity of actual project datasets. Each letter of the FAIR acronym corresponds with a set of sub-requirements:

- Findable
  - F1. (Meta)data are assigned a globally unique and persistent identifier;
  - F2. Data are described with rich metadata (defined by R1 below);
  - F3. Metadata clearly and explicitly include the identifier of the data they describe;
  - F4. (Meta)data are registered or indexed in a searchable resource;
- Accessible
  - A1. (Meta)data are retrievable by their identifier using a standardised communications protocol;
  - A2. Metadata are accessible, even when the data are no longer available;
- Interoperable
  - I1. (Meta)data use a formal, accessible, shared, and broadly applicable language for knowledge representation;
  - I2. (Meta)data use vocabularies that follow FAIR principles;

    - I3. (Meta)data include qualified references to other (meta)data. 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 [60];

- Reusable
    - R1. (Meta)data are richly described with a plurality of accurate and relevant attributes.

More details about the FAIR principles and how they can be implemented are described in [60].

Appendix E
Containers

E.1 Containers - Semantic Web

Linked Data Platform

The Linked Data Platform specification (LDP) presents guidelines for storage 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 (Listing E.1). 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 (Appendix C).

```
<> a pim:Storage,
    ldp:Container,
    ldp:BasicContainer,
    ldp:Resource;
    ldp:contains <picture1>,
                 <modelArchitecture.gltf>,
                 <myProjects>,
                 <profile/>,
                 <semanticsArchitecture.ttl> ;
    dct:modified "2023-06-14T10:51:22.000Z"^^xsd:dateTime .

# LDP resources
<picture1> a ldp:Resource .
<modelArchitecture.gltf> a ldp:Resource .
<myProjects> a ldp:Resource .
<semanticsArchitecture.ttl> a ldp:Resource .

# nested LDP container
<profile/> a ldp:Container,
        ldp:BasicContainer,
        ldp:Resource .
```

Listing E.1: Example LBD container definition

Data Catalog Vocabulary (DCAT)

The Data Catalog Vocabulary (DCAT) is *an RDF vocabulary designed to facilitate interoperability between data catalogues published on the Web* [4]. DCAT defines a domain-agnostic way of aggregating federated datasets in a catalogue (`dcat:Catalog`). A catalogue 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 content of the dataset. Distributions of the same dataset can differ 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` (to retrieve a dump of the dataset) or the `dcat:accessURL`. Access URLs allow to access the dataset via a database endpoint. 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`).

An example RDF description of a DCAT catalogue is given in Listing E.2.

```
<> a dcat:Catalog, dcat:Dataset;
    dcat:dataset arch:1a801545,
        arch:ec7d582b,
        eng:30613998 .
```

Listing E.2: Example DCAT Catalog and Dataset descriptions

Ontological extensions for using the DCAT vocabulary to describe metadata of AECO project resources have been proposed in [19], such as the Construction Dataset Context (CDC) ontology [18]. DCAT is the preferred vocabulary for metadata complying to the FAIR principles [119] (Appendix D).

E.2 Containers - Industry

ISO 19650

ISO 19650 [84], which is the present main international standard on CDEs, defines the BIM maturity stage 2 as 'BIM according to the ISO 19650 series'. The standard considers a CDE as a technical solution and process workflow, and provides guidelines and requirements for managing information related to the construction and operation of built assets, focusing on BIM and the

use of information containers. This includes categorisation of actors in a collaborative project team as either 'appointing party', 'lead appointed party' and 'appointed party'. Cross-organisational 'task teams' are then identified for the different tasks in the project. In Chapter 3, it is noted that this structure aligns quite well with a federated network of vaults.

ISO 19650 currently consists of five approved parts:

1. ISO 19650-1:2018 [84] - Concepts and Principles: This part of the standard sets out the fundamental concepts and principles for managing information over the whole life cycle of a built asset. It covers topics like the importance of information management, the roles and responsibilities of various parties, and the use of CDEs.

2. ISO 19650-2:2018 [85] - Delivery Phase of the Assets: This part of the standard focuses on the delivery phase of a built asset and provides specific guidance on how information should be managed during the design and construction stages. It covers aspects like the organisation and formatting of information, information exchanges, and documentation.

3. ISO 19650-3:2020 [86] - Operational phase of the assets: This part of the standard is mainly oriented towards appointing parties (e.g., owner or operator), and helps them to set information requirements during the operational phase of the asset. That is, the transfer of relevant information from a Product Information Model (PIM) (delivery phase) to an Asset Information Model (AIM) (operational phase). This includes the appointment of actors and organisations with specific responsibilities in this regard.

4. ISO 19650-4:2022 [87] - Information exchange: This part of the standard describes the decision-making process regarding the exchange of information between the parties involved in a collaborative AECO project. In particular, the transition of the publication stages defined in Part I is further documented, i.e., the steps to take to shift from a resource with status 'WIP' to a status 'shared' and 'published'.

5. ISO 19650-5:2020 [88] - Security-minded approach to information management: This part of the standard focuses on information security workflows in cross-organisation collaborations. These workflows contain the steps to cultivate an overall security mindset across organisations that deal with information about the built environment. It is not explicitly targeting CDEs.

A sixth part, 'Health and Safety' is still under development at the time of writing. It will be concerned with health and safety information in projects related to the built environment.

More information and guidance on the ISO 19650 series can be found in [36].

DIN SPEC 91391

The German DIN SPEC 91391 standard [48] is based on the ISO 19650 series and currently one of the few that further specifies these concepts into a practical setup of components and functionalities. The specification describes the functional requirements for a CDE, with a focus on the planning and construction phases. A CDE then 'realises the requirement of a redundancy-free availability of all project information and is the project-wide *single source of information* for all project participants' [48]. The standard acknowledges that there are no standards for BIM level 3 yet, but indicates that any BIM level 3 ecosystem should also fully cover the requirements for BIM level 1 and 2. A 'CDE conforming to the DIN SPEC 91391' is a modular infrastructure minimally including the following components:

1. a workflow management component (including user management)
2. a data management component
3. an administration component
4. technical facilities and digital infrastructure

The specification emphasises the role of metadata in these various components and has thereby similar metadata requirements as the FAIR principles: unique metadata that allows an information container to be addressed, retrieved and categorised based on specific parameters. Part 2 of the DIN SPEC 91391 [49] defines the OpenCDE interface for lossless and mutual data exchange between CDEs. This includes a set of predefined (obligatory) metadata parameters for the container as well as for its content, such as ID, name, container type, description, creation details etc. The OpenCDE specification comes with an API specified in the OpenAPI 3.0 format. The DIN SPEC 91391 OpenCDE is not to be confused with the OpenCDE initiative set up by buildingSMART [79], which is a 'portfolio of API standards', including the BCF API [29]and the Documents API [30].

ISO 21597 - Information Container for Linked Document Delivery

The Information Container for Linked Document Delivery (ICDD) is an industry-initiated standard (ISO 21597) for exchanging heterogeneous asset data during its life cycle, in the form of multi-models. Multi-model containers for the AECO industries were devised in context of the German Mefisto project [153], to facilitate cross-domain resource applicability and storage of resources in a persistent way, without changing the original media types [152]. A parallel predecessor project of ICDD using multi-models is the Dutch COINS project [118].

ICDD can be situated on the edge of BIM Level 2 and 3, as it makes use of Linked Data, but at the same time recognises the fact that the industry is still largely file-oriented. However, making this balance, it cannot fully exploit all benefits of Linked Data, such as *deep linking* of identifiers at a data level [65]. The specification is published in two parts, namely *Part 1: Containers* [89] and *Part 2: Link types* [90]:

1. ISO 21597-1:2020 [89] - Container: This part defines the basic structure of an information container for multi-models.
2. ISO 21597-2:2020 [89] - Link Types: This part specialises the generic link type defined in Part 1, providing more semantic accuracy on how resources can be interrelated in a sub-document way.

Data is structured in a ZIP-compressed folder structure (.icdd), containing the following minimal structure:

- a header file (*index.rdf*) in the top-level folder, describing the contained documents (minimally including a local file path (`ct:InternalDocument` or URL (`ct:ExternalDocument`) using the Container ontology [81];
- Linkset file(s) (RDF) describing the links between sub-document identifiers in the containers using the Linkset ontology [82];
- a *payload triples* folder containing the linkset files;
- a *payload documents* folder containing the documents in the container;
- an *Ontology resources* folder, optionally containing RDF documents describing the Linkset and Container ontologies. Instead of being embedded in the ICDD container, these can be referenced externally as well.

Besides defining a structure for container-based information exchange, the ICDD standard describes how to link objects together on a sub-document level (e.g., relating an IFC element to a spreadsheet cell) via GUIDs (Figure E.1).

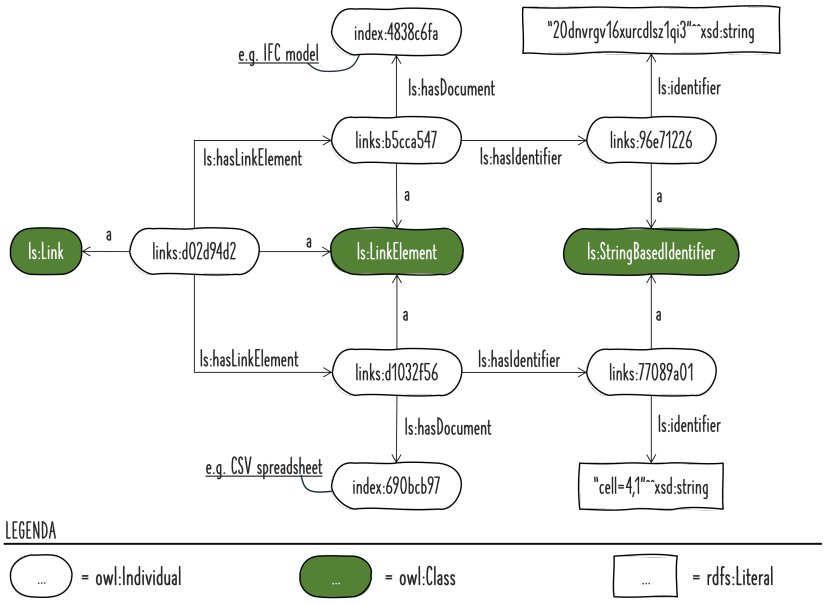

Figure E.1: Link between two documents in an ICDD container, connected through an `ls:LinkSet`

ICDD is essentially a collection of project data, it is not a database itself. Therefore, reliable querying and linking operations can only happen after uploading it to a database, exposing the data through an API to be used by micro-services etc. Example initiatives to create a (centralised) CDE based on ICDD are documented in Senthilvel et al. [157] and Hagedorn et al. [65].

Appendix F
User Interfaces

End-user applications exist in many forms. Some interfaces allow to 'directly' interact with data (e.g. a SPARQL or SQL query pane), while other interfaces allow more indirect interaction, hiding all complexities behind proxy components (e.g. 3d viewer, buttons, sliders, Natural Language Processing (NLP) input etc.). Some interfaces are hybrid as well, e.g. allowing fine-grained selection of elements with SPARQL, at the same time visualising them in a 3D viewer. Direct interaction interfaces give full expressivity to the client, but the end user must be acquainted with the used data model. Indirect interaction interfaces can optimise the experience for a broader audience, but the expressivity will be hard-coded (e.g. a click on a button will trigger a predefined query to run under-the-hood).

F.1 Standalone Applications

BIM and CAD authoring tools are desktop-based applications which have shaped the way people think about computer-aided information creation and design. Lately, many vendors of BIM authoring tools have shifted focus towards entire Web ecosystems - CDEs. The status of the authoring tool thereby changes from being the one-and-only way to interact with a BIM *file* to being one of many possible windows to project data in the cloud. In proprietary CDEs, this data will be primarily exposed through a vendor-specific API, which can be called by external services. Examples of such infrastructures are the BIMserver Javascript API [123], Autodesk Platform Services (APS) [5] and the Trimble Connect API [171]. As such APIs and libraries will have a published Javascript library, highly specialised, standalone Web applications and headless services that interact with the CDE can be easily created using plain Javascript or any frontend framework. When higher modularity and reuse of components is necessary, the concept of micro-frontends will offer a solution.

F.2 Micro-Frontends

In the field of Web interfaces, the concept of micro-frontends has recently gained traction. Micro-frontends can be considered the frontend equivalent of micro-services that have been powering the Web for a long time now: a loose-

coupled network of small, semi-independent, specialised modules, capable of addressing larger usage scenarios. Because they are isolated components, they can be programmed by independent teams in any framework that can be compiled to HTML, JavaScript and CSS (e.g. React [113], Angular [61], Vue [202] or even frameworks that are yet to be launched), and be independently deployed and updated. In this sense, a micro-frontend-based approach is more future-proof than solutions which rely upon a single front-end framework. Of course, modules that need to exchange information with one another need an agreed-upon I/O interface, which can be provided by an 'application shell'. This 'application shell' then loads remote micro-frontend modules and integrates them into a single Web interface. Micro-frontends have been adopted by major companies such as Springer, SAP and Ikea [168]. An overview of motivations, benefits and issues of a micro-frontend based architecture is provided in [135].

One long existing type of micro-frontends are Iframes (Inline Frames) are HTML elements used to embed another HTML document within the current document. Each module is encapsulated within its own iframe, meaning that they operate in separate isolated contexts. Iframes provide strong isolation between separate modules, making it easier to manage and update individual modules independently. However, communication and integration between different iframe modules often require explicit message passing or event-based mechanisms. The iframes can load independent applications or components and communicate with each other through defined APIs or message passing mechanisms.

However, iframes are mainly used for embedding external content. Sometimes a tighter integration between independent micro-frontend modules is necessary, allowing more direct communication and interaction, and code sharing. Existing frameworks that allow this are, amongst others, Piral [160], Webpack Module Federation [186], single-SPA [159] and Bit [38]. Contrasting with iFrames, these frameworks do not provide a nested browser window to another HTML page; instead the components code is shared as a module and 'injected' in the final interface.

The standard approach for client-side composition of web pages is nowadays to code local modules and bundle these at buildtime ('buildtime integration') into a set of Javascript, HTML and CSS files. This results in a static integration of code into a single bundle with predefined dependencies. An advantage of this way of working is that (syntax) errors are located before the application

is running, that the end result is consistent and predictable, and that performance optimisation is possible before bundling. On the other hand, 'run-time integration' happens when modules are loaded while the application is already running, i.e., remote modules can be fetched in an asynchronous way and then integrated in the overall framework. This approach requires more robust error-handling procedures, but allows full separation of the modules, which can be developed and deployed individually on separate servers.

The choice between iframes and a more integrated approach for code-sharing, as well as the choice between runtime and buildtime integration, depends on factors such as the specific requirements, architectural preferences, communication needs, and deployment workflows of the envisaged micro-frontend-based interface.

Currently, micro-frontends are mostly associated with reducing the workload for frontend development within companies, due to their independence. To the author's knowledge, they have not been applied to address the heterogeneity of multi-models from the perspective of end user interfaces in a Web-wide context (see Chapter 7).

Appendix G

Identifier Conformance for Selectors

This appendix lists some frequently occurring resources and their internal sub-document identifiers. This list is based on the‘*fragment selectors*’ mentioned in WADM [149] (Table G.1). Some extensions proposed by the author of this dissertation with selectors for AECO-related media types are listed in Table G.2.

Name	Fragment Specification (dct:conformsTo)	Description (rdf:value)
HTML	http://tools.ietf.org/rfc/rfc3236	Example: namedSection
PDF	http://tools.ietf.org/rfc/rfc3778	Example: page=10&viewrect=50,50,640,480
Plain Text	http://tools.ietf.org/rfc/rfc5147	Example: char=0,10
XML	http://tools.ietf.org/rfc/rfc3023	Example: xpointer(/a/b/c)
RDF/XML	http://tools.ietf.org/rfc/rfc3870	Example: namedResource
CSV	http://tools.ietf.org/rfc/rfc7111	Example: row=5-7
Media	http://www.w3.org/TR/media-frags/	Example: xywh=50,50,640,480
SVG	http://www.w3.org/TR/SVG/	Example: svgView(viewBox(50,50,640,480))

Table G.1: Fragment Selector specifications for various file types. Source: [149].

Name	Fragment Specification (dct:conformsTo)	Description (rdf:value)
IFC	https://technical.buildingsmart.org/resources/ifcimplementationguidance/ifc-guid/	Example: 1xS3BCk291UvhgP2dvNMQJ
RDF	http://www.w3.org/1999/02/22-rdf-syntax-ns#	Example: http://example.org/building#room1
Images	https://iiif.io/api/image/3.0/	Example: {image-URL}/pct:20,10,25,55/max/0/default

Table G.2: Fragment Selector details for various file types as proposed in this dissertation.

Appendix H

Vocabulary: ConSolid

The namespace of the ConSolid vocabulary is https://w3id.org/consolid#.

```
# Ontology publication details
<https://w3id.org/consolid#> rdf:type owl:Ontology ;
    owl:versionIRI <urn:absolute:0.0.1> ;
    <http://purl.org/dc/terms/contributor> "Erik Mannens" , "Jakob Beetz"
     , "Pieter Pauwels" ;
    <http://purl.org/dc/terms/creator> "Jeroen Werbrouck" ;
    <http://purl.org/dc/terms/issued> "2022-02-01T12:00:00"^^xsd:dateTime
     ;
    <http://purl.org/dc/terms/license> <https://creativecommons.org/
     licenses/by/1.0/> ;
    <http://purl.org/dc/terms/title> "consolid vocabulary" .

# Object Properties
:aggregates rdf:type owl:ObjectProperty ;
    rdfs:comment "The consolid:Reference-s or other consolid:
     ReferenceCollection-s aggregated by this consolid:ReferenceCollection
     ." ;
    rdfs:domain :ReferenceCollection ;
    rdfs:label "aggregates" .

:hasSatellite rdf:type owl:ObjectProperty ;
    rdfs:domain <http://www.w3.org/ns/dcat#Dataset> ;
    rdfs:range :Satellite ;
    rdfs:label "has Satellite"@en .

:hasServiceRegistry rdf:type owl:ObjectProperty ;
    rdfs:subPropertyOf :hasRegistry ;
    rdfs:domain :PartialProject ;
    rdfs:range :ServiceRegistry ;
    rdfs:label "has service registry"@en .

:hasSparqlSatellite rdf:type owl:ObjectProperty ;
    rdfs:subPropertyOf :hasSatellite ;
    rdfs:range :SparqlSatellite ;
    rdfs:label "has SPARQL satellite"@en .

:hasShapeCollection rdf:type owl:ObjectProperty ;
    rdfs:domain dcat:Catalog ;
    rdfs:range :ShapeCollection ;
    rdfs:subPropertyOf dcat:dataset ;
    rdfs:label "has rule collection"@en ;
    rdfs:comment "The rule collection of a catalog in the ConSolid
     ecosystem."@en .

# Classes
:Project rdf:type owl:Class ;
    rdfs:subClassOf <http://www.w3.org/ns/dcat#Catalog> ;
```

```
    rdfs:comment "A ConSolid Project is a DCAT Catalog that resolves to a federated multi-model. It may be recursive, i.e. ConSolid Projects may 'aggregate' each other via dcat:dataset" ;
    rdfs:label "Project" .

:ProjectResource rdf:type owl:Class ;
    rdfs:subClassOf <http://www.w3.org/ns/dcat#Dataset> ;
    rdfs:comment "A Project Resource is a dcat:Dataset describing project information such as BIM models, project plans, etc." ;
    rdfs:label "Project Resource" .

:ValidationResource rdf:type owl:Class ;
    rdfs:subClassOf <http://www.w3.org/ns/dcat#Dataset> ;
    rdfs:comment "A Validation Resource is a dcat:Dataset describing a resource with the purpose of validation." ;
    rdfs:label "Validation Resource" .

:ReferenceRegistry rdf:type owl:Class ;
    rdfs:subClassOf <http://www.w3.org/ns/dcat#Dataset> ;
    rdfs:comment "A Reference Registry is a dcat:Dataset describing a resource containing Reference Collections (i.e. 'ConSolid linksets')." ;
    rdfs:label "Reference Registry" .

:ReferenceCollection rdf:type owl:Class ;
    rdfs:comment "A Reference Collection collects digital references of the same 'abstract' concept. It may be seen as the top-level identifier of this concept as well." ;
    rdfs:label "Reference Collection" .

:Reference rdf:type owl:Class ;
    rdfs:comment "A Reference is the digital manifestation of an abstract concept." ;
    rdfs:label "Reference" .

:ShapeCollection rdf:type owl:Class ;
    rdfs:subClassOf dcat:Catalog ;
    rdfs:comment "A Shape Collection is a DCAT Catalog exclusively containing shapes." ;
    rdfs:label "Shape Collection" .

:Satellite rdf:type owl:Class ;
    rdfs:subClassOf <http://www.w3.org/ns/dcat#DataService> ;
    rdfs:comment "A consolid:Satellite is a highly trusted service, connected to a limited set of Solid Pods, mostly just one. It may function as an alternative storage mechanism, providing specific access to (nested) ldp:Containers, dcat:Datasets, dcat:Catalogs or dcat:Distributions. Satellites can also be thought of as 'digital assistents' to a Pod, aiding in synchronisation, data management etc. When working on a Solid Pod, it must be able to check access rights." ;
    rdfs:label "Satellite" .

:SparqlSatellite rdf:type owl:Class ;
    rdfs:subClassOf :Satellite ;
    rdfs:comment "A SPARQL Satellite is a specifiic sort of satellite, mirroring RDF resources on the Pod and allowing to query their union with SPARQL. The satellite should check the access rights for every
```

```
     result it acquires." ;
    rdfs:label "SPARQL satellite" .

:RuntimeProperty rdf:type owl:Class ;
    rdfs:comment "A Runtime Property is a property that cannot be
     explicitly present in the data, but needs to be generated dynamically
     . E.g. when in order to conform to a specific shape, a dataset must
     refer to a project ID, but in ConSolid a dataset may be part of
     multiple projects. In this case, the project ID must be generated
     dynamically." ;
    rdfs:label "Runtime Property" .
```

Listing H.1: The ConSolid Vocabulary. https://w3id.org/consolid#.

Appendix I

Vocabulary: PBAC

The namespace of the PBAC vocabulary is https://w3id.org/pbac#.

```
# Ontology publication details
<https://w3id.org/pbac#> rdf:type owl:Ontology ;
  owl:versionIRI <urn:absolute:0.0.1> ;
  <http://purl.org/dc/terms/contributor> "Erik Mannens" , "Jakob Beetz" ,
     "Pieter Pauwels" ;
  <http://purl.org/dc/terms/creator> "Jeroen Werbrouck" ;
  <http://purl.org/dc/terms/issued> "2023-09-29T12:00:00"^^xsd:dateTime ;
  <http://purl.org/dc/terms/license> <https://creativecommons.org/licenses
     /by/1.0/> ;
  <http://purl.org/dc/terms/title> "PBAC vocabulary" .

# Object Properties
:visitorRequirement rdf:type owl:ObjectProperty ;
    rdfs:domain :DynamicRule ;
    rdfs:range :Requirement ;
    rdfs:label "visitor requirement"@en ;
    rdfs:comment "The visitor requirement of a dynamic rule."@en .

:dataRequirement rdf:type owl:ObjectProperty ;
    rdfs:domain :DynamicRule ;
    rdfs:range :Requirement ;
    rdfs:label "data requirement"@en ;
    rdfs:comment "The data requirement of a dynamic rule."@en .

:hasTrustedAuthority rdf:type owl:ObjectProperty ;
    rdfs:domain :Requirement ;
    rdfs:range :TrustedAuthority ;
    rdfs:label "has trusted authority"@en ;
    rdfs:comment "The trusted authority of a ."@en .

# Classes
:RuleCollection rdf:type owl:Class ;
    rdfs:subClassOf dcat:Catalog ;
    rdfs:label "Rule collection"@en ;
    rdfs:comment "A collection of rules."@en .

:CredentialResource rdf:type owl:Class ;
    rdfs:label "Credential resource"@en ;
    rdfs:subClassOf dcat:Dataset ;
    rdfs:comment "A DCAT dataset containing the metadata about a set of
     signed statements that can be used for access control."@en .

:AccessResource rdf:type owl:Class ;
    rdfs:label "Access resource"@en ;
    rdfs:subClassOf dcat:Dataset ;
    rdfs:comment "A DCAT dataset containing the metadata of a PBAC dynamic
      rule."@en .
```

```
44
45 :TrustedAuthority rdf:type owl:Class ;
46     rdfs:label "Trusted authority"@en ;
47     rdfs:comment "A trusted authority for making statements about another
        entity."@en .
48
49 :Requirement rdf:type owl:Class ;
50     rdfs:label "Requirement"@en ;
51     rdfs:comment "A requirement for accessing a resource."@en .
52
53 :DynamicRule rdf:type owl:Class ;
54     rdfs:label "Dynamic rule"@en ;
55     rdfs:comment "An access constrol rule in the PBAC framework, which can
         check the properties of both the requester and the requested
        resource."@en .
56
57 :Visitor rdf:type owl:Class ;
58     rdfs:label "Visitor"@en ;
59     rdfs:comment "An agent requesting access to a resource."@en .
60
61 :Issuer rdf:type owl:Class ;
62     rdfs:label "Issuer"@en ;
63     rdfs:comment "An agent issuing a statement about another entity."@en .
```

Listing I.1: The PBAC Vocabulary. https://w3id.org/pbac#.

Appendix J

Vocabulary: Mifesto

The namespace of the Mifesto vocabulary is https://w3id.org/mifesto#.

```
# Ontology publication details
<https://w3id.org/mifesto#> rdf:type owl:Ontology ;
    owl:versionIRI <urn:absolute:0.0.1> ;
    <http://purl.org/dc/terms/contributor> "Erik Mannens" , "Jakob Beetz"
     , "Pieter Pauwels" ;
    <http://purl.org/dc/terms/creator> "Jeroen Werbrouck" ;
    <http://purl.org/dc/terms/issued> "2023-09-29T12:00:00"^^xsd:dateTime
     ;
    <http://purl.org/dc/terms/license> <https://creativecommons.org/
     licenses/by/1.0/> ;
    <http://purl.org/dc/terms/title> "Mifesto vocabulary" .

# Object Properties
:code rdf:type owl:ObjectProperty ;
    rdfs:subClassOf dcat:accessURL ;
    rdfs:comment "The URL of the loadable code of a micro frontend module
     ." ;
    rdfs:domain dcat:Distribution ;
    rdfs:label "code" .

:readsIdentifier rdf:type owl:ObjectProperty ;
    rdfs:comment "What kind of identifier the module can interpret." ;
    rdfs:label "reads identifier" .

:writesIdentifier rdf:type owl:ObjectProperty ;
    rdfs:comment "What kind of identifier the module can write." ;
    rdfs:range :IdentifierDefinition ;
    rdfs:domain dcat:Distribution ;
    rdfs:label "writes identifier" .

:compatibleMedia rdf:type owl:ObjectProperty ;
    rdfs:comment "The media type that the module can handle." ;
    rdfs:range :IdentifierDefinition ;
    rdfs:domain dcat:Distribution ;
    rdfs:label "compatible media" .

:usesVocabulary rdf:type owl:ObjectProperty ;
    rdfs:comment "The vocabulary that the module uses for semantic
     enrichment." ;
    rdfs:domain dcat:Distribution ;
    rdfs:range owl:Ontology ;
    rdfs:label "uses vocabulary" .

:hasModule rdf:type owl:ObjectProperty ;
    rdfs:comment "The module that is used in the interface." ;
    rdfs:domain :Component ;
    rdfs:range :Manifest ;
```

```
    rdfs:label "has module" .

:hosts rdf:type owl:ObjectProperty ;
    rdfs:comment "The module that is used in the interface." ;
    rdfs:domain :OrganisationalComponent ;
    rdfs:range :Component ;
    rdfs:label "hosts" .

:hasRoute rdf:type owl:DatatypeProperty ;
    rdfs:comment "The route resolving to a mifesto:Page instance." ;
    rdfs:domain :Page ;
    rdfs:label "has route" .

:hasSPARQL rdf:type owl:DatatypeProperty ;
    rdfs:comment "Links a Bundler variable to a SPARQL query that is used
     to retrieve the data from the local RDF store." ;
    rdfs:label "has SPARQL" .

:hasDimensionSetting rdf:type owl:ObjectProperty ;
    rdfs:comment "The dimension setting that is used for a Page." ;
    rdfs:domain :Component ;
    rdfs:range :DimensionSetting ;
    rdfs:label "has dimension setting" .

:hasDimension rdf:type owl:ObjectProperty ;
    rdfs:comment "The dimension that is used for a module hosted by a Page
     ." ;
    rdfs:domain :Dimension ;
    rdfs:range owl:DatatypeProperty ;
    rdfs:label "has dimension" .

:initialColumns rdf:type owl:DatatypeProperty ;
    rdfs:comment "The initial number of columns for a grid dimension." ;
    rdfs:domain :GridDimension ;
    rdfs:label "initial columns" .

:initialRows rdf:type owl:DatatypeProperty ;
    rdfs:comment "The initial number of rows for a grid dimension." ;
    rdfs:domain :GridDimension ;
    rdfs:label "initial rows" .

:isActive rdf:type owl:DatatypeProperty ;
    rdfs:comment "Indicates whether a resource is currently activated in
     the interface or not." ;
    rdfs:label "is active" .

# Classes
:Store rdf:type owl:Class ;
    rdfs:subClassOf dcat:Catalog ;
    rdfs:comment "A store is a dereferenceable, federated catalog pointing
      to other stores, configurations and manifests of Mifesto modules." ;
    rdfs:label "Store" .

:Configuration rdf:type owl:Class ;
    rdfs:subClassOf dcat:Catalog ;
    rdfs:comment "A Configuration is a semantic description of a GUI
     consisting of federated micro frontend modules." ;
    rdfs:label "Configuration" .
```

```
96
97  :Manifest rdf:type owl:Class ;
98      rdfs:comment "A Manifest is the semantic description of a micro
         frontend module.";
99      rdfs:subClassOf dcat:Dataset ;
100     rdfs:label "Manifest" .
101
102 :Component rdf:type owl:Class ;
103     rdfs:comment "A Component indicates that something is to be loaded in
         the interface of a particular configuration." ;
104     rdfs:label "Component" .
105
106 :Page rdf:type owl:Class ;
107     rdfs:subClassOf dcat:Catalog ;
108     rdfs:comment "Class indicating that something is to be a separate page
          in the interface of a particular configuration." ;
109     rdfs:label "Page" .
110
111 :IdentifierDefinition rdf:type owl:Class ;
112     rdfs:label "Identifier Definition" .
113
114 :OrganisationalComponent rdf:type owl:Class ;
115     rdfs:subClassOf :Component ;
116     rdfs:comment "Class indicating that a resource is an organisational
         component, i.e it does not have any content of its own, but requires
         child components" ;
117     rdfs:label "Organisational Component" .
118
119 :DimensionSetting rdf:type owl:Class ;
120     rdfs:comment "Class indicating that a resource is a dimension setting
         in the interface" ;
121     rdfs:label "Dimension Setting" .
122
123 :GridDimensionSetting rdf:type owl:Class ;
124     rdfs:comment "Class indicating that a resource is a grid dimension
         setting, i.e. there must be a row-column layout for this Component or
          Page " ;
125     rdfs:subClassOf :DimensionSetting ;
126     rdfs:label "Grid Dimension Setting" .
127
128 :Dimension rdf:type owl:Class ;
129     rdfs:comment "A Dimension contains a semantic description of the
         dimensions of an module." ;
130     rdfs:label "Dimension" .
131
132 :GridDimension rdf:type owl:Class ;
133     rdfs:comment "A Grid Dimension contains a semantic description of the
         dimensions of an module, as a grid definition with rows and columns."
          ;
134     rdfs:subClassOf :Dimension ;
135     rdfs:label "Grid Dimension" .
```

Listing J.1: The Mifesto Vocabulary. https://w3id.org/mifesto#.