FACULTY OF ENGINEERING AND ARCHITECTURE

BIM without 3D models: Towards an adaptive BIM environment for renovations using Linked Data

Robbe Michiels
Student number: 01706531

Supervisor: Prof. ir.-arch. Paulus Present
Counsellors: Ir.-arch. Jeroen Werbrouck, Prof. dr. ir. arch. Ruben Verstraeten

Master's dissertation submitted in order to obtain the academic degree of
Master of Science in de ingenieurswetenschappen: architectuur

Academic year 2022-2023

GHENT UNIVERSITY

Acknowledgements

First of all, I would like to thank Paulus Present for being my promoter and for always providing a pragmatic perspective in an otherwise purely academic environment. I'd also like to thank Jeroen Werbrouck for being my counsellor for the past year, without whom this thesis would not have been nearly as high quality as it is today. I have been greatly inspired by your own research and I am grateful to you for guiding me through the world of Linked Data and software development. I would like to thank both of you again for introducing me to the concepts and practical applications of BIM and coding in general.

I would also like to thank the people who have supported me throughout the year. Hannelore and Flor, for always being eager to have lunch together, listening to my struggles with my thesis and discussing our current and future situation at length. Jas, for discussing both of our theses even though neither of us had a clue what the other was talking about. Hannah, for your unconditional support throughout. Finally, I would like to thank my parents, who have always gone out of their way to support me throughout my entire, and sometimes bumpy, education.

Over the past year I have enjoyed exploring a combination of aspects that fascinate and motivate me in architecture and engineering. Alternating between reading, discussing and coding has deepened my understanding of all the topics discussed and gives me a drive to turn this into practical implementations in the coming years.

Enjoy reading!

Robbe Michiels
8th of June, 2023

Deze masterproef vormt een onderdeel van een examen. Eventuele opmerkingen die door de beoordelingscommissie tijdens de mondelinge uiteenzetting van de masterproef werden geformuleerd, werden niet verwerkt in deze tekst.

This master's dissertation is part of an exam. Any comments formulated by the assessment committee during the oral presentation of the master's dissertation are not included in this text.

BIM without 3D models:

Towards an adaptive BIM environment for renovations using Linked Data

Robbe Michiels

Supervisor: Ir.-arch. Paulus Present

Counsellors: Ir.-arch. Jeroen Werbrouck,
Prof. dr. ir.-arch. Ruben Verstraeten

Master's dissertation submitted in order to obtain the academic degree of Master of Science in de ingenieurswetenschappen: architectuur

Abstract

The implementation of Building Information Modelling (BIM) in the Architecture, Engineering, and Construction (AEC) industry has significantly improved information handling during construction projects. However, the industry struggles in achieving widespread standardisation, while addressing the variety of requirements in the involved sectors. In addition, current BIM applications primarily focus on 3D modeling rather than a common data model.

This study addresses this issue by proposing a modular approach using Linked Data to create an extensible BIM environment. The research focuses on adopting BIM in renovation practices, where existing buildings pose unique challenges. The aim is to create a BIM model that gradually increases in complexity throughout the entire project with the process of BIMification. To integrate micro front-ends in a BIM environment, the required metadata is evaluated and brought into a specific ontology. The theoretical framework offers a solution for creating an open BIM environment with improved interoperability, that can be initialised with just information. A prototype application demonstrates the extensibility of the BIM environment through the integration of micro front-ends. This research contributes to the development of a micro front-end-based BIM environment, enabling any application to edit BIM models through external services.

Keywords: BIM, Linked Data, Micro Frontends, BIMification

Extended Abstract

The Architecture, Engineering and Construction (AEC) industry has been revolutionised by the development of Building Information Management (BIM) as a workflow for improved information handling during construction projects. However, its implementation hasn't been equally developed across the industry. Current BIM applications focus more on 3D modelling rather than on creating a common data model for all stakeholders. To remedy this, the Industry Foundation Classes (ifc) have been developed to provide a standardised BIM format. Although the ifc format has proven its value in contemporary practice, it has struggled with international implementation due to its monolithic nature. This prevents the format from providing a tailored solution to the diverse needs of the industry.

These issues have led to the current research discourse towards modular data models that could adapt and extend its use as the construction project evolves. In this approach, a BIM model is built from several smaller BIM models provided by the stakeholders. This has the advantage that each stakeholder can use a specialised application for their discipline, while still contributing to a single model.

One possible solution for creating such a modular BIM model is to use Linked Data, a concept from the Semantic Web. In this approach, data is directly linked to each other, rather than just the files that contain the data. This inherently enables greater interoperability while separating the data format from the application.

Research Objectives

As a use case, this thesis focuses on adopting BIM in renovation practices. This sector is lagging behind the rest of the AEC industry in terms of digitisation, due to the inherent problem of the existing building. Current practice often involves an intensive remodelling process of the existing situation prior to the start of a BIM workflow. However, the existing building provides a substantial pool of information that would make it even more amenable to a BIM workflow that focuses on creating a common data model.

Each stakeholder in a construction project has different needs for information about an existing building. In order to leverage this diversity, it is necessary to have an adaptive BIM environment. To achieve this, this thesis uses Linked Data to create an extensible data model. The current research discourse on Linked Data in AEC has already covered most aspects of construction, but the tools and applications produced rarely provide the necessary metadata to implement them in a larger BIM environment. The latter is important to make this adaptive BIM environment because it allows the user interface (UI) to change to the needs of both the stakeholder and the BIM model.

The aim of this research is therefore twofold. First, it discusses a process to create a BIM model from the information already present in an existing building. For this, the process of BIMification (Scherer R. & Katranushkov P. 2018 [2]) will be adapted to gradually increase the modelling complexity. This will lead to the initialisation of a BIM model without a 3D model.

Secondly, to make any BIM environment adaptive, smaller applications or services are implemented to extend the use of the initial environment. To enable this implementation, each application should provide some metadata or information about its inner workings. Thus, this research will evaluate the metadata required to implement smaller applications into a larger BIM environment. The implementation itself will be achieved through the use of micro front-ends distributed by a store. These micro front-ends will then be described in their own manifest, detailing the specifications of the service provided. To enable this store, the Micro Front-End Store Ontology (Mifesto) (Werbrouck et al. 2022 [4]) will be revised to facilitate implementation in a Linked Data-based BIM environment.

Combining these research objectives should provide a theoretical framework for developing a micro front-end based BIM environment, with the goal of initiating BIM models where first information is created before the modelling complexity is gradually increased. As a proof of concept, a prototype application is created[1]. Here, a user can initiate a rudimentary BIM model based on the Building Typology Ontology (BOT) (Rasmussen et al. 2020 [1]). External micro front-ends can then be implemented through a store component of the application to extend its UI. The environment can include external functionalities that weren't part of the original application; thus creating specialised workflows. The structure of this prototype is based on the principles of ConSolid (Werbrouck et al. 2022 [3]) and the possibility of many-to-many enrichment.

[1]The code can be found at `https://github.com/RobbeMic/thesis_Digital_Environment`

Outline

In **Chapter 1** the current situation in BIM applications is outlined, with a particular focus on the structure of the data model they provide. This leads to the scope of the thesis, which sets out the research questions, followed by the structure of the remaining chapters.

Chapter 2 discusses the current research discourse on creating BIM models for the existing context. It considers several ways of approaching the subject of BIM for renovations. First, the practice of scanning a building to create a 3D model is reviewed. Then the BIM4Ren research is discussed, focusing in particular on the pilots related to initialising a BIM model. In addition, the proposed scheme for combining these pilots into a collaborative platform is highlighted. Finally, the process of BIMification is reviewed in the context of initialising an ”information-first” BIM model.

As an introduction to the topic, **Chapter 3** goes over the concepts of Linked Data relevant to this thesis. The chapter begins with an introduction to the basics of Linked Data and the Semantic Web. This includes Resource Description Framework (RDF) graphs and SPARQL Protocol and RDF Query Language (SPARQL) queries for representing and retrieving data respectively. It then puts this into the context of the AEC industry by discussing BIM specific Linked Data implementations. Finally, the chapter highlights additional frameworks in the context of web development, such as the Solid ecosystem and micro front-ends.

Chapter 4 discusses the methodology of the proposed theoretical framework. This starts by focusing on creating a BIM model for existing buildings, for which the BIMification process is reviewed and a modification is proposed. The structure for an extensible BIM environment is outlined, starting from an adaptation of the collaboration schema proposed in the BIM4Ren research. The resulting graph scheme for a Linked Data-based BIM model is then discussed. Finally, a revision of the Mifesto ontology is proposed. The revision focuses on providing an accessible and practical framework for the federation of micro front-ends into a BIM environment.

The prototype application serves as a proof of concept for the theoretical framework and is outlined in **Chapter 5**. Each component is briefly discussed in the context of the extensible BIM environment rather than the functionalities themselves. The focus is on the interaction between the user, the main application and the micro front-ends, as it is the store concept and the Mifesto ontology that are being tested.

Ultimately, in **Chapter 6**, the prototype is evaluated in the context of providing an extensible BIM environment. This leads to a reflection upon the whole methodology of the theoretical framework. In this reflection, the shortcomings of the scope of the thesis come to the fore, which leads to this scope being passed in a possible future work section. Finally, the thesis concludes with a brief answer to the research questions posed in the introduction.

Conclusion

The proposed theoretical framework consists of three successive components: the BIMification process, the extensible BIM environment and the Mifesto ontology. First, the BIMification process provides an approach to gradually increase the complexity of a BIM model. While the use of Linked Data isn't necessary to adopt this process, it does provide a viable way to create an information-first BIM model without the need for 3D representation. The extensible BIM environment is then able to represent the data that is described in a RDF graph. Furthermore, the environment can be extended with external functionalities provided by micro front-ends, implemented using the application store concept. Finally, the revision of the Mifesto ontology provides a structured approach for describing the metadata required for both navigation and implementation of the micro front-ends. Due to the nature of Linked Data, the manifests provided by the micro front-ends can be extended with additional semantics. The whole framework illustrates a possible solution for many-to-many enrichment in the domain of BIM software.

The developed prototype highlights that the proposed theoretical framework can be used as a rationale for an open BIM environment; in which any application can edit any BIM model through any external service.

References

[1] Mads Holten Rasmussen et al. "BOT: The building topology ontology of the W3C linked building data group". en. In: *Semantic Web* 12.1 (Nov. 2020). Ed. by Krzysztof Janowicz, pp. 143–161. ISSN: 22104968, 15700844. DOI: 10.3233/SW-200385. URL: https://www.medra.org/servlet/aliasResolver?alias=iospress&doi=10.3233/SW-200385 (visited on 01/12/2023).

[2] Raimar J. Scherer and Peter Katranuschkov. "BIMification: How to create and use BIM for retrofitting". en. In: *Advanced Engineering Informatics* 38 (Oct. 2018), pp. 54–66. ISSN: 14740346. DOI: 10.1016/j.aei.2018.05.007. URL: https://linkinghub.elsevier.com/retrieve/pii/S1474034617305372 (visited on 01/12/2023).

[3] Jeroen Werbrouck et al. "ConSolid: a Federated Ecosystem for Heterogeneous Multi-Stakeholder Projects". (in press). 2023.

[4] Jeroen Werbrouck et al. "Practical Semantic Enrichment of AEC Multi Models with Dynamic Micro Frontend Configurations". en. Manuscript under review. Dec. 2022.

Uitgebreid abstract

De *Architecture, Engineering and Construction (AEC)* industrie is gerevolutioneerd door de ontwikkeling van *Building Information Management (BIM)* als een benadering voor verbeterde informatieverwerking tijdens bouwprojecten. De toepassing ervan is echter niet overal even sterk ontwikkeld. De huidige BIM toepassingen richten zich meer op 3D-modellering dan op het creëren van een gemeenschappelijk datamodel voor alle belanghebbenden. Om dit te verhelpen zijn de *Industry Foundation Classes (ifc)* ontwikkeld om een gestandaardiseerd BIM formaat te bieden. Hoewel het ifc formaat zijn waarde heeft bewezen in de hedendaagse praktijk, heeft het moeite met internationale implementatie vanwege zijn monolithische karakter. Hierdoor kan ifc geen aangepaste oplossing bieden voor de uiteenlopende noden binnen de sector.

Dit heeft geleid tot het huidige onderzoek naar modulaire datamodellen die zich kunnen aanpassen en uitbreiden naarmate het bouwproject evolueert. In deze benadering wordt een BIM model opgebouwd uit verschillende kleinere modellen die door de actoren worden aangeleverd. Dit heeft het voordeel dat elke betrokkene een gespecialiseerde toepassing voor zijn discipline kan gebruiken, terwijl ze toch bijdragen aan één enkel model.

Een mogelijke oplossing om een dergelijk modulair BIM model te creëren is om gebruik te maken van Linked Data, een concept van het Semantisch Web. Bij deze aanpak worden gegevens rechtstreeks aan elkaar gekoppeld, in plaats van enkel aan de bestanden die de gegevens bevatten. Dit maakt inherent een grotere interoperabiliteit mogelijk en scheidt het dataformaat van de toepassing.

Onderzoeksdoelen

Als casus richt dit proefschrift zich op de toepassing van BIM in de renovatiepraktijk. Deze sector loopt op het vlak van digitalisering achter op de rest van de AEC industrie, vanwege het inherente probleem van het bestaande gebouw. De huidige gang van zaken omvat vaak een intensief remodelleerproces van de bestaande situatie vooraleer een BIM werkwijze wordt opgestart. Het bestaande gebouw biedt echter een aanzienlijke hoeveelheid informatie, waardoor het nog beter geschikt zou zijn voor een BIM werkwijze die gericht is op het creëren van een gemeenschappelijk datamodel.

Elke partij in een bouwproject heeft verschillende behoeften aan informatie over een bestaand gebouw. Om deze diversiteit te benutten is een adaptieve BIM omgeving nodig. Om dit te bereiken zal dit proefschrift Linked Data gebruiken om een aanpasbaar datamodel te creëren. Het huidige discours over Linked Data in AEC heeft de meeste aspecten van de bouw al behandeld, maar de geproduceerde tools en toepassingen bieden zelden de nodige metadata om ze in een grotere BIM omgeving te implementeren. Dit laatste is belangrijk om deze adaptieve BIM omgeving te creëren aangezien het toelaat om de *user interface (UI)* zich te laten aanpassen naargelang zowel de noden van de gebruiker als het BIM model.

Het doel van dit onderzoek is daarom tweeledig. Ten eerste wordt een proces besproken om een BIM-model te creëren op basis van de informatie die reeds aanwezig is in een bestaand gebouw. Hiervoor wordt het proces van BIMificatie (Scherer R. & Katranushkov P. 2018 [2]) aangepast om de complexiteit van het model geleidelijk te verhogen. Dit zal leiden tot de aanvang van een BIM model zonder 3D model.

Ten tweede, om eender welke BIM omgeving adaptief te maken, moeten kleinere applicaties of functionaliteiten geïmplementeerd worden om zo de bruikbaarheid van de omgeving te verbreden. Om deze implementatie mogelijk te maken, zou elke applicatie een metadata moeten aanrijken, afwel informatie over de applicatie zelf. Hierom zal dit onderzoek de metadata evalueren die nodig is om kleinere applicaties te implementeren in een grotere BIM omgeving. De implementatie zelf zal gebeuren door het gebruik van micro front-ends gedistribueerd door een *store*. Deze micro front-ends zullen dan worden beschreven in hun eigen manifest, waarin de specificaties van de geleverde diensten worden beschreven. Om deze opslag mogelijk te maken zal het *Micro Front-End Store Ontology (Mifesto)* (Werbrouck et al. 2022 [4]) worden herzien om implementatie in een Linked Data-gebaseerde BIM omgeving te vergemakkelijken.

Wanneer deze onderzoeksdoelen gecombineerd worden, zou er een theoretisch kader aangerijkt worden voor de creatie van een micro front-end gebaseerde BIM-omgeving. Dit met het bijkomend doel om BIM modellen te initiëren vanuit enkel informatie, alvorens de graad van modelleer complexiteit verhoogd wordt. Als *proof of concept* is een prototype ontwikkeld[2]. Hiermee kan een gebruiker een simpel BIM-model starten op basis van de *Building Typology Ontology (BOT)* (Rasmussen et al. 2020). Externe micro front-ends kunnen dan worden geïmplementeerd via een *store* component van de applicatie om de UI uit te breiden.

[2]De code is te vinden op `https://github.com/RobbeMic/thesis_Digital_Environment`

De omgeving kan externe functionaliteiten bevatten die hier oorspronkelijk geen deel van uitmaakten; zo kunnen gespecialiseerde workflows ontstaan. De structuur van dit prototype is gebaseerd op de principes van ConSolid (Werbrouck et al. 2022 [3]) en de mogelijkheid tot *many-to-many* verrijking.

Overzicht

In **hoofdstuk 1** wordt de huidige situatie van BIM-applicaties geschetst, met nadruk op de structuur van het datamodel. Dit leidt tot het onderzoeksveld van deze thesis, waarin de onderzoeksvragen worden uiteengezet, gevolgd door de structuur van de overige hoofdstukken.

Vervolgens wordt in **hoofdstuk 2** het huidige discours omtrent het creëren van BIM-modellen voor de bestaande context besproken. Daarbij worden verschillende methodes overwogen om het onderwerp BIM voor renovaties te benaderen. Eerst wordt de techniek van het scannen van een gebouw om een 3D-model te maken onder de loep genomen. Daarna wordt het BIM4Ren-onderzoek besproken, waarbij de aandacht vooral uitgaat naar de pilots met betrekking tot het opzetten van een BIM-model. Daarnaast wordt het voorgestelde schema voor het combineren van deze pilots in een collaboratief platform toegelicht. Tenslotte wordt het proces van BIMificatie besproken in de context van het initialiseren van een ”information-first” BIM-model.

Als inleiding op het onderwerp worden in **hoofdstuk 3** de voor deze thesis relevante concepten omtrent Linked Data besproken. Het hoofdstuk begint met een inleiding in de basisprincipes van Linked Data en het Semantisch Web. Dit omvat *Resource Description Framework (RDF)* grafen en *SPARQL Protocol and RDF Query Language (SPARQL)* queries voor het representeren en opvragen van gegevens. Vervolgens wordt dit in de context van de AEC industrie geplaatst met specifieke Linked Data implementaties. Tenslotte worden aanvullende kaders in de context van webdesign toegelicht, zoals het Solid ecosysteem en micro front-ends.

Hoofdstuk 4 bespreekt de methodiek dat wordt voorgesteld door het theoretische kader. Beginnende met het creëren van een BIM-model voor bestaande gebouwen, waarvoor het BIMification-proces wordt bekeken en een aanpassing wordt voorgesteld. De structuur voor een uitbreidbare BIM omgeving wordt geschetst, uitgaande van een aanpassing van de collaboratie schema voorgesteld in het BIM4Ren onderzoek. Vervolgens wordt de resulterende graaf structuur voor een Linked Data-gebaseerd BIM model besproken. Tenslotte wordt een herziening van de ontologie van Mifesto voorgesteld. De herziening richt zich op het bieden van een toegankelijk en praktisch kader voor het integreren van micro front-ends in een BIM omgeving.

Het prototype dient als *proof of concept* voor het theoretische kader en wordt beschreven in **hoofdstuk 5**. Elke component wordt kort besproken in de context van de uitbreidbare BIM omgeving in plaats van de functionaliteiten zelf. De nadruk ligt op de interactie tussen de gebruiker, de hoofdapplicatie en de micro front-ends, aangezien het concept van de *store* en de Mifesto ontologie worden getest.

Uiteindelijk wordt in **hoofdstuk 6** het prototype geëvalueerd in de context van een uitbreidbare BIM-omgeving. Dit leidt tot een reflectie op de hele methodiek van het theoretisch kader. In deze reflectie komen de beperkingen van het onderzoeksveld van de thesis naar voren, wat leidt tot een sectie over mogelijk toekomstig werk. Tot slot wordt de thesis afgesloten met een kort antwoord op de in de inleiding gestelde onderzoeksvragen.

Conclusie

Het voorgestelde theoretische kader bestaat uit drie opeenvolgende componenten: het BIMificatieproces, de uitbreidbare BIM-omgeving en de Mifesto ontologie. Ten eerste biedt het BIMificatieproces een methode om de complexiteit van een BIM-model geleidelijk te vergroten. Hoewel het gebruik van Linked Data niet noodzakelijk is voor dit proces, is het wel een haalbare manier om een informatie-gericht BIM-model te creëren zonder een 3D model. De uitbreidbare BIM omgeving is dan in staat om de gegevens die beschreven zijn in een RDF graaf weer te geven. Verder kan de omgeving worden uitgebreid met externe functionaliteiten die worden geleverd door micro front-ends; geïmplementeerd met behulp van het *application store* concept. Tenslotte biedt de herziening van de ontologie van Mifesto een gestructureerde aanpak voor het beschrijven van de metadata die nodig is voor zowel navigatie als implementatie van de micro front-ends. Door de aard van Linked Data kunnen de manifesten van de micro front-ends worden uitgebreid met extra definities. Het gehele kader illustreert een mogelijke oplossing voor *many-to-many* verrijking binnen het domein van BIM software.

Het ontwikkelde prototype benadrukt dat het voorgestelde theoretische kader kan worden gebruikt als basis voor een open BIM omgeving, waarin elke toepassing elk BIM model kan bewerken via elke externe dienst.

Referenties

[1] Mads Holten Rasmussen et al. "BOT: The building topology ontology of the W3C linked building data group". en. In: *Semantic Web* 12.1 (Nov. 2020). Ed. by Krzysztof Janowicz, pp. 143–161. ISSN: 22104968, 15700844. DOI: 10.3233/SW-200385. URL: https://www.medra.org/servlet/aliasResolver?alias=iospress&doi=10.3233/SW-200385 (visited on 01/12/2023).

[2] Raimar J. Scherer and Peter Katranuschkov. "BIMification: How to create and use BIM for retrofitting". en. In: *Advanced Engineering Informatics* 38 (Oct. 2018), pp. 54–66. ISSN: 14740346. DOI: 10.1016/j.aei.2018.05.007. URL: https://linkinghub.elsevier.com/retrieve/pii/S1474034617305372 (visited on 01/12/2023).

[3] Jeroen Werbrouck et al. "ConSolid: a Federated Ecosystem for Heterogeneous Multi-Stakeholder Projects". (in press). 2023.

[4] Jeroen Werbrouck et al. "Practical Semantic Enrichment of AEC Multi Models with Dynamic Micro Frontend Configurations". en. Manuscript under review. Dec. 2022.

Contents

List of Figures

Listings

List of abbreviations

Chapter 1

Introduction

1.1 Current Situation

Over the last two decades, the construction industry has been revolutionised by the development of Building Information Management (BIM) models. Unlike traditional Computer Aided Design (CAD) software, this approach uses a central data model that incorporates information from multiple stakeholders, such as architects, engineers, electricians, etc. This results in better information handling as these different disciplines can interact more easily during design, resulting in fewer errors on site.

1.1.1 BIM technologies

Current BIMapplications focus primarily on 3D modelling and incrementally adding information to the modelled geometry as the project evolves; e.g. BrycsCAD BIM or BlenderBIM both focus on first creating a near-complete 3D model and then semantically enriching it. Alternatively some applications focus on predetermined classes, each with their distinct semantic properties and geometrical representations; e.g. in Revit the user first decides to add a wall, which is then implemented with fixed properties and a given geometry. These approaches work well for new buildings where the environment or urban context is of little importance. However, when there is an existing building or the information about the surroundings is significant, e.g. roadworks, the ”model-first” approach becomes cumbersome if these BIMmodels weren’t created nor properly maintained in the original construction project. In such cases, most current BIMapplications either lack the necessary support or require time-consuming ”after-the-fact” modelling of the existing situation.

In addition, most current BIM applications use proprietary or ”closed” file formats that can only be interpreted by the specific application or digital environment. This undermines the original intention of using a central data model from which the various stakeholders can draw and communicate their information. To remedy this, the buildingSMART initiative has created an environment agnostic file format, the Industry Foundation Classes (ifc) format, to provide a standardised description of building information and elements [8]. Although the format has proven it’s value in contemporary practice, its implementation is often lackluster in most CAD software, as a lot of valuable information is lost in the conversion process from or to the ifc format. Furthermore, the monolithic nature of the ifc schema struggles to provide an international implementation as the needs and requirements of building projects vary widely between countries and disciplines [7].

1.1.2 Multi-models

This aforementioned variety of both requirements and needs in the Architecture, Engineering and Construction (AEC) industry creates a need for a more modular data model. When considering the different stakeholders that contribute in a single construction project, the variety in information generated becomes too vast for a single monolithic data model. Hence the existence of multiple specialised applications even within the digital environment of a single software vendor e.g. the Autodesk suite with Revit, Civil 3D, Fusion 360, etc.; each with their unique file format.

To remedy this, the current research discourse is looking into data formats in which BIM models are built up from multiple sources and/or BIMmodels. In a so called multi-model, the different stakeholders would each contribute their relevant information to a single BIMmodel or digital twin; preferably using a discipline specific application. This in turn would create a more complete BIMmodel that is up-to-date with the latest changes in the project, and is thus extensible throughout a building’s lifecycle. Furthermore does this modular BIMapproach allow each stakeholder to retain intellectual property over their contributions to the construction project as the building blocks from which the digital twin gets its information are created, provided and maintained by the stakeholders themselves. [7]

1.1.3 Towards linked data

As described above, there is a growing need for a modular data model that can adapt and extend its use as a construction project evolves. One possibility for a more data-driven approach to BIM, moving away from ”model-first” file formats, can be achieved through the adoption of linked data technologies.

Linked data is a concept derived from the development of the Semantic Web. Rather than linking files, as is the case with the current World Wide Web (Web), the semantic web aims to link information or data directly. This is an approach where the data is independent of the application and therefore inherently allows for greater interoperability. Especially for the AEC industry, notorious for its closed data formats and inefficient information transfers, a linked data workflow seems promising. As such, the adoption of the Semantic Web in construction is the subject of current (2023) academic research on BIMtechnologies [20].

1.1.4 Renovation management

This thesis focuses on one of the sectors within the AEC industry that is lagging furthest behind in the digitisation process, namely renovation and retrofitting practices. This lack is due to the inherent ”problem” of the existing context [28], where a lot of information and complexity is already present. Therefore, during the analysis and design process of an existing building, a lot of information is gathered and considered before a plan is produced. However, contemporary design practices adopt a rather ”primitive” approach to dealing with this data; written documents supported by photographs, copies and drawings.

More digitised workflows do exist, but they involve intensive scanning methods followed by a great deal of effort to convert the captured data into semantically enriched BIM objects. In this process, there is an immense upfront cost before any substantive analysis or design can actually begin. This sequence is encouraged by the prominent BIM tools.

1.2 Scope of the thesis

In the following section I will discuss the context from which this thesis approaches the problems raised previously and how this leads to the research question. In addition, the resulting prototype is presented and the structure of the thesis is outlined.

1.2.1 Issue at hand

As outlined above, the current AEC industry is struggling with further digitisation and the automation of tedious tasks. This is particularly true in the context of renovations, where the barrier to entry to a BIM-based workflow is rather high. However, when considering the process that a renovation project goes through, it should be even more amenable to a data-first approach than new construction.

Because there is already an existing building, there is a great deal of analysis that drives the design process, i.e. there is a substantial pool of information that is generated before any architectural plans or 3D models are created. In addition, each stakeholder has different needs for this information and produces different outputs about the building in question. To leverage this diverse data-pool, adopting a modular BIM format seems inevitable as it allows to adapt and extend itself throughout the building's life-cycle. Such an extensible BIM model creates opportunities for the adoption of relatively new technological advances such as linked data to create a common data model; the digital twin.

However, as the BIM model itself becomes modular, it also becomes uncertain about the included information. Thus the digital environment through which the stakeholders access the digital twin should be able to adapt itself to the needs of both the stakeholder and the BIM model. This would allow for immense specialisation of tools within a specific discipline while still enabling to include any information from any stakeholder.

1.2.2 Research question

The current research discourse on linked data in AEC has already produced a multitude of frameworks; e.g the DOT ontology by Hamdam et al. [14] or the framework for linked data-based heritage BIM by Bonduel M. [5]. Which, when combined cover most aspects of construction, renovation, restoration, etc. Each of these frameworks generates its own, often very small, definitions, tools and applications that could be implemented in a larger or more general application. However, these frameworks rarely provide the necessary metadata, or information about the framework itself, to allow implementation into general applications. This information about these frameworks is needed to link, mix and reference multiple frameworks, tools or applications in the same building project. Because of the heterogeneous nature of the data and the almost "limitless" disciplines that could consume the data, current monolithic applications struggle to provide a common data model. It is also this complexity which raises the need for an extensible BIMenvironment. Hence the importance of a dynamic user interface (UI) built up from multiple smaller components; resulting in a structure that can support digital twins which are themselves build up from multiple smaller BIM models.

Therefore, the research question of this thesis is twofold:

- ”In the context of a renovation project, how do you create a BIMmodel without a 3D model?”
- ”What metadata is required for enabling the implementation of smaller services into a Linked Data-based BIMenvironment?”

At the time of writing, such applications do not exist; an application aimed at creating a linked data based BIMenvironment for a specific project. As such, the ambition of this thesis is to provide a theoretical framework for the development of a micro front-end based BIMenvironment, focusing on a proposed ontology for describing these micro front-ends and their implementation.

1.2.3 Prototype

As a proof of concept, a prototype UI was developed. The prototype allows the creation of a linked data BIMworkflow starting from information without a direct 3D representation. The prototype allows the creation of an empty linked data BIMdashboard to which several micro front-ends can be added to create BIMobjects. These front-end modules are loaded from outside the UI and configured using the required metadata mentioned above. In the prototype the examples are focused on adding damage instances and conceptual objects as these are typically types of information which can’t be included in common BIMapplications. In addition, after the information is created, 3D geometry can be appended to specific elements of the BIM model.

1.2.4 Structure of the thesis

In the following chapters, I first consider the challenges and opportunities of transforming real-world information into a digital environment (Chapter 2). This chapter outlines a preferred structure for initialising an ”information first” BIMmodel. This is followed by an introduction to linked data in the AEC industry (Chapter 3). The methodology of the theoretical framework developed is then presented as a possible answer to the research question (Chapter 4). This theoretical framework is then tested in a prototype (Chapter 5). Finally, the prototype is evaluated in regards of the resulting BIMstructure, the theoretical framework is reflected upon in relation to the initial problem statement above, and possible future work is highlighted (Chapter 6).

Chapter 2

The ”BIMification” of the existing context

As mentioned earlier, there is a wide variety of challenges within the digitalisation process of the AEC industry. This complexity stems from the wide variety of buildings themselves, before the multitude of stakeholders are taken into account. This complexity leads to scarce opportunities for abstraction of information, as at some point in the building’s lifecycle this information will be required by a previously unknown stakeholder. This creates the need for a digital environment that can be adapted and extended to handle this variety of both desired assets and actions.

In the following chapter I discuss possible options for transforming information about the physical context into digital data. First, I review the now almost common practice of ”scan to BIM”, which follows a traditional model-first approach. Secondly, I discuss the results of the BIM4Ren research project, which focused specifically on the development of BIMtools for the renovation industry. Next, I give an overview of a ”data-first” approach, where the BIMmodel is built incrementally, and finally I conclude with a proposed workflow for initiating an information model about the existing context.

2.1 Scan to BIM

”Scan to BIM” is the process of using 3D scanning or remote sensing to create a 3D model that is then enriched to create a full BIMmodel. In most cases, laser scanning, photogrammetry or a combination of both is used to create a virtual point cloud, a large number of individual points, preferably coloured, that can be interpreted as larger geometry. This point-cloud is then remodelled into geometric shapes representing walls, doors, windows, etc. In a third step, the semantic information is added to these 3D geometries to create a BIMmodel.

The main advantage of this approach is that it produces relatively accurate measurements of the buildings in their full 3-dimensional complexity, while generating the 3D representation is quite fast. However, this approach has some major drawbacks, the main one being that this method only generates information about the visible surfaces. It's in the nature of a non-destructive scanning method that the scanner only generates information about the surfaces of the building elements, primarily their 3D position and the colour observed. As such, a scan-to-BIM process can't generate information about the underlying structure or materials.

Secondly, remote sensing produces a point-cloud that needs to be converted into actual 3D objects. This is usually done by actively remodelling the building to match the point-cloud measurements, which is a lot of work and therefore a large upfront cost. This process could be more automated and is the subject of current research in computer science, but such algorithms have a hard time detecting edges and gaps in either the building or the scan.

Ultimately, the problem remains that a renovation project must be sufficiently developed to know exactly which parts of the building or context need to be scanned and to what Level of Development (LOD). This means that either the entire existing building has to be scanned at a fairly significant LOD at the start of the project, or a preliminary analysis has to be carried out using traditional non-BIM workflows. This results in information that has to be migrated into the resulting BIMmodel.

2.2 BIM4Ren

The BIM4Ren project is a research collaboration through development in which 23 partners from 10 European countries have developed pilot applications and tools specifically for BIMin existing buildings; with a focus on energy renovation. In this project the pilots were divided into 5 categories; Strategic Planning, Data Acquisition, Data Driven Design, Data Management and Support [3]. In the following I will focus on the Data Acquisition and Support aspect, which concerns the initialisation of the BIMmodel and the interaction between the different stakeholders.

First of all, one of the pilots, "Plans2BIM", tried to directly address the problem of initially creating the BIMmodel of the existing building by elevating existing plans and sections. This produces a BIMmodel of relative quality very quickly, but it can't deal with the complexity and imperfections of existing buildings, which is essentially what makes a 3D model more interesting than a plan [12]. Other case studies and pilots assumed that a BIMmodel was available, either from a previous building project or created from a 3D scan.

In addition, the BIM4Ren project aimed to deliver a One Stop Access Platform (OSAP) outlining a possible structure for collaboration between the specific pilot tools, the BIMmodel and the different stakeholders [19]. The OSAP is intended as a platform from which users can interact with the multitude of applications and tools, external catalogues and the digital twin. In the proposed structure, BIM4Ren makes use of KROQI, a tool developed by the French authorities in the context of the Digital Transition Plan in the Building and is a platform aimed at collaboration in the construction sector and communication with external services[18].

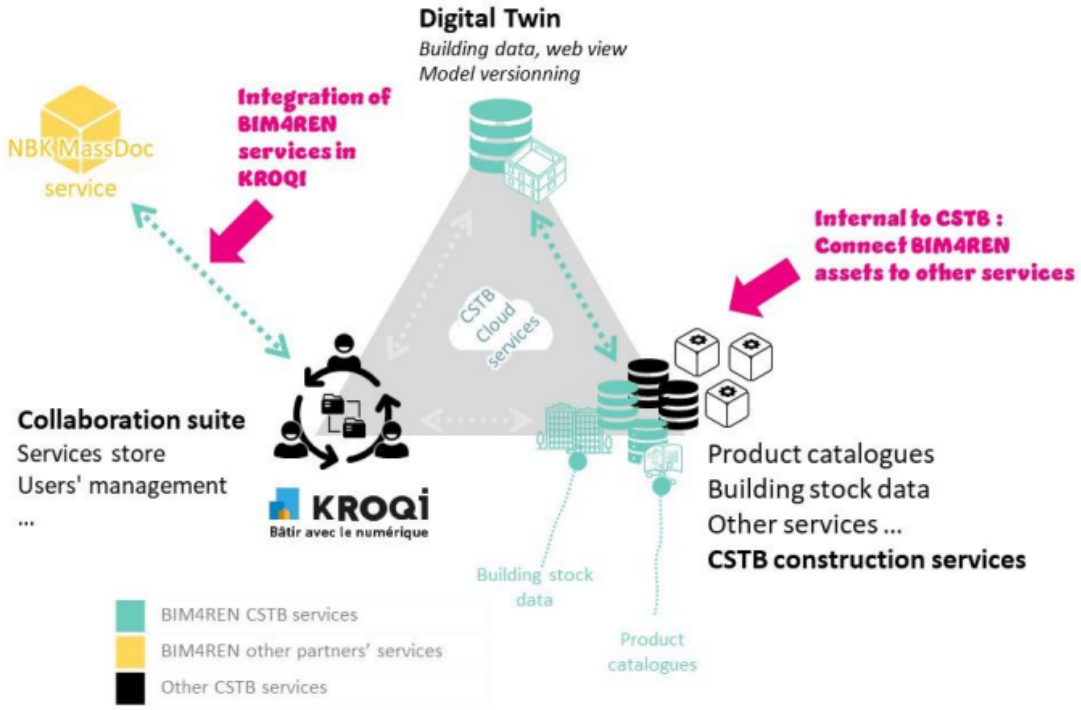

Figure 2.1: Potential collaboration schema as proposed by the BIM4Ren reproduction plan

The proposed schema (Figure 2.1) starts with a collaboration platform that acts as an interface through which users or stakeholders interact with the building project. From this interface, connections are made between the ”digital twin” and external services such as product catalogues, building stock data, etc. In order to allow the user to use these external services without requiring technical knowledge, the OSAP provides the interaction between the digital twin and the external service. However, this proposed structure for an extensible or connected BIMplatform is rather proprietary and aims to promote the developed services as a market-ready ecosystem [19]. To illustrate, this schema highlights that the KROQI platform allows the integration of the specific services developed by the BIM4Ren project, which is essentially a list of external services.

2.3 The process of ”BIMification”

BIMification, as described by Scherer R. & Katranuschkov P. (2018) [26], is a structured approach to creating a ”building information model” of an existing building. The whole process consists of three main steps to create a solid information base, as shown below (Figure 2.2). The BIMification process itself consists of two phases; ”Anamnesis” and ”Diagnosis”. Anamnesis is traditionally a medical term and is used in the context of renovations as an observation of the building, without drawing any definite conclusions, and as a recollection of information. The diagnosis then consists of the conclusions and implications drawn from the observations; more specifically, this step leads to a BIMmodel of the building, including all the gaps, inconsistencies, damages, etc. Finally, BIMification is followed by ”BIM-based retrofit design” or, more traditionally, the design phase of a building project, which consists of ”Therapy”. In other words, the Therapy is the action or design proposal that follows from the findings of the Diagnosis.

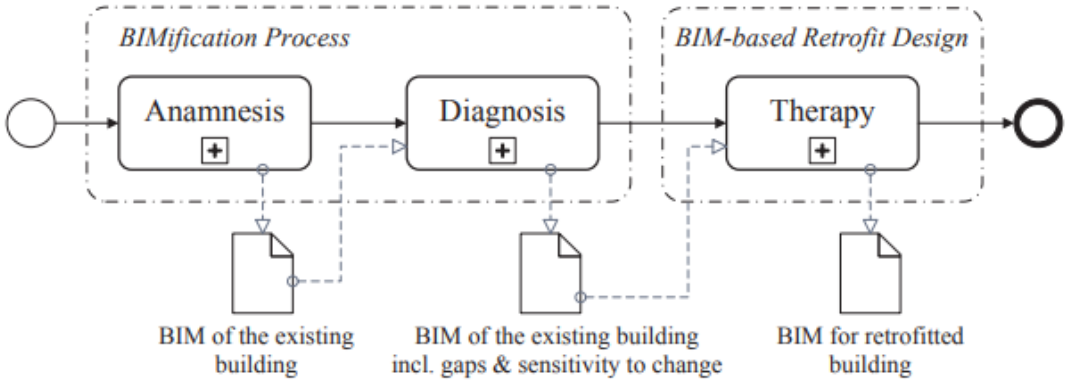

Figure 2.2: 3-stage BIMification process as proposed by Scherer R. & Katranuschkov P. (2018)

The method proposed by Scherer R. & Katranuschkov P. (2018) [26], within the anamnesis phase, starts with the creation of a full geometric BIMification of the whole building; the initial problem of high upfront conversion effort and cost remains. However, the proposed framework allows a shift from a ”model-first” to an ”information-first” BIMapproach, as the main workflow is concerned with gathering information in a structured and holistic manner. This also shifts the perception of BIM from ”Building Information Modelling” to ”Building Information Management”, allowing a greater focus on collaboration through a shared data model, rather than focusing primarily on accurate 3D representation.

2.4 Conclusions

To answer the question of how to initiate a BIM model of an existing context, the first step is to shift the focus from BIM as ”Building Information *Modelling*” to ”Building Information *Management*”. By moving away from traditional model-first approaches such as scan-to-BIM, a renovation project is allowed to mature first and find out which parts of the existing building are critical; these could then be modelled in greater detail than non-critical areas.

In an information-first BIM approach, there is a need for an adaptable digital environment that can meet different needs throughout the project, while still drawing from the same data source. In such an adaptive environment, the user interacts with both the digital twin (or project database) and multiple external services. This in turn creates a BIM application that can be adapted to the user’s needs and expertise. In this way, the UI acts as both a dashboard and an editor for the BIM model.

As mentioned earlier, the ”digital twin” serves as a project database from which all stakeholders can draw their information. However, when dealing with an existing building or context, there is already a vast amount of non-digital data, namely the current state of the building. For this reason, there is a need for a ’BIMification’ process that captures the relevant information in the BIM model. This process consists of three stages, Anamnesis, Diagnosis and Therapy, through which the complexity of the BIM model is gradually increased. Ultimately, this structure, combined with an adaptive BIM environment, should allow the existing building to be analysed and the project objective to be identified before detailed plans and 3D models are produced.

In the remainder of this thesis I will focus on the implementation of external services in such an adaptive BIM environment through the case of initialising a BIM model as a database without the need for a 3D representation. In the following chapter I will discuss a possible technological approach to creating an extensible BIM model or an adaptive BIM environment.

Chapter 3

Linked Data

For creating modular BIM models composed of multiple sources, one possible solution is to use Linked Data. This approach allows the direct linkage of data or information, regardless of the applications that process them. Therefore, this thesis will utilise Linked Data to create linked BIM models and an extensible digital environment for these models.

In this chapter, I will first give a brief introduction to Linked Data and its core concepts. I then discuss how Linked Data could be applied within the AEC context, and how Linked Data can be used in a BIM workflow. Then, I highlight additional ontologies or frameworks relevant to the scope of this thesis. Finally I conclude with a preferred workflow for adopting Linked Data in BIM. This provides a knowledge base for the following chapter on my proposed methodology for creating a BIM environment that is adaptable and extensible.

As a disclaimer, this chapter doesn't provide a complete and comprehensive introduction to all the concepts associated with Linked Data and the Semantic Web. However, it attempts to provide a foundation and background to the concepts that are relevant to the scope of this thesis.

3.1 Introduction to Linked Data and open standards

From the beginning, the World Wide Web (Web), as developed by Tim Berners-Lee in 1989, was intended to serve as an information space, designed to be both useful for human-to-human communication and consumable by machines. However, earlier versions of the Web and the information on it were primarily designed for human consumption or interaction. Although some data may be clearly structured in a database, the structure of the data is not apparent to a machine browsing the Web. [1] The latest development of the Web, known as Web 3.0, has focused on providing data on the Web in a machine-readable way, while linking it to other data on the Web; hence the name Semantic Web.

Unlike the traditional ”web of documents” written in HTML, the Semantic Web uses the Resource Description Framework (RDF) language, which allows data to reference each other to provide meaning or reasoning on that data. In this concept, Uniform Resource Identifier (URI)’s are used to identify any kind of object or concept, and are thus used as it’s name. This is in contrast to the traditional Web which uses Uniform Resource Locators (URLs) to just locate documents. Now, by using Hypertext Transfer Protocol (HTTP) URI’s, people can look up these names or useful information, provided that they are structured according to World Wide Web Consortium (W3C) standards. Eventually, the data itself should be linked to external resources, other URI’s, to create a web of data [2]. This is described in more detail in Tim Berner-Lee’s five-star open data schema.

As indicated above, machine-readable web data is standardised in the Resource Description Framework (RDF), a framework for representing data on the Web. In RDF, the core structure is a set of triples, each consisting of a subject, object and predicate. Such a set of triples is then called a graph, and can be visualised as a directed arc diagram (Figure 3.1), with each triple represented as a ”node-arc-node”. These graphs are inherently unordered and unindexed, allowing the data model to be extensible; new edges and nodes can be added at any time. [9]

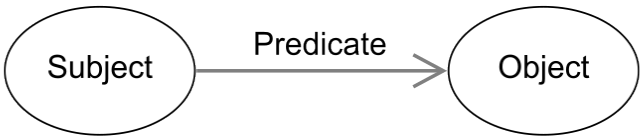

Figure 3.1: Basic RDF triple comprising (subject, predicate, object), [9]

A RDF graph can have three types of nodes, Internationalized Resource Identifiers (IRIs), literal values and blank nodes, while arcs are always IRI’s. IRI’s were introduced instead of just URI’s as the latter only allow a certain set of American Standard Code for Information Interchange (ASCII) characters. These IRI’s are assembled from an absolute scheme, a host, a path, and may contain a fragment; all pointing to another resource. However, with only IRI’s, an RDF graph would only reference other resources without any concrete human-readable information; hence the inclusion of literal values such as strings of text and numbers. Ideally these literal values are data-typed, indicating what kind of value the literal is, and in the case of text, language-tagged. Finally, blank nodes have been included to denote unknown values. [11]

Ontologies are used to give meaning, or definitions, to these nodes and arcs. These ontologies are a collection of IRI’s with the intention of using them in other RDF graphs.

Usually they start with a namespace IRI, followed by a fragment to denote a specific object [10]. The main standards for defining these semantics are either RDF Schema (RDFS) or Web Ontology Language (OWL), though many more specific ontologies exist and are built using these standards.

In order to store these RDF graphs as files in a human- and machine-friendly way, there are several syntaxes; e.g. RDF/XML, Turtle, N-Triples, N3, RDF/JSON, etc. [32] For the purpose of this thesis, I will use the Terse RDF Triple Language (Turtle) as it can be written in a compact and natural text form. [22]

The most common abbreviation in Turtle is the inclusion of the predicate `<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>`, denoted as "a", which defines the type of an object in the graph. Additional predicates on the same object are separated by a semicolon, while the end of an object description is marked by a period. In addition, the Turtle language allows the use of prefixes, denoted by "@prefix", and the use of local IRI's, which contain only the fragment. Below (Listing 3.1) is an example Turtle file describing a door with a defect, which in turn was covered in external documentation.

```
@prefix inst: <http://example.org/>.
@prefix beo: <https://pi.pauwel.be/voc/buildingelement#>.
@prefix dot: <https://w3id.org/dot#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.

<#frontDoor> a beo:Door, bot:Element;
    rdfs:label "front door";
    dot:hasDamage <#damageInstance>.

<#damageInstance> a dot:DamageElement, dot:Defect;
    dot:coveredInDocumentation inst:Example_Documentation.
```

Listing 3.1: Example Turtle (ttl) file

To retrieve data from or rather query existing RDF graphs, I will use SPARQL Protocol and RDF Query Language (SPARQL) as recommended by the W3C in the context of the Semantic Web. SPARQL is used to query required and optional data in a graph along with their possible conjunction or disjunction [16].

A SPARQL query consists of a set of triples, just as in RDF graphs, but each of the object, predicate or subject can be a variable; denoted by a question mark and the variable name. The result is then a solution sequence corresponding to data that matches the graph pattern of the query. Consequently, the result sequence may be empty, contain one result, or contain multiple results.

There are several types of SPARQL queries, each designed to either fetch, add, update or remove information from the graph. The most commonly used queries are SELECT, ASK, DESCRIBE, INSERT and REMOVE. A SELECT form returns the variables bound in a query pattern match directly. Conversely, a DESCRIBE form is used to return a RDF graph containing RDF data about the resource. The ASK form returns a boolean, true or false, whether the query pattern matches the data. [16] In addition, the INSERT and REMOVE update operators are used to add or remove data from the specified RDF graph; they differ from the core query forms in that they require specific authorisation to modify the resources. [13]

Below (Listing 3.2) is an example of a SELECT query where the returned variables are defined in the SELECT query and the required graph pattern is defined in the WHERE clause. As with Turtle, SPARQL allows the definition of prefixes to improve human readability.

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dcat: <http://www.w3.org/ns/dcat>
PREFIX dot: <https://w3id.org/dot#>

SELECT ?object ?objectName ?damage
WHERE {
    ?project a dcat:Catalog;
        dcat:hasPart ?object.

    ?object dot:hasDamage ?damage;
        rdfs:label ?objectName.
}
```

Listing 3.2: Example SPARQL SELECT query

The resulting sequence is shown below (Table 3.1) in a table containing the selected variables and their returned bindings. In this example, the query returns all objects in the project dataset that are linked to a dot:Damage instance. It is important to note that only the variables specified in the WHERE clause are returned.

object	objectName	damage
inst:frontDoor	”front door”	inst:damageInstance_01
inst:cornice_01	”cornice of the facade”	inst:damageInstance_23
inst:door_f1h6	”door livingroom - hall”	inst:damageInstance_56

Table 3.1: Example of a returned query bindings sequence

3.2 BIM through Linked Data

In the context of the AEC industry, several relevant ontologies and frameworks have been developed with the aim of creating a linked BIMformat. Initially, the ifcOWL ontology was developed as a RDF equivalent to the original ifc EXPRESS schema. Although it is the recommended ontology endorsed by buildingSMART, it suffers from the same monolithic structure and complexity. [6]

As concluded earlier in section 2.4, there is a need for a format for Building Information Management (BIM) that can adapt and extend to the needs and requirements of the construction project. To provide such an environment using linked data, the Building Typology Ontology (BOT) has been developed by the W3C Linked Building Data (LBD) Community Group, specifically for combining other ontologies (Rasmussen et al. 2020 [24]).

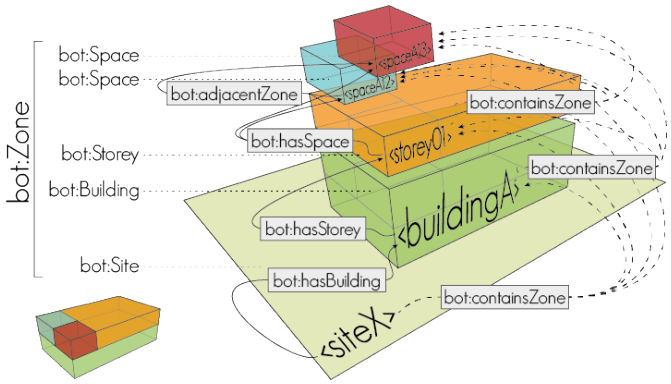

Figure 3.2: Representation of instances and their relations in the BOT ontology (Rasmussen et al., 2020 [24])

The aim of BOT is to represent interlinked information by explicitly defining necessary relations between building elements. Within the ontology there is a distinction between elements and zones. Elements are tangible objects such as walls, windows, etc; whereas zones are typically areas or spaces. A third main class is defined as an interface, a conceptual interaction between two items, at least one of which is either an element or a zone; e.g. a door, which is an element, connects two rooms and thus all three share the same interface. As shown above (Figure 3.2), BOT contains four hierarchical subclasses of zones: Site, Building, Floor and Room. Finally, each zone or element can be assigned a 3D model. Notably, in the proposed schema, the 3D geometry is optional and included after the creation of the data instance. You may notice that it doesn't contain any classes for defining geometry. This is intentional, as inclusion of geometry descriptions is a whole research topic in itself, and the RDF data model is a relatively inefficient implementation for linked lists, and thus for 3D geometry [7].

This means that the modular structure of BOT lends itself to being incorporated into an information-first BIM approach. Thus allowing the building to be described and data to be collected before any 3D modelling or representation is in place.

Moreover, as one of the fundamental principles of the Semantic Web, this Linked Data approach allows a single BIM model or digital twin to be built from multiple sources. More specifically, when a database such as a digital twin is described in a RDF graph, linking to external resources is as natural as internal linking. This, in particular, applies when the linked resources are also described as a graph and exposed on an IRI, making it possible to query this link and thus consider these sources as part of the same digital twin. Finally, separating the datasets into multiple documents allows for intellectual property integrity, as each stakeholder maintains their own documents, and thus their contributions. For example, the inspector could write his report on the damage and its structural consequences after a damage inspection and analysis. This report and the damage it contains are then linked to the relevant elements. This in turn serves as a reference for a potential structural engineer to determine the building's structural integrity, without requesting various documents from multiple stakeholders.

There is also a trend towards minimal but highly specialised data models to be implemented as extensions to larger datasets. This allows for highly customised workflows tailored to specialised disciplines. This is a prerequisite for moving to a fully federated data model, as there is a reason for the current diversity of specialised software: each discipline or stakeholder has a different view of the same building element. [31] However, in order to federate these specialised data graphs, there is a need for an overarching data model to which each contributor can link their information. This is achieved by implementing a "linkset object", which represents the abstract concept of an element, as shown below (Figure 3.3). The relevant information and representations are then linked to the concept, rather than to the building elements themselves. [30] This once again reinforces the ability to do more conceptual and analytical design before modelling a 3D representation.

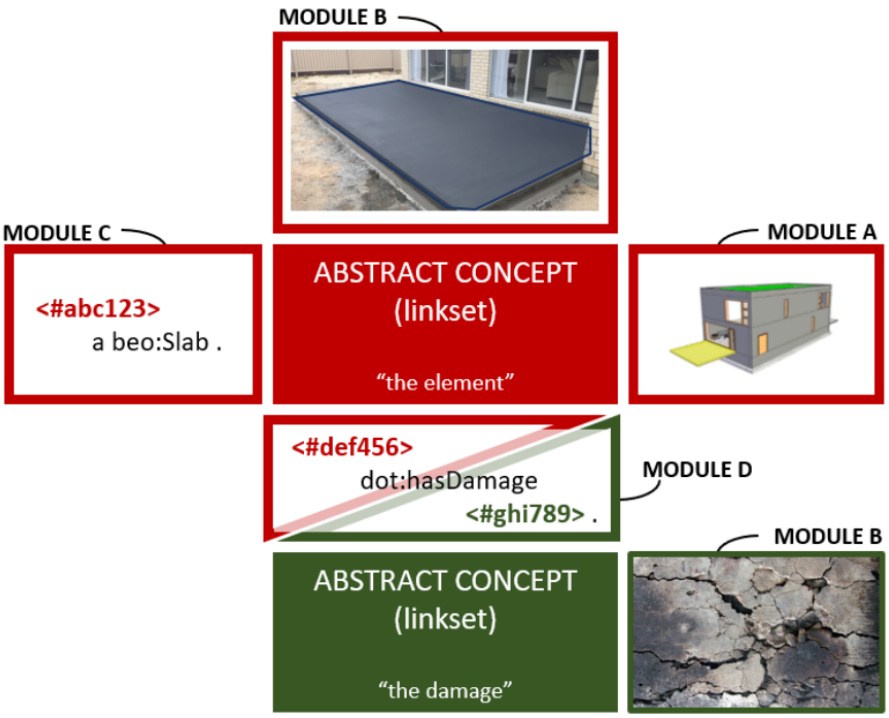

Figure 3.3: Visualisation of linkset objects linking together multiple semantics of conceptually the same item. (Werbrouck et al., 2022 [31])

An example of such a minimal ontology is Damage Topology Ontology (DOT), intended for practical damage assessment with the modular BOT ontology. However, instead of requiring specific classes or properties to be present in the dataset, DOT can be used as a stand-alone ontology. The ontology is minimal in the sense that it provides a core data model for representing damage, which can be further extended with specialised ontologies, e.g. an ontology for damage degradation. In this vein, the geometric representations are separated from the topology, allowing the damage to be analysed first [15]. Due to it's ability to be used as a stand-alone ontology that allows to first create information, DOT is used throughout this thesis as an example ontology in an information-first BIM approach in the context of renovation.

3.3 Additional frameworks

Alongside Linked Data developments in the AEC industry, there are several other relevant frameworks not specifically developed for BIM. This section discusses the Solid platform and it's utility for federated BIM models. This is followed by the relevance of micro front-ends in creating an extensible BIM environment.

Solid is a platform for storing users' data independently of the application that creates or consumes it. In Solid, users store their data in a web-accessible personal online datastore (Pod); which can be stored on any Pod provider. [25] This puts the users in control of their data. It is maintained from a single source of truth, their Pod, rather than having the same information in different states of flux on multiple platforms. This approach results in decentralised web applications that access information from multiple Pods [27]; as shown below (Figure 3.4).

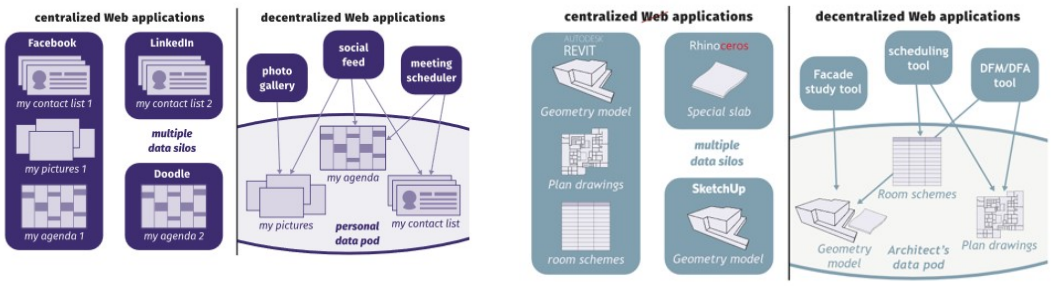

Figure 3.4: Web applications in the Solid ecosystem (left) [27] applied to the AEC industry (right) [23] (as displayed by Werbrouck et al., 2022 [30])

If we apply this concept to the AEC industry, we find that the Solid infrastructure aligns with the benefits highlighted earlier (Section 3.2) of adopting a Linked Data approach to BIM. In particular, a single federated BIM model made up of smaller data sets provided by the various stakeholders, while retaining the highly specialised workflows required for each discipline. [30] To facilitate such a linked BIM model and to provide the necessary metadata about the construction project, the ConSolid vocabulary has been developed by Werbrouck et al. [29]. The vocabulary allows a project to be defined as a dataset containing multiple sub-datasets representing the contributions of each stakeholder, which are defined as partial-projects. Within the overarching project dataset, aggregators or linkset concepts are included to which each stakeholder can link their information. This results in the following data structure (Figure 3.5).

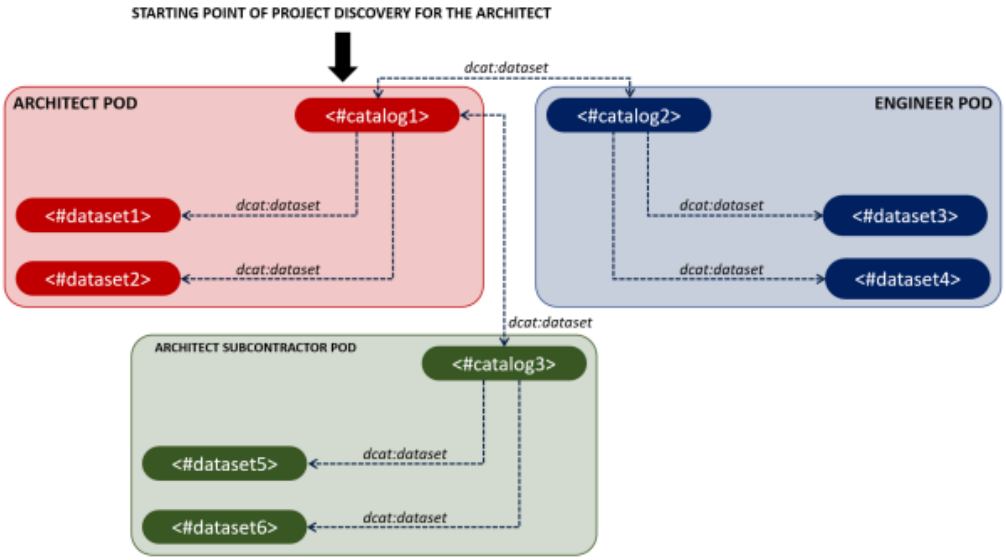

Figure 3.5: Data structure of a federated BIM model in the ConSolid ecosystem (Werbrouck et al., 2022 [30])

In the context of federated BIM models, interaction with these databases becomes increasingly important. Without interaction between people and servers, the exposed data is irrelevant. Following the trend of making applications more specialised and modular, interactions with a federated dataset shift towards a collection of micro-services. In the Web context, these micro-services become micro front-ends, providing an interface through which the aforementioned user interacts with the dataset. An overview of the benefits and issues of using micro front-ends can be found in Peltonen et al. [21]. In a nutshell, an adaptive and extensible BIM model, such as a ConSolid project, poses a major challenge for monolithic interfaces. Due to the uncertainty and indefiniteness of the data, the digital environment itself must be able to adapt and extend. [31] The importance of an extensible BIM environment is confirmed in the BIM4Ren Replication Plan [19] and has already been discussed in section 2.2.

3.4 Application design

In order to implement these micro front-ends into the aforementioned BIM environment and thus make it extensible, there are multiple options to consider. This section provides a brief introduction to implementing external services into a larger application or environment. First the usage of plugins and APIs is discussed. Then module federation is introduced together with it's relationship to micro front-ends. These concepts move away from Linked Data but are relevant to a practical implementation of extensible BIM applications in the Semantic Web.

The first option is using a plugin to extend the usage of a certain application. Traditionally, a plugin is a downloadable component, which can be implemented into the digital environment. This approach is already widely used within CAD software as they are relatively easy to implement without any technical skills, e.g. there are over a thousand Revit plugins featured on the Autodesk website. However, plugins provide a static package of code which builds upon the existing features of the digital environment for which it was built. This means that the plugin relies on the code provided by the digital environment. Thus when the digital environment changes one of these features, the plugin has to be altered as well.

The second option is using an Application Programming Interface (API) to extend the BIM environment. In contrast to the plugin, an API is built to be implemented into any application and describes how it should be implemented. It provides a piecewise code package, which allows any application to implement parts of the API as long it adheres to the provided protocols. However, when the provided features of the API change, the digital environment which uses the API must change their implementation code as well. This results in an environment that can be tuned to very specific specialisations but they either require the APIs to be implemented initially or they require the user to have the technical skills to make the connection between the BIM environment and the API.

Both plugins and APIs suffer from a great dependency on the existence of certain information, when the dependency changes, the other has to change as well. The result is that the BIM environment isn't fully extensible for any user, as is the case with APIs, or that the extensible features are built for a single specific BIM environment, as is the case with plugins. Thus, for the digital environment to be both extensible and mutually independent, it is important that any application, or one of its modules, doesn't expect another application to be part of the configuration. [31]

A third approach is to use module federation, to extend the BIM environment. Here, an application exposes its smallest unit of shareable code, called a module, to be implemented by any digital environment. This practice is called a federation because the digital environment serves as a centralised dashboard, in which each implemented module keeps relative internal autonomy. The result is a smooth integration of extensions to the digital environment while each module operates independently; neither the environment or the module need to be altered when the other changes [17]. For example, a dashboard application in a ConSolid-based environment, could implement a micro front-end for adding damage instances to the BIM model.

Instead that the micro front-end was specifically designed to be used within that specific dashboard application, the micro front-end is implemented without depending on the specifics of the dashboard. Because the micro front-end provides its own graphic user interface (GUI), it can be implemented into any environment without relying on already existing structure.

However, when we want some interaction between the BIM environment and the federated micro front-ends, some metadata has to be shared to configure key parameters. E.g. in the previous example, the dashboard wants to communicate the selected element in the BIM model to its micro front-ends. To enable this interaction, the micro front-end should provide some information about itself, metadata, so that the passed information is matched with the corresponding variable. This metadata is preferably provided automatically in a machine-readable manner, in order to resolve this interaction automatically.

Ultimately, the practice of module federation provides a viable approach for implementing micro front-ends into a BIM environment. For the purposes of this thesis, I will use module federation to create a prototype as a proof of concept for a modular BIM environment (Chapter 5).

3.5 Conclusions

With the emergence of the Semantic Web, Linked Data has come to the fore as an extensible data model. These databases are made machine-readable by RDF. In addition, syntaxes and query languages, such as Turtle and SPARQL respectively, allow for the creation of and iteration over multiple small datasets as if they were a single database.

In the context of BIM, current research discourse has moved away from the ifc format in favour of highly modular and specialised data formats. This progress has been made possible by the development of ontologies such as BOT; a RDF ontology that aims to provide a modular framework for Linked Data BIM models. In addition, ecosystems such as the Solid platform have paved the way for the complete decoupling of data from the applications that create and consume it. This is incredibly interesting in the BIM context as it enables the creation of a federated digital twin. One in which each stakeholder can provide their own information, while still maintaining their own highly specialised workflows. However, as the data model becomes adaptive and extensible, the interface through which people interact with that data must also become extensible due to Linked Data's inherent uncertainty. Preferably, this results in a BIM environment in which multiple micro front-ends are loaded and swapped out as the needs of the project change.

To integrate these micro front-ends smoothly, the current approach is to use module federation. This is because the code of the external micro front-ends are shared directly, allowing complete separation between the micro front-end and the BIM environment.

The focus of the remainder of this thesis will be on the creation of an extensible BIM environment through Linked Data and the federation of micro front-ends. This reinforces the earlier conclusion (section 2.4) where the scope shifted towards implementing external services in an adaptive BIMenvironment; wherein a BIMmodel can be initialised without a 3D model.

Chapter 4

Methodology

The concept of BIMification was introduced in Chapter 2, along with a preferred workflow for creating an information model of the existing context. Here, in order to cope with the enormous amount of existing data, the existing building, the workflow gradually increases the complexity of the BIMmodel. In addition, the BIM4Ren research highlighted the need for an adaptive digital environment.

To support such environment, the Semantic Web and Linked Data were introduced in Chapter 3. In this context, RDF provides a data model that is highly extensible and can link any type of information while being readable for both machines and people. In addition, Solid's principles proved to be incredibly interesting for adaptable BIM, as it allows data and applications to be decoupled and independent from one another. This enables that different applications and services can operate on the same data models. Specifically, the BOT ontology provides a framework for creating modular BIM models, while allowing for different specialised workflows. However, as the data model becomes extensible and thus uncertain, the interface through which users interact with the data must also be adaptive and extensible.

In this chapter I debate a methodology for creating an extensible BIM environment. This is done with a focus on creating BIM models for renovations and initiating a BIM model without creating a 3D model. Firstly (section 4.1), I outline an adaptation of the BIMification process that is more suited to both a Linked Data approach and a gradual increase in the complexity of the semantic information and the 3D model. Next, I discuss a schema for collaboration between stakeholders, multiple datasets and external services, resulting in a proposed structure for a BIM environment that can be extended by a micro front-end ”store” (section 4.2). I then discuss the resulting data model or BIM model built from multiple sources through different applications, stakeholders and services (section 4.3).

Finally, I present a Micro Front-End Store Ontology (Mifesto) as developed by Werbrouck et al. [31], which forms the basis for my own adaptation and proposal for such an ontology (section 4.4). The chapter concludes (section 4.5 with a brief overview of the entire workflow and a reflection on the resulting BIM model.

4.1 Focus on Anamnesis

As mentioned earlier (section 2.3), Scherer R.& Katranuschkov P.'s (2018) [26] BIMification method provides an approach to gradually increase the semantic enrichment and thus complexity of BIMmodels of existing buildings (Figure 4.1). In the framework there are three subsequent phases, namely Anamnesis, Diagnosis, and Therapy. The BIMification process itself exists of the Anamnesis and Diagnosis, and describes the analysis of the building before a renovation plan is created. Then what is typically called the "design process" includes the Therapy phase as part of the retrofitting design.

In the Anamnesis phase, the first step starts by creating a complete geometric representation of the entire building. This provides a partial solution to the challenge of creating a BIM model of an existing building; the high upfront conversion effort remains. However, the BIMification process allows a shift towards an information-first BIM approach. This is because the main workflow is concerned with gathering information in a structured and holistic manner; which provides a solution to the challenges renovation design faces when adopting BIM workflows.

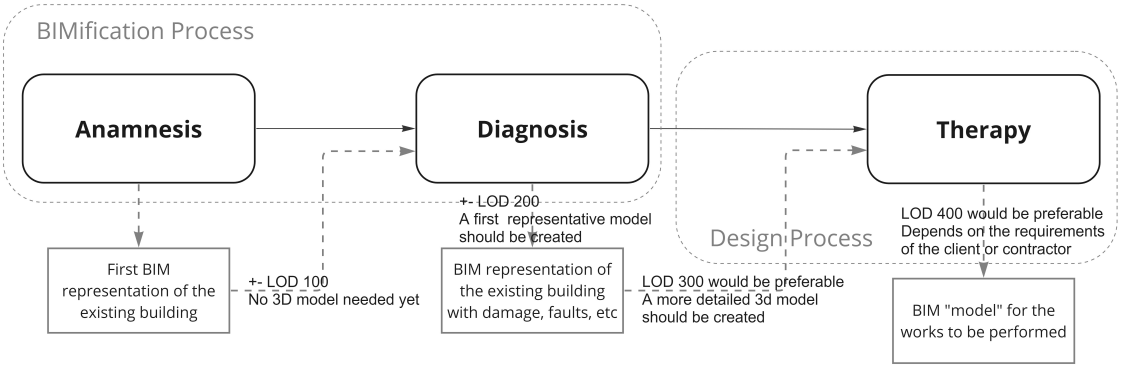

Figure 4.1: Proposed BIMification process with preferred LOD indications

Therefore, as shown above (Figure: 4.1), I propose a variation of the BIMification schema in which the concepts of Anamnesis, Diagnosis and Therapy persist. However, the LOD of the BIMmodel increases as the project matures. The different LODs are defined here per Level of Development Specification by BIMForum (2021) [4].

Starting with the Anamnesis, there is almost no need for a 3D model, as in a traditional renovation or retrofit project, the first phase is focused on gathering information and surveying the building. When the project enters the Diagnosis phase, only a rough 3D model is needed. Here, indications of the presence of certain elements are more important than the actual dimensions, so only a LOD of 100 is required. At this stage, the low LOD model serves mainly as a human readable representation and as a basis for further refinement. During Diagnosis, the LOD should be increased to 200 to create a representative BIMmodel of the existing building, including damage, defects, etc. When the BIMification process is complete and enters the design process, the BIMmodel should be able to handle a LOD of 300 in the relevant areas, i.e. areas where precise work needs to be done and accurate measurements are required. Finally, in the design process or Therapy phase, the retrofit design should be further developed into a BIMmodel, preferably at LOD 400 in areas where work needs to be carried out.

To enable this gradual increase in LOD, a Linked Data-based BIM format is proposed. This allows to first create semantically enriched elements before a 3D representation is required. To provide some clarity in this initially ”empty” BIM model, a basic hierarchical structure of the building is created, including the floors, spaces, etc. This minimal description of the building serves as the overarching hook to which further enrichment of the BIM model can link to.

For example, a database is set up for the renovation of a residential house, and a basic structure of the house is described in a RDF graph. In this graph, the site and building are included as elements of the BOT ontology, together with each floor and its corresponding rooms or areas. Then in the Anamnesis phase, a crack in the facade, some discoloration in neighboring rooms, etc are noticed and linked to the relevant elements in the elements and spaces. These are then classified as certain types of damage, such as water damage in a room facing the street as a result of the crack in the facade, in the Diagnosis phase and added to the graph of the BIM model. Eventually it is decided that the facade will be reconstructed to provide more insulation and simultaneously repair the earlier water leakages. However, in this example, the rear construction was completely fine and will be left unaltered. Then in the Therapy stage, only the facade and its adjacent elements have to be modelled in detail to provide a clear plan of the works to be performed.

This way, contractors have a detailed BIM model to work from, while the architect doesn’t have to create an extremely detailed digital twin of the entire building. Furthermore does this framework allow stakeholders to directly contribute to the BIM model as it allows information to be linked to the conceptual elements of the building.

The result is a BIM model that starts of as a mere database for the construction project and throughout the project's development, this database grows into the building's digital twin. Although Linked Data is proposed as a suitable data model for the gradual increase of information, the process of BIMification is a workflow that can be adopted, at least partially, regardless of software.

4.2 Extensible BIM Environment

Once the digital twin moves to an adaptive format such as RDF, the need for an extensible BIM environment arises, as discussed in section 3.3. This is reinforced by the results of the BIM4Ren research (section 2.2), which proposed a collaboration schema (Figure 2.1) for a BIM environment where users interact with both the digital twin and external services. However, this schema provides a rather proprietary ecosystem, as the proposed structure is specifically built from services developed during the BIM4Ren project.

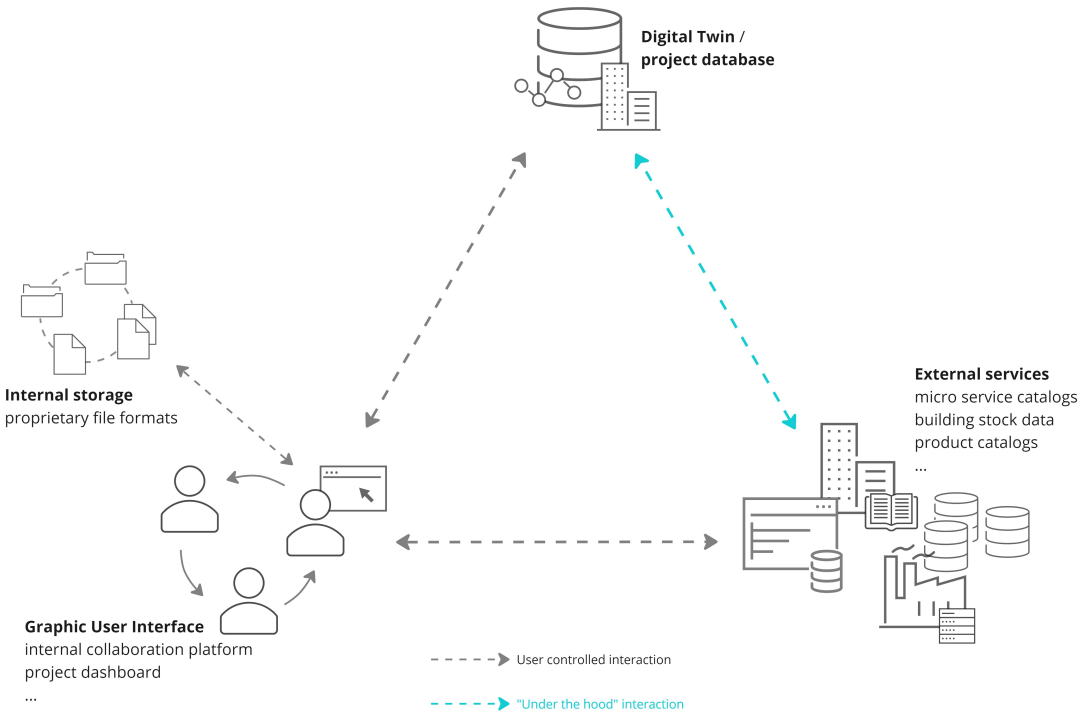

Figure 4.2: Proposed collaboration schema

In order to create a more general and open approach to an adaptive and extensible BIM environment, an alternative collaboration schema is proposed. This schema has the goal of providing a digital environment which can be tailored to the specific needs and expertise of any stakeholder.

As shown above (figure 4.2), the structure starts with the user interacting with the project through a GUI that acts as a collaboration platform and dashboard for the project. The GUI should allow interaction with the project's database or digital twin, independent external services and local files or file repositories within the stakeholder's organisation. Furthermore, the interaction between the project database and any external services is managed ”under the hood”; directly by the external services. This results in a BIM environment that extends itself with external functionalities when needed. While it reduces the technological barrier to entry, allowing the user to use the platform without the need for technical skills to make the connections between every component in the environment.

However, this schema doesn't necessarily provide an integrated environment in which any external service can communicate with any BIM model. Furthermore, the connection between the GUI and the various external services is managed by the user, requiring them to know of the existence of the service and know how to implement them.

In order to create a BIM environment in which multiple applications can contribute to multiple BIM models through a variety of external services, the collaboration schema is extended (Figure 4.2). The new structure follows the principles of ConSolid as laid out in section 3.3 to create an environment that is even more adaptable while managing the integration of external services. In this structure, the digital twin is turned into a federated BIM model consisting of a multitude of smaller datasets provided by the different stakeholders, as shown below (Figure 4.3). These datasets can be queried as one large digital twin and are connected by an overarching RDF graph that describes the overall building and its conceptual elements. In addition, by using Linked Data to create these datasets, proprietary file formats can still be referenced in the graph, e.g. the graph can link to a Revit file exposed via a regular URL; provided that appropriate metadata is present.

To ensure the UI can implement the external services as a fully integrated environment, the shift towards micro front-ends is made. In this approach, each external service provides their own small UI component, which can be implemented directly. To ensure the integration of these micro front-ends happens smoothly, their implementation is managed ”under the hood” by a micro front-end store. This has the additional benefit that the store can validate the external functionalities while at the same time making them navigable. In this manner, the micro front-end code is shared directly with the federating UI, so that the micro-front-end has direct access to the federated digital twin.

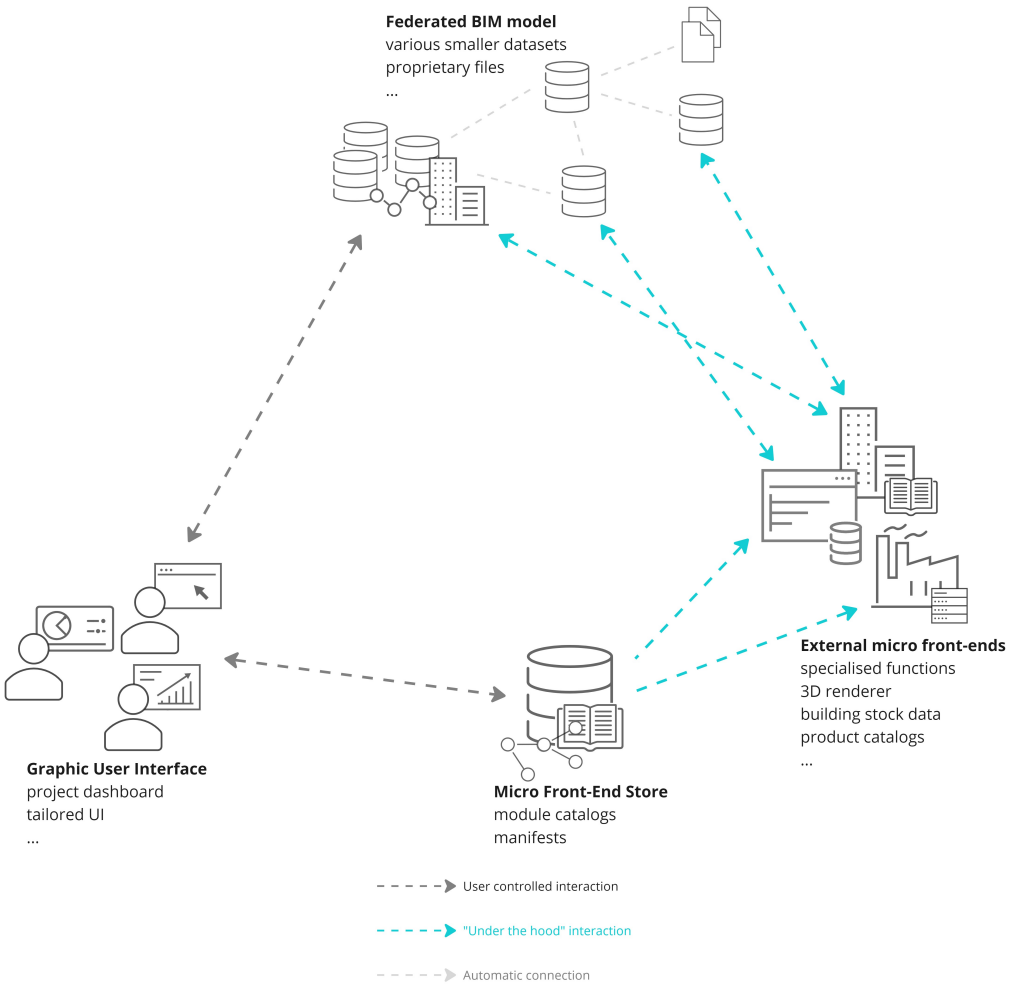

Figure 4.3: Extended collaboration schema

As an example, a damage inspector documents the structural defects of a construction using specialised application for damage markup. However, the inspector wants to include some images together with the assessment of the impact on the structural integrity. For both of these a micro front-end is federated into the application to extend its functionality. Then, instead of delivering a pdf of the complete report, the report is presented as a partial dataset of the BIM model, from which the damage instances together with the structural impact is linked to the corresponding elements. Finally, when the structural engineer wants to assess the structure, their preferred application can in turn be extended to easily display the structural impact found by the damage inspector.

Ultimately, this allows users to take advantage of the adaptive BIM environment while reducing the technical skills required and creating a more integrated user experience (UX). Furthermore does this allow for a workflow in which handovers are removed, putting quality checks over the entire federated BIM model in place.

This is because instead of requesting several files from multiple parties, a stakeholder can simply retrieve the information from the datasets and extend their application to display this in a structured way.

4.3 Graph Schema

When using the methodology described above (sections 4.1 & 4.2), the construction project, or at least its BIM model, is structured in a specific graph schema. The goal is to create a BIM model in which each stakeholder can contribute directly, each using highly specialised software tailored to their expertise.

This structure starts with an overarching dataset which describes the building and its basic elements. Then each stakeholder produces their own dataset with their contributions to the construction project. By linking to the conceptual elements in the building's overarching dataset, each representation of a certain element retain their coherency; in other words, to make sure they are describing the same element. As the data model and the environment are extensible, the information in these datasets can be referenced by any future stakeholder. The latter is interesting when making BIM models more accessible for management throughout the building's lifecycle. In addition, to retrace the particular extension and participant who created the data, each resource or piece of information should have an associated origin. This origin describes which application, when and by whom the data was added to the BIM model. This is necessary because the graph could theoretically be extended indefinitely by an infinite number of applications.

In the following example, I will use the personal online datastore (Pod) concept to represent any type of dataset stored in a file exposed by a server, possibly locked behind an authentication method; for simplicity, I will refer to these as "a Pod".

In the Belgian context, the overarching dataset is likely to be stored on the architect's Pod, or on the client's if they are a professional developer. This overarching project dataset contains the overall project description and aggregated the partial project datasets. In addition, it is also this Pod where the BOT description of the building and the conceptual elements will be stored. The partial projects contain a specific description of a stakeholder's contributions and, of course, the contributions themselves; which are stored on the stakeholder's Pod. Both the stakeholder and the overarching dataset have a conceptual element which are linked to each other. For example, the report of the damage inspector is stored on their own Pod and references the building description provided by the architect.

Each would have a conceptual element of the facade, which reference each other to link both representations together. This ensures that each dataset is readable without necessarily having to go through all the separate Pods. Eventually, the entire federated BIM model might be copied to a Pod specific to the building once the construction project is complete, which in turn could be used for building lifecycle management.

The result is a federated BIM model made up of smaller contributions delivered by the various stakeholders. Whether this can still be called a BIM ”model” is debatable, however the result is a tool for Building Information Management in the broader sense of the term. In the Belgian context, this generally means that the architect is responsible for creating the overarching BIM model to which the additional data is linked. Each stakeholder is then responsible for maintaining the information they have provided. As this information can be directly linked, the need for model handovers is reduced, instead quality checks can be performed. The addition of origins make any part of the graph navigable in terms of contributions, editing history and which extension was used to create the data. This enables the produced data and workflow to be replicated.

The downside of this approach is that multiple parties are expected to maintain their data so that the next construction project can benefit from the previous digital twin. A possible solution would be to merge the different datasets into a single repository once the construction project is delivered. This would recreate the single source of truth from a single BIM model, providing all the generated information about the building in a single database. Because the data is structured as a RDF graph, the information is software agnostic and therefore remains accessible even after further development in the used applications. However, the question of who owns the data comes to the fore by effectively copying everyone’s contributions into a single BIM model. Where previously intellectual property was easier to manage because each stakeholder provided their contributions as separate datasets, it is now blurred again by combining everything into a single BIM model. Although quite pressing and increasingly important, the discussion concerning digital ownership within the AEC industry is a whole research topic in its own right and falls outside the scope of this thesis.

4.4 Mifesto ontology

As mentioned above (section 4.2), the BIM environment is extended through the use of micro front-ends. These are navigated and implemented through a micro front-end store to reduce the technical skills required to benefit from such an environment.

To create this micro front-end store, I will be using Linked Data, however, describing the needed metadata of these micro front-ends requires an ontology. For this I will use the Micro Front-End Store Ontology (Mifesto) by Werbrouck et al. (2022) [31] as a foundation for a proposed revision of the Mifesto ontology. The ontology should enable a micro front-end to describe their inner workings, with the aim to be implemented through a store. In the following subsections I will highlight how the revision deviates from the original ontology. This will be done in three levels, on the level of the store (section 4.4.1), the manifest (section 4.4.2) and on the level of the origin (section 4.4.3).

In the following sections, I provide graphical representations of the Mifesto as developed by Werbrouck et al (2022) [31] and of the proposed revision; in which the specific changes will be highlighted. The sections go over the most important alterations, the complete revision is included in appendix A[1].

Furthermore are the classes and properties of the used ontologies prefixed, as is common practice in Linked Data. The used prefixes are shown below (Listing 4.1). The Mifesto ontology however, isn't published yet, these instances will get the prefix "mifesto:" but this prefix isn't included in the listing.

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix dcat: <http://www.w3.org/ns/dcat>.
@prefix dct: <http://purl.org/dc/terms/>.
@prefix fno: <https://w3id.org/function/ontology#>.
@prefix bot: <https://w3id.org/bot#>.
@prefix consolid: <https://w3id.org/consolid#>.
```

Listing 4.1: Used prefixes

[1]Also made available on github: https://github.com/RobbeMic/thesis_Digital_Environment

4.4.1 mifesto:Store

The store itself is a container resource, in Linked Data this is often called a catalog, meaning that it contains a collection of other resources. The store thus references modules, interface configurations or other catalogs, as shown below (Figure 4.4). The store holds a number of manifests, these are documents which describe the configuration, functionalities, required input, expected output, etc. of the external micro front-ends. These can be used to validate applications by placing requirements on the expected output; e.g. that the output has an origin attached.

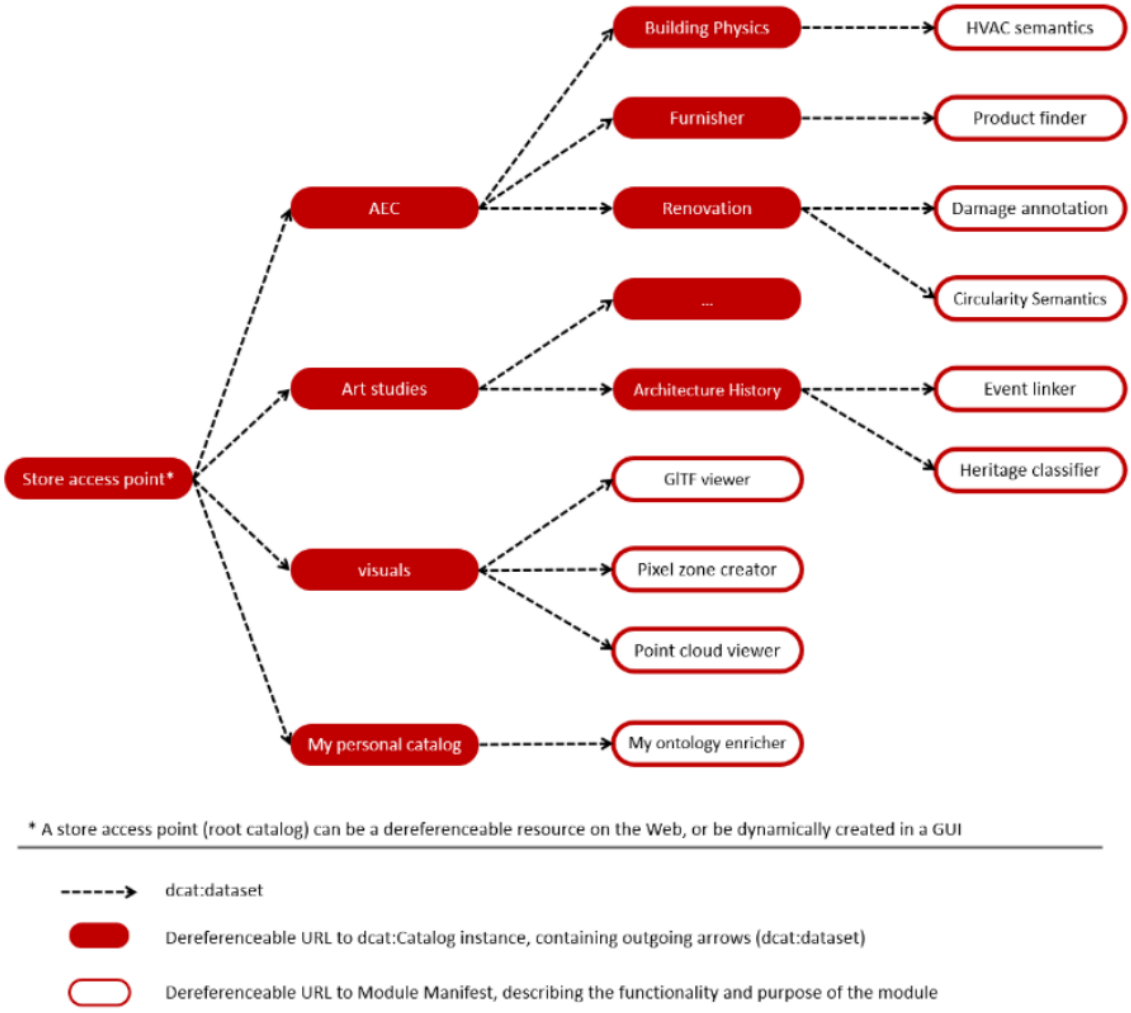

Figure 4.4: Example of a nested catalog as visualised by Werbrouck et al. [31]

The manifest also serves as the ”manual” through which the dashboard implements the micro front-end, as shown below (Figure 4.5). The user requests a list of module descriptions and by selecting to implement a particular micro front-end, the code is implemented ”under the hood”, just as in any application store. In addition, the ontology provides an interface configuration instance that contains instructions for generating a UI. [31]

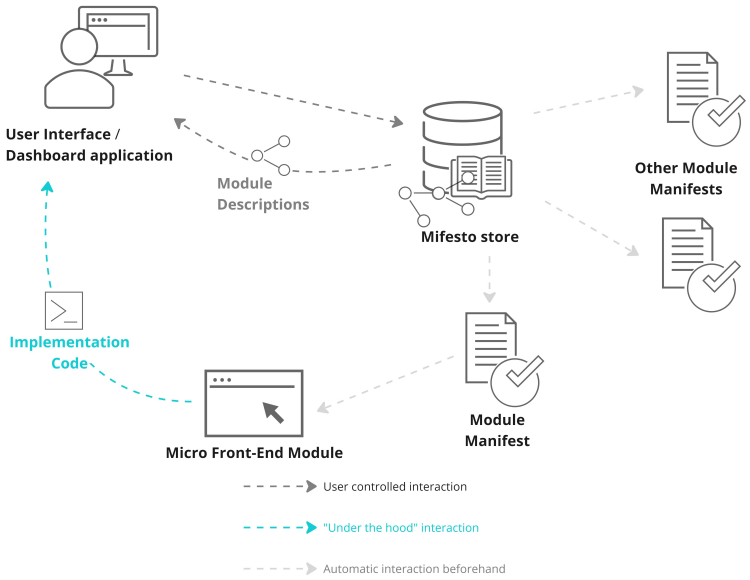

Figure 4.5: Implementation of an external module through a micro front-end store

For the actual definition of the store instance, Werbrouck et al. prescribe using a class of mifesto:Store which is equal to dcat:Catalog. Then dcat properties are used to define the store and reference its contents. This is an appropriate practice for the purpose, so I haven't added any more definitions. As a result, multiple stores can point at the same document, providing a single source of truth for the module manifest; maintained by the micro front-ends themselves. In this manner, all of the aggregated manifests can be retrieved with a simple SPARQL query (listing 4.2).

```
SELECT ?module

WHERE {
    ?store a mifesto:Store, dcat:Catalog;
        dcat:hasPart+ ?module.

    ?module a mifesto:Manifests.
}
```

Listing 4.2: SPARQL query to retrieve all the mifesto:Manifests.

4.4.2 mifesto:Manifest

At the level of the manifest, the Mifesto ontology developed Werbrouck et al. [31] is visualised is figure 4.6. In contrast, the revised ontology (Figure 4.7), adds a definition of whether the micro front-end has either a functionality module or a visualisation module. These are respectively defined in a mifesto:FunctionModule and a mifesto:ViewerModule instance. Both of these modules have a separate release described in a mifesto:Release instance, this separation between the module and their release is needed in order to provide new versions of the module without having to remove the older version.

In the release there are object properties for defining the in- and output of the micro front-ends. For the input, the original ontology provides the mifesto:readsParameter and the mifesto:readsIdentifier properties. Respectively, these are used to link the release with a consumed parameter and to indicate the compatibility of the release with a certain identifier. [31] As both describe compatible or even required parameters of the micro front-end, these are consolidated into a single mifesto:consumesInput property. This new property is defined as pointing to a fno:Parameter instance, which in turn has further properties to define a class, label or requirements the parameter should conform to. There is an example of such a fno:Parameter included in the example manifest below (Listing 4.3)

Similarly, for the output, the mifesto:readsParameter and mifesto:readsIdentifier properties are included in the original ontology. These two object properties either link the release with a returned parameter or indicate what the release registers in the database. These have also been consolidated into a single mifesto:expectedOutput property, which points to a fno:Output instance. This output then further demarcated by a fno:type or dcat:conformsTo property describing its format. In addition, the revision includes a mifesto:outputDescription property to give the user a detailed description of what data will be added to their project graph. When the mifesto:expectedOutput and mifesto:outputDescription are combined, the user should have a specific and technical description of what the micro front-end will alter or add to the project graph.

Additionally, the mifesto:code property is replaced by the mifesto:hasModule property, instead of pointing directly to the exposed code, and thus points to a mifesto:Module instance. This Module instance has a misto:isExposedOn property, which performs the same function as the mifesto:code property, but by making the module a separate instance, it allows more metadata to be added to the module's code; e.g. a dcat:conformsTo describing the module's stack description or coding language.

Finally, the mifesto:Manifest instance itself is given three more properties, namely mifesto:moduleName, mifesto:moduleDescription and mifesto:hasDepiction, which are for a name, description and depiction to be displayed in the store respectively. In fact, for several of these additions to the ontology, there are already existing properties or classes with the same functionality, as which they are then directly defined. However, the direct inclusion in the ontology makes the adoption of the ontology more suitable for newcomers to RDF, as it reduces the amount of ontologies they need to understand. This accessibility is needed as every micro front-end developer is expected to provide a manifest of their application, which doesn't necessarily uses Linked Data. Moreover, it doesn't reduce the modularity of the ontology, since RDF allows you to reason that these are the same because they are defined as rdfs:subPropertyOf.

As an example, the manifest for a micro front-end for rendering 3D models is included. The micro front-end has both a functionality and visualisation module, for adding objects and displaying them respectively. Both modules have a separate release and actual module, however, the parameters are shared when appropriate. This allows to ensure that identical parameters are also defined identically. In order to keep the RDF listing concise, below (Listing 4.3) is a part of the entire manifest, which can be found in Appendix B.

```
<#manifest> a mifesto:Manifest;
    mifesto:moduleName "3d_renderer";
    mifesto:hasViewerModule <#viewerModule>;
    mifesto:hasFunctionModule <#functionModule>.

<#viewerModule> a mifesto:ViewerModule;
    mifesto:hasVersion <#release_viewerModule_1>.

<#release_viewerModule_1> a mifesto:Release;
    mifesto:versionNumber 0.1;
    mifesto:consumesInput <#_datasetParameter>,<#_itemParameter>;
    mifesto:hasModule <#module_viewerModule>.

<#module_viewerModule> a mifesto:Module;
    rdfs:label "./viewer";
    mifesto:isExposedOn "http://localhost:3100/remoteEntry.js".

<#_itemParameter> a fno:Parameter;
    rdfs:label "item";
    fno:type bot:Element;
    fno:required "true"^^xsd:boolean.
```

Listing 4.3: Example of a micro front-end manifest, describing its two connected modules

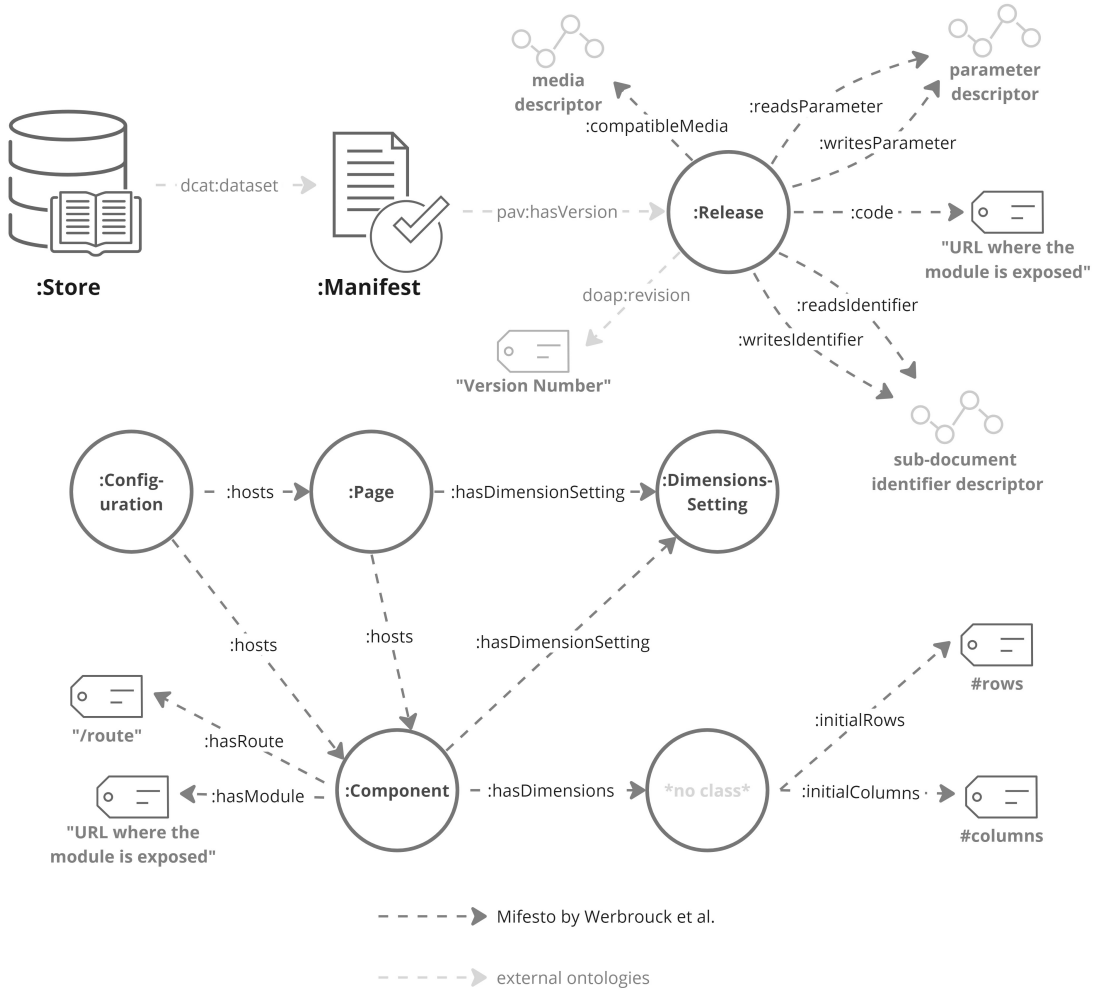

Figure 4.6: The Micro Front-End Store Ontology (Mifesto) as described by Werbrouck et al. [31]

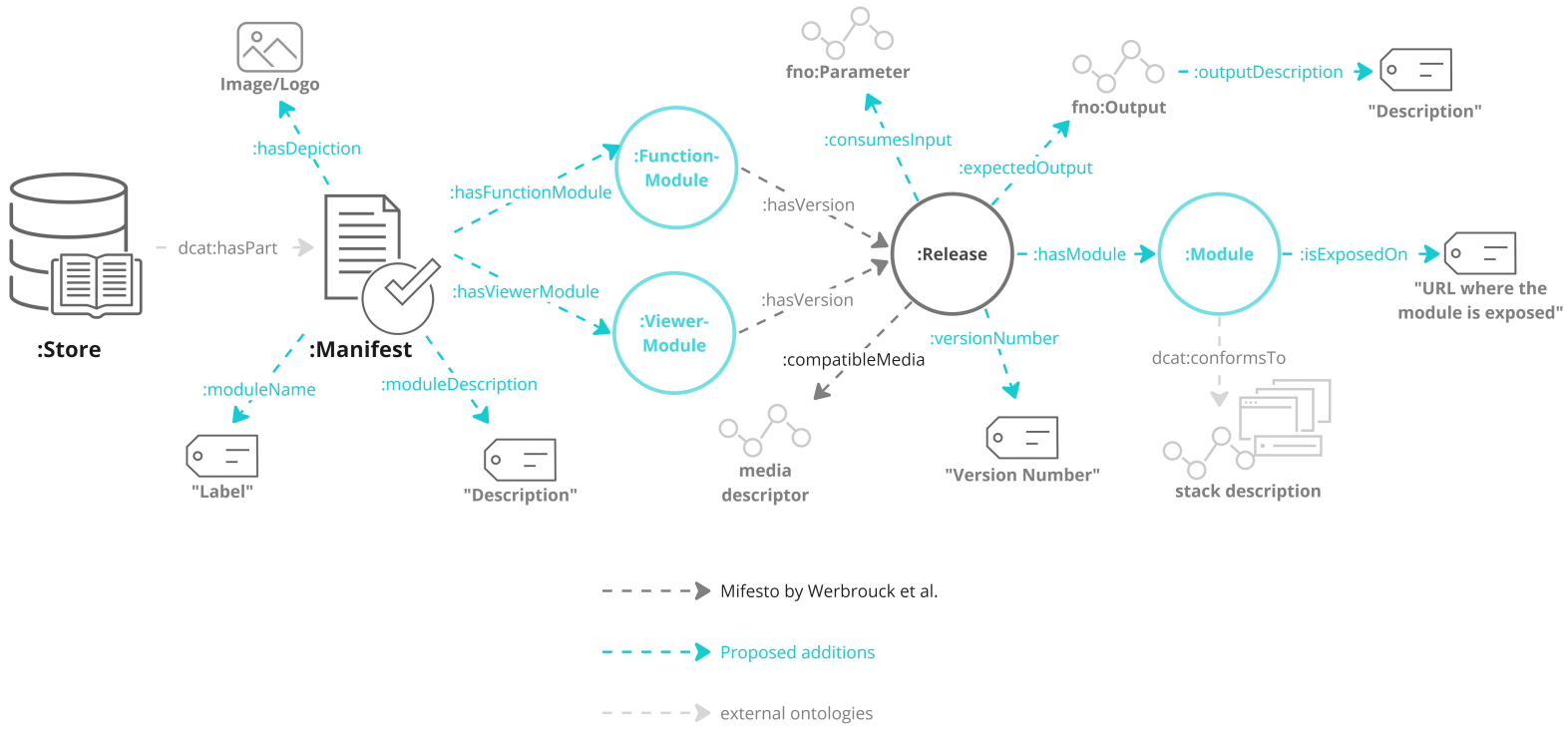

Figure 4.7: The proposed alteration to the Micro Front-End Store Ontology (Mifesto)

It’s worth noting that I didn’t include any properties or classes to describe the configuration of the micro front-end on the web page. This is because including a full style description would be an extensive ontology on its own. This is handled somewhat by defining the module as either a functionality or a visualisation component. The reasoning is that since the code is directly shared, the layout is also shared. Ideally, this layout should scale to the space available. If we look at traditional CAD systems, there is often a scalable viewport to increase workflow efficiency and accessibility. In a function component, the shared module shouldn’t contain much more than a set of buttons associated with functions, these buttons may be given a layout or this may be overwritten by the host web page. On the other hand, a viewer is intended for any kind of visualisation of the graph, e.g. this could be a simple list focused on a particular aspect of the industry, or a 3D renderer that finds all the connected geometry and renders it to create a coherent 3D model. In RDF, however, it is possible to define a short style description in the Mifesto without it being mandatory, allowing a more comprehensive ontology to be used later to describe the layout within a particular framework. So there is no real negative impact from the inclusion of a concise style description in the Mifesto.

4.4.3 mifesto:Origin

As described above (section 4.3), the addition of an origin is necessary to allow backtracking of data in the graph. Thus, a class to describe the origin of a resource is included in the Mifesto. This class and its properties describe which micro front-end or application created the resource, when and by whom. In this implementation, the origin of a resource refers to the manifest of the micro front-end, with the assumption that this provides a comprehensive description.

Although practical, this definition of an origin would be more suited to be included inherently in the dcat ontology. In essence, the inclusion of mifesto:Origin is a consequence of dataset management instead of federating micro front-ends through a store. However, in the meantime it remains included in Mifesto to allow users to find the micro front-end with which another stakeholder has created a part of the graph.

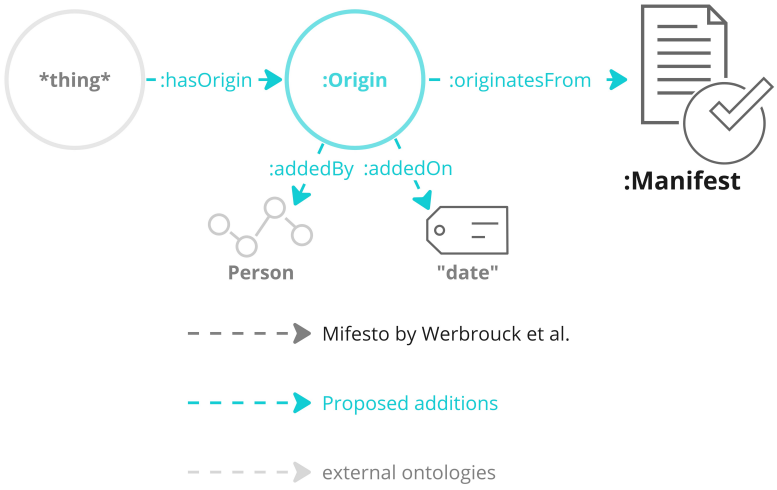

Figure 4.8: The Mifesto Origin class

4.5 Conclusion

In an information-first BIM approach, the need arises to be able to create a BIM model without necessarily creating a 3D model. To this end, the proposed methodology was developed to create an extensible BIM environment, which could adapt itself to the information in the BIM model.

The method starts with the process of BIMification, which advocates increasing the LOD of a BIM model as a project matures. To enable this gradual increase, an extensible data model is adopted. This approach uses BOT, and linkset objects to allow to link information about specific building elements to their conceptual idea before any representation is present.

However, the shift towards an adaptive BIM model requires an extensible BIM environment. Here a UI serves as a dashboard for the project and allows for interaction with the project's digital twin and external services. This schema is extended with the ConSolid principle to create a federated BIMmodel, consisting of smaller datasets provided by different stakeholders. The UI is tailored to the various needs and expertise with the implementation of specialised micro front-ends. This integration is managed by the micro front-end store, reducing the required technical skills and creating a more integrated user experience.

In order to support the integration of micro front-ends into a store, an adaptation of the Mifesto ontology is proposed. In this structure, each micro front-end has a manifest describing their inner workings and provides a human-readable description. Then there is the Store which is a catalog comprising both manifests and other catalogs. By querying over the store users of the adaptive UI can find and implement modules whose manifest were included.

The resulting BIM model is structured using a RDF graph, starting from a project catalog containing the overall project description and the smaller datasets from which the project consists. The overarching dataset includes the BOT description of the building and any conceptual elements. The contributions of each stakeholder are then defined in the partial projects and use conceptual objects to link overlapping concepts to each other. Each resource has an attached origin to describe from where it originates. This makes each part of the graph navigable in terms of contributions and which extension was used to create the data.

Chapter 5

Prototype

As mentioned in the introduction (section 1.2.2 & 1.2.3), there are currently no existing applications that enable the workflow described above (section 4.5). Therefore, a prototype has been developed to provide a ”proof of concept” for the proposed methodology. More specifically, the prototype attempts to create a dashboard application that uses a Mifesto store to implement micro front-ends specialised in modifying a rudimentary Linked Data-based BIMmodel.

In the following chapter, I will first discuss the different components that make up the prototype and the role that each component attempts to fulfil (section 5.2). Next, I will discuss how these components interact to create the provided UX (section 5.3). Finally, I discuss the results of the prototype and conclude with a short reflection on the prototype as a proof of concept (section 5.4). A more detailed evaluation of the prototype is given in Chapter 6 (section 6.1).

The complete code of the prototype can be found in a github repository[1]. This repository contains several folders, each containing a different component of the prototype. A guide to setting up or navigating through the prototype is detailed in the repository’s readme.

5.1 Requirements or Dependencies

The web applications are built using JavaScript, more specifically the React framework with webpack 5[2]. Webpack is a JavaScript application bundler that provides a basic module federation setup, hence the choice of this particular framework. The databases were created using the Solid Community Server[3], which runs locally with Node.js and allows both local files to be exposed as information in the Turtle format.

[1] https://github.com/RobbeMic/thesis_Digital_Environment

[2] More information can be found at https://webpack.js.org/concepts/.

[3] More information can be found at https://solidproject.org//self-hosting/css.

To query this information, the Comunica framework[4] has been implemented, this framework allows the execution of SPARQL queries within a JavaScript application. Furthermore, Comunica provides an extension specifically for querying multiple links within Solid Pods[5], while also being able to handle the required authentication.

For all 3D rendering, the three.js library was used in combination with the react-three-fiber renderer. Three.js is a JavaScript library that uses the webgl system to create a lightweight 3D environment in the browser[6]. React-three-fiber is then used to implement this lightweight but powerful renderer into the React ecosystem[7], allowing it to be used like any other component in the prototype's environment.

5.2 Structure of the components

Overall, the aim of the prototype is to initiate a BIM model that allows semantic enrichment of elements without 3D representation. The structure of the prototype and its components are analogue to the collaboration structure presented in section 4.2. The prototype consists of three distinguishable parts, the dashboard which serves as the main entry point for the user, the external micro front-ends that are loaded into the dashboard, and the database or BIM model where the information is actually stored. The following sections cover each of these to highlight the key elements in their structure and discuss the role each component has to perform.

5.2.1 Dashboard

The dashboard application is the overarching GUI through which the user interacts with the BIM model and the external modules that are loaded into it. The dashboard is built as a React application using webpack 5 to enable module federation by default. This setup is run locally in the browser to simulate a web application. The dashboard provides several functionalities, the most important of which are authentication, access to the variety of projects stored on a personal Pod, access to the project data itself, and access to the micro front-end store.

[4] More information can be found at `https://comunica.dev/docs/query/`.

[5] More information can be found at `https://www.npmjs.com/package/@comunica/query-sparql-link-traversal-solid`

[6] More information can be found at `https://threejs.org/manual/#en/fundamentals`.

[7] More information can be found at `https://docs.pmnd.rs/react-three-fiber/getting-started/introduction`

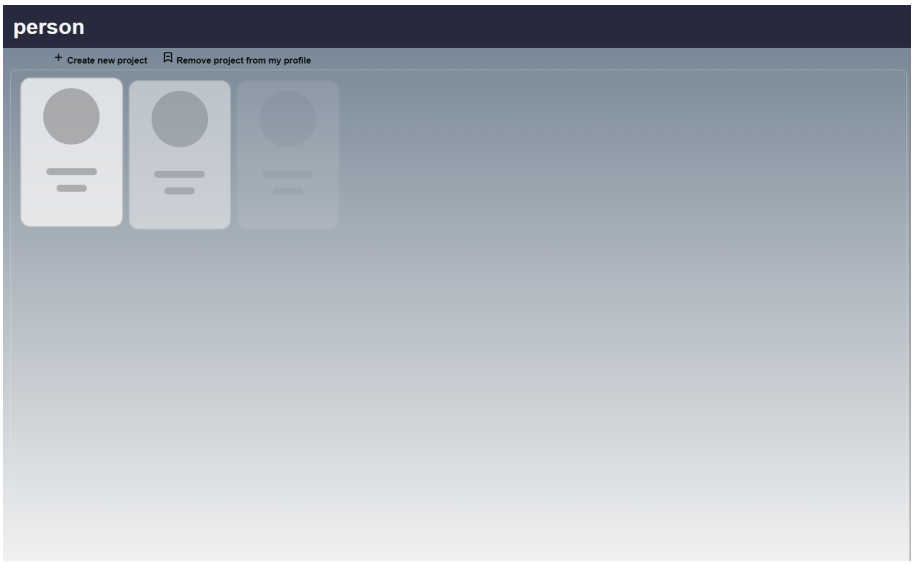

Figure 5.1: UI from the profile page, loading the projects connected to a person's Pod.

The micro front-end store is part of the dashboard application and refers to a single Mifesto Store catalog. This is because in the small scale of this prototype there is only one catalog. In addition, the use of the store is considered as a basic functionality to test the concept of an extensible BIM environment and the Mifesto ontology.

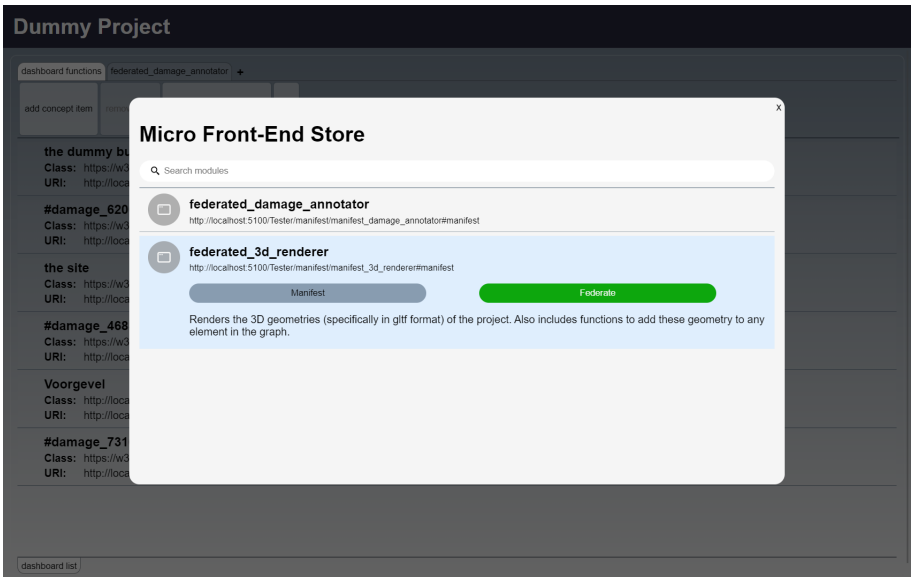

Figure 5.2: The UI from the store component

5.2.2 Micro Front-End services

The various micro front-ends are also created using Webpack 5. This allows the modules to be federated into the dashboard as well as used as standalone web applications. The metadata for the modules is provided in the manifest of the micro front-end, which is stored in a separate Pod for easy exposure of graph data, rather than having the application expose it and manage authorisation itself. The injected code of the functionality modules are assumed to return a list of buttons. These buttons can each provide further functionality, e.g. a button to add a damage instance can call a form that pops up, where the relevant information has to be filled in before this information is added to the graph. Such a window, that pops up on demand, is often called a modal in the context of web development and will be referred as such in the figure below (Figure 5.3) and the remainder of this thesis.

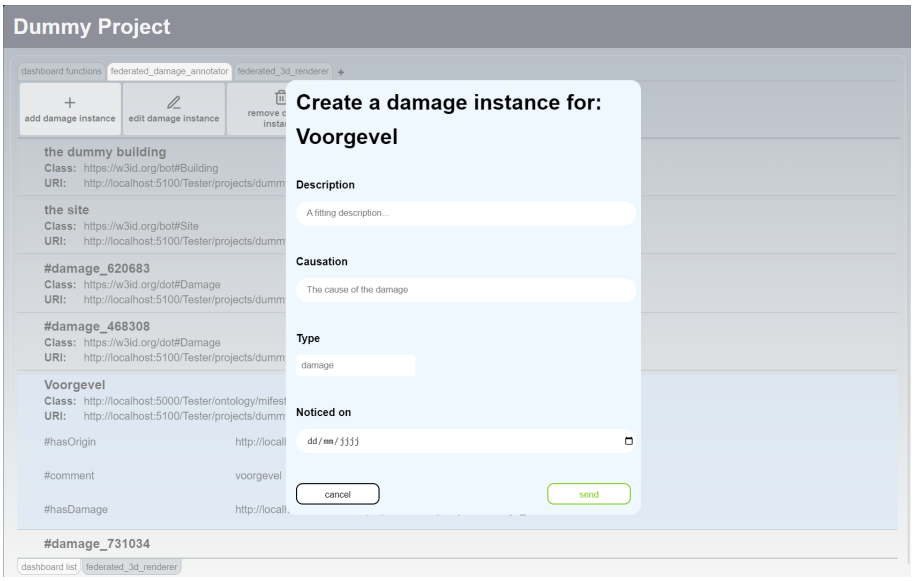

Figure 5.3: The modal called by the damage micro front-end.

However, the visualisation modules are expected to return a container element in which all the visualisation functionality is present. These assumptions are made to provide a practical implementation in the dashboard.

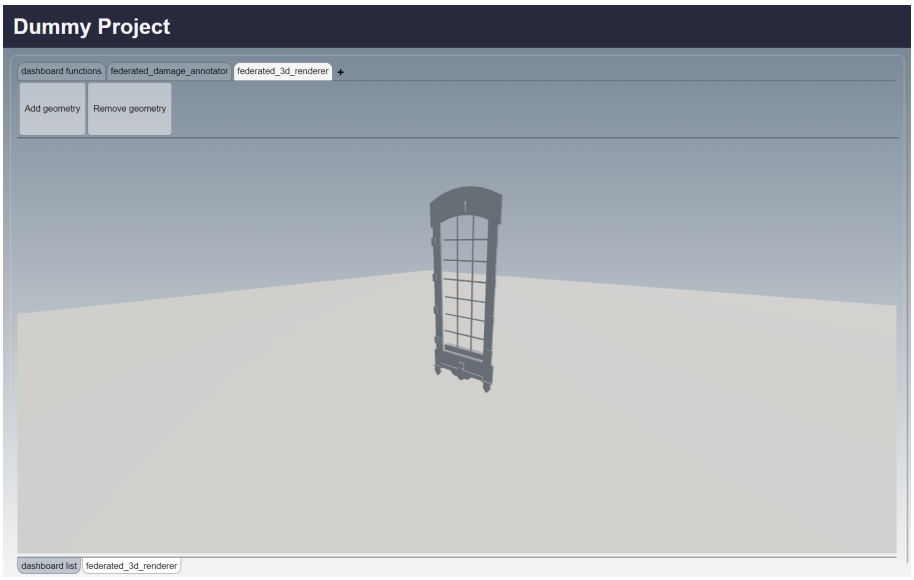

Figure 5.4: The visualisation component from the 3D rendering micro front-end.

The latest iteration of the prototype incorporates two micro front-ends into the store. Their purpose is to test the practical implications of a micro front-end store in combination with first adding information to the elements and then adding 3D representation. The first contains a functionality module for creating and editing damage instances. For each function, the module calls a modal that requires more information, the subject of this information is always the previously selected element. The second micro front-end is a 3D renderer that includes a functionality module and a visualisation module. The visualisation module searches the project graph for any 3D models and when a geometry is clicked, the corresponding element is selected in each component. The 3D models are included as separate files, as RDF descriptions are rather impractical for directly incorporating 3D geometry. This also tests the functionality of using external ”closed” file formats stored on the Pod providing server. The functionality module of this micro front-end includes two simple buttons for adding or removing a 3D model to or from the selected object in the graph.

5.2.3 Digital Twin/Project Database

Two separate local Pod provider servers were used to store the graph data in combination with ”closed” file formats. These are both Community Solid Servers, each with a different level of authorisation. One is a pure development server, allowing applications to freely read and write information to the server. The other requires applications to log in to an account managed and trusted by the server.

The personal Pods and the revised Mifesto ontology reside on the more secure server, as it requires authorisation before changes can be made. This provides a basis for testing the workflow in relation to databases that require trusted applications. On the other hand, the project's Pods, other "closed" files and the module manifests are stored on the development server, allowing rapid testing of functionality throughout the development of the prototype. The latter implies that the BIM models didn't require authorisation in the prototype, but it is recognised that data security is critical in real-world implementations.

5.3 Interactions

The interactions between the dashboard, the micro front-ends and the various Pods are all handled "under the hood" to ensure a smooth user experience (UX). First, the default landing page of the dashboard provides a form to access a Pod provider, where the user must log in to find the specific URI associated with the person's personal Pod. The user is then redirected to the dashboard, which now reads the information on the personal Pod to retrieve any projects listed. Clicking on a project will take the user to that project's dashboard. For querying across multiple personal online datastore (Pod)s, the Comunica framework has been used in both the dashboard and micro front-ends. It's not necessary that both use the same query engine, as the principle of querying URI's is universal, but for practical reasons only one query framework is used.

This "project page" lists the elements present in the rudimentary BIM model and provides basic functionality. Most important in the context of this "proof of concept" is the access to the store, which retrieves all the manifests listed in a Mifesto Store. The micro-front-ends described by these manifests can then be implemented in the dashboard by clicking on the "install" button, which will retrieve and inject the exposed code. To reuse this module in the future, the micro front-end is added to the list of preferred modules on the user's profile. The latter is implemented to tailor the environment to the user's preferences and is configured the same for each project the user opens. However, it is possible that in another dashboard application, the list of saved modules is project and user specific.

The interaction between the dashboard and the micro front-ends is managed by a shared state. To elaborate, the selected item and the associated database or Pod is stored as a locally defined parameter. The parameter itself and the function to update this parameter are passed to the micro front-ends, which in turn are updated when this state changes. The names that the passed variables should have are retrieved from the manifest as specified in their mifesto:consumesInput properties. This ensures that the micro front-ends understand what parameter is being passed to them.

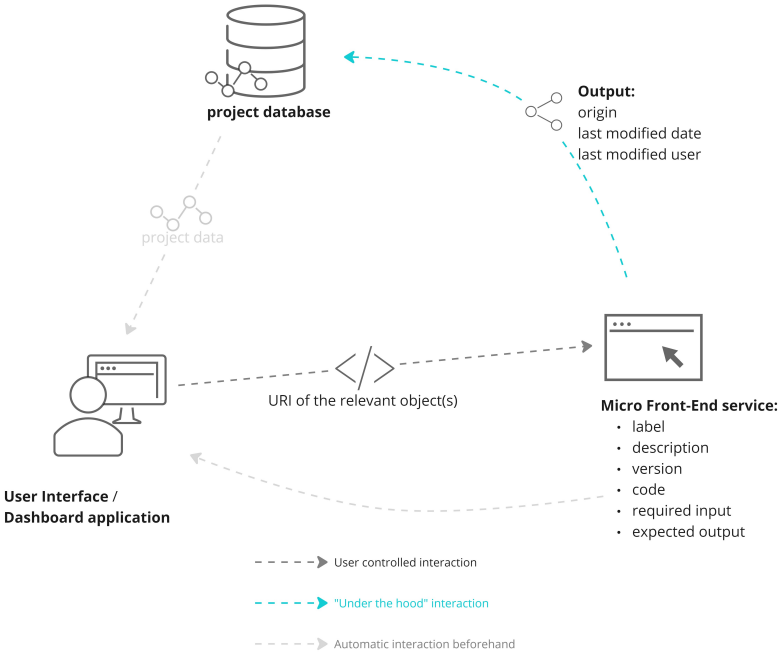

Figure 5.5: Micro front-end interactions as implemented in the prototype.

However, for practical reasons, this interaction assumes that the same variable uses the same IRI as specified in the dashboard and across manifests; this could be made dynamic, but validating the type of data being passed is beyond the scope of this thesis. The modules then interpret these shared parameters to enable or disable their buttons or other functions. To keep this shared state as lightweight as possible, only the URI of the instances being passed is passed, as visualised above (figure 5.5). Each module then accesses the database directly, managing the connection between the micro front-end and the database ”under the hood”.

5.4 Results and Conclusion

The resulting environment is an application that can federate multiple modules and interact with multiple databases, while still being experienced as an integrated GUI. Interaction between the different micro front-ends is enabled by sharing a state from the dashboard application, allowing an element to be selected in one visualisation module while performing functions directly with another module. The various ”project Pods” serve as rudimentary and lightweight BIM models, with elements serving as generic non-geometric representations. In these Linked Data-based BIM models, semantic information can be added before any geometric representation of the data is present.

However, the dynamic linking of different modules allows such a representation to be added later. Furthermore, in theory, any number of micro front-ends can be created following the same template to extend the environment and further tailor it to the specific needs of a hypothetical stakeholder.

The resulting BIM model can be considered to meet the requirements of LOD 100 as defined by the BIMForum [4]. This provides a proof of concept for initialising a BIM model as described in the proposed BIMification process (section 4.1); where the first step was to include analysis information to create a BIM model with LOD 100. In addition, the prototype provides an implementation of a BIM environment that follows the collaboration schema specified in section 4.2. However, the prototype isn't without its shortcomings, the most important being the lack of validation of the modules or their injected code. I will further evaluate the prototype in the context of the proposed methodology in Chapter 6 (section 6.1).

Chapter 6

Evaluation and Conclusions

This master's thesis proposed an alternative approach to creating a BIM model of an existing building without an up-front and labour-intensive 3D remodelling process. To create this "BIM model without a 3D model", the prospects of using Linked Data to create an extensible BIM environment were explored. This research resulted in a proposal for the BIMification of existing buildings, where the importance of such an extensible BIM was demonstrated. Alongside this data model level workflow, an alternative collaboration level workflow was proposed. This workflow leverages the ability to link information from the Resource Description Framework (RDF) to create a BIM environment that is highly modular and highly specialised; while eliminating the need for duplication of information. Finally, the metadata required to enable each module in such a modular environment was explored. This led to the proposal of an amendment to the Micro Front-End Store Ontology (Mifesto), an ontology for describing micro front-ends for the purpose of dynamic implementation.

A prototype has been developed as a proof of concept for this proposed BIM approach. The prototype will be evaluated in terms of its internal workings and the interaction between the user and the database (section 6.1). This will lead to a reflection on the proposed methodology and ontology (section 6.2) in the context of the original question of creating a BIM model without 3D representation. The reflection highlights the strengths and weaknesses of the approach and raises questions that have remained unanswered. The latter is then extended to possible future research topics or work to be done in order to put the methodology into practice (section 6.3). Finally, the thesis concludes (section 6.4) with a concise answer to the original research questions posed in section 1.2.2.

6.1 Evaluation of the prototype

The prototype was developed to test the practical implementation of the methodology proposed in Chapter 4. Specifically, the aim was to create a web application that uses a Mifesto store to insert micro front-ends. There was also an ambition to be able to create an extensible BIM environment that could create a rudimentary data model for Building Information Management. In the following sections, first, each component will be evaluated separately in context of their intended role (section 6.1.1). The interaction between these components is then discussed in the context of user experience (section 6.1.2).

6.1.1 the components separately

The dashboard is the first application the user interacts with and is the environment through which the other components are accessed. Its role is to allow the user to login to their Pod, to provide an overview of the data in a project and to provide access to the store component. The resulting UI is a sequence of pages that is in line with more traditional CAD applications: first a login, then an overview of projects, and then the project itself, together with the functionalities provided by the environment. The dashboard manages to provide the information we need relatively quickly, and it fulfils its basic requirements: access to the store, the database and external modules. However, it falls short of providing a comprehensive BIM environment, as it is currently not implemented to provide anything other than conceptual instances without the implementation of external modules. In addition, the dashboard needs to reload the entire page to update the retrieved data, which affects the usability.

Next is the store, accessed via a ”project page” on the dashboard. The role of the store can be defined as accessing the various manifests, displaying the relevant information and providing the information required to implement the external modules. The store that is able to retrieve and display the relevant information to the user. However, for a completely smooth user experience, the micro front-end should be loaded when the user requests it, not just when the page loads. In addition, if multiple catalogues are aggregated into the store, the modules should be grouped by catalogue or provide a way to filter through the modules. Finally, a search function should be implemented, along with the ability to retrieve a specific manifest using the URI as input.

The external micro front-ends act as a separate application from which specific modules can be implemented into the dashboard. The overall aim of these modules is to add specialised functionality to the environment created by the dashboard. In the prototype, two micro front-ends were developed, resulting in two functionality modules and one visualisation module.

When implemented correctly, the modules are indistinguishable from the functionalities provided by the bare-bones dashboard. The micro front-ends themselves manage to provide specific functionalities to the dashboard. However, in the current application, there is still a considerable amount of parameter definitions that needs to overlap in order to implement the different modules.

Finally, there is the database where the data is stored, in this case implemented as a different Solid Pod for each project. The database's purpose is to provide a common data environment that can become a BIM model or even a digital twin. The database itself doesn't really do anything but allows the individual components of the prototype environment to create a rudimentary BIM model that meets the requirements of LOD 100. This could then be further developed into a full digital twin of the described building. However, this implies that the various Pods on which the BIM data is stored will need to be maintained by multiple parties over an extended period of time.

6.1.2 the interaction between components

In the prototype there are three types of interaction, between the components and the database, between the dashboard, the store and the modules, and between the modules themselves. Each of these interactions should be smooth and without any technical knowledge on the part of the user, in order to provide the user experience of a single monolithic application.

The interaction between the database and the different components takes place to modify the graph stored on the database. This is done completely "under the hood" and is triggered by user input, e.g. clicking a button to execute a SPARQL query. This allows the user to perform complex functions without any knowledge of query principles or the underlying data structure.

The interaction between the dashboard, the store and the external micro front-ends is established to allow the user to choose which micro front-ends to use to extend the capabilities of the environment. This provides the user experience of any other app-store, however, instead of downloading the external micro front-ends, the separate components are loaded in each time the dashboard is opened. As the elements returned by the micro front-ends are all HTML elements, it is possible to style them relative to the parent element in which they are implemented. This is used to create a consistent layout across the heterogeneous components.

Then the interaction between the modules themselves is required to provide a smooth user experience in which the different components behave as if they were part of the same ecosystem. These interactions are managed by the dashboard by providing an overarching parameter structure. In this way, it is possible to select an element in one module and perform a function on it with another module. For example, in the 3D renderer, the user can select a geometry and directly add a damage instance to the corresponding element in the graph.

When these interactions are combined, an environment is created in with a smooth and ”low barrier of entry” user experience. In addition does this setup allow for other components to be added into the environment as long as they follow the same template. The resulting BIM environment can thus be extended and adapted to any specific need or workflow. Furthermore does the aggregation of any micro front-end allow for many-to-many enrichment.

The latter meaning that there could be multiple environments, in which various micro front-ends could be used in order to contribute to a BIM model which is built up from a variety of applications. However, the downside of allowing any application to modify the BIM model, is that the environment is prone to malware. Hence it is important that the actual code of the modules can be validated before implementation outside of a prototype environment.

6.2 Reflection on the methodology and proposed ontology

To evaluate the methodology, each part is discussed separately in terms of its strengths, weaknesses and practical implications (sections 6.2.1, 6.2.2 & 6.2.3). The possibilities and consequences of BIM without a 3D model are then discussed (section 6.2.4). Finally, there is a reflection upon data ownership and intellectual property in the context of a federated BIM model (section 6.2.5).

6.2.1 Process of BIMification

The process of BIMification, with the addition of increasing LOD throughout the design process, has been introduced to allow the BIM model to gradually increase in complexity. This was done by shifting to an information-first BIM approach, which allows for a more traditional design process where information is gathered before any drawing or modelling is required. In addition, this approach allows partial sets of data to be retained, e.g. a certain part of the model is left at a lower LOD because no works need to be carried out there.

However, as the elements are semantically enriched before any representation for these elements is enforced, the BIM model becomes harder to interpret. This leads to the importance of naming conventions for the conceptual elements in order to achieve consensus among all stakeholders. In addition, because the semantic model is built without a large remodelling effort, the existence of information is more uncertain than in a traditional BIM model.

For the practical implementation of the BIMification process, the main disadvantage of the framework is that it is difficult to implement with the current industry leaders, or at least requires some conversion work. However, the incremental nature of the framework allows this implementation to be done piece by piece. For example, a plugin could be developed to add information to the BIM model that wouldn't otherwise be included.

6.2.2 Extensible BIM Environment

The extensible BIM environment was proposed to allow the digital environment to be fully customised to various needs and workflows. The resulting application created an integrated user experience despite being an aggregation of several applications and micro front-ends; as shown in the prototype (Chapter 5). The main advantages of such an environment are that it allows any micro front-end to modify the BIM model and, by using Linked Data, it allows multiple datasets to be linked together to form a federated BIM model.

The downside to such an extensible environment is that the resulting BIM approach becomes more complex to get started with. To elaborate, if the environment becomes extensible, it will require more work to tailor it to a specific workflow.

However, the benefits of a practical implementation of such an extensible BIM environment for its users are staggering. As mentioned earlier (section 6.1.2), the environment allows for many-to-many enrichment, meaning that the user can use any application to federate any micro front-end to edit any BIM model. This means that the user has full control over the functionality they want to include in their application, allowing for greater market competition rather than being locked into a specific ecosystem. In addition, the framework allows for the combination of different geometry formats and is able to extend beyond the construction industry. For example, information about the heating system could be included, which then serves a function for the management of the building. Finally, with some minor adaptations, the framework already has some market-ready options; in the context of the BIM4Ren research [19], several services have been developed that take a similar approach to the proposed environment.

6.2.3 Mifesto Ontology

The proposed revision of the Mifesto ontology by Werbrouck et al. (2022) [31] has been developed to allow the federation of micro front-ends into other applications. The revision introduced the differentiation between functionality modules and visualisation modules and managed the complexity of in- and output parameters. In addition, the revision includes more definitions that would previously refer to other ontologies. Although this somewhat undermines the concept of linking definitions, it provides a more comprehensive ontology for developers new to Linked Data. To address the duplication of definitions, these newly included instances are defined as sub-instances of the original definition, allowing machines to reason that they are the same.

The revision did not include the definitions for describing the configuration of the modules, as it's assumed that in web design the modules are styled relative to the parent container. However, this relatively short configuration description may still be present in an eventually published ontology, it is simply not considered necessary within this thesis. Furthermore, the revision adds a bit of complexity by providing a deeper nesting of instances for the description of the actual code of the module. It could be argued that these descriptions could be directly linked to the actual release of the module.

The practical implementation of this revision to the Mifesto ontology could be easily adopted, as it doesn't fundamentally change the goal of the ontology. However, the current version of the revision provides an instance to describe the origin of a resource within a RDF graph. Ideally, this definition should be added to the DCAT ontology, an ontology for describing datasets, but to provide a temporary solution, the instances are added to the Mifesto. Finally, any practical implementation of a Mifesto store should provide validation of both the output and the technology stack used to build the micro front-ends. This is because the current iteration is relying on the manifest to be true to the actual code of the application.

6.2.4 BIM without a 3D model

As the BIM model moves away from a "model first" approach to a Linked Data-based BIM, its focus shifts completely to Building Information Management. This allows any type of representation of elements to be added to the BIM model. Although in practice, given the nature of the AEC industry, this will most likely be either a textual format, a plan or a 3D model. However, when all three can be combined into a single BIM model, each plays a different role throughout the design process. By using an open and extensible data format, each stakeholder can contribute directly to the BIM model.

This eliminates the need for handovers as the different data sets are federated into one large digital twin. Instead, quality checks are put in place to evaluate the state and progress of the federated BIM model and thus the execution plans. Furthermore, the adoption of a linked data-based data model allows the BIM model to be machine interpretable, opening up the possibility of machine learning on multiple BIM models.

However, moving away from 3D representation, requires more elaborate BIM management plans to ensure that all stakeholders have a consensus on naming conventions. Without this consensus, it is easy to imagine that two stakeholders have described the same element differently, resulting in a duplicate element in the BIM model.

6.2.5 Data ownership

When the digital twin is composed of multiple datasets, many questions arise regarding the management of this federation of documents and its intellectual property. In theory, intellectual property is easier to define when each stakeholder contributes to the BIM model with their own dataset. However, this assumes that either the data only needs to last until the end of the construction project, or that each stakeholder needs to maintain their dataset for an extended period of time. The former is problematic in terms of sustainable information, i.e. ideally the data will be useful afterwards for the management of the building or for a eventual renovation. The latter option is virtually unsustainable as it is both difficult and expensive to ensure that the data is properly maintained. The question then becomes: When does intellectual property end? Is there a point at which the data becomes the property of the building itself or its owner?

6.3 Future work

As mentioned in the previous section 6.2, there are many aspects that remain unexplored, both for BIM without geometry and for an extensible BIM environment. The most recurring theme throughout the evaluation is the need for compliance checking in external micro front-ends. This would make the extensible BIM environment a more secure and navigable workspace.

In addition, the potential for many-to-many enrichment has been raised as one of the benefits of the extensible BIM environment. This is largely an ideological assumption, namely that a wider open market is beneficial to the users of the software. However, the concrete contributions of many-to-many enrichment remain to be defined. This, in turn, would fit into the research discourse on the benefits, albeit financially, of BIM in the AEC industry and design.

Another valid and relatively pressing issue is the research discourse on intellectual property within the digital space of Architecture, Engineering and Construction. This thesis hasn't touched on this issue except for a small reflection. However, as our built, physical environment becomes increasingly intertwined with its digital counterpart, data ownership and intellectual property are becoming increasingly important issues. In addition, this research could be extended to the role of BIM in different collaborative structures and thus responsibility structures.

Finally, the proposed framework for an extensible BIM environment could serve as a starting point for creating a platform for BIM-based micro front-ends. This could be the starting point for an "Advanced Topic" course to introduce students to Linked BIM and Building Informatics in general; by providing an environment in which they could implement self-built applications.

6.4 Conclusion

In the introduction of this thesis two research questions were posed (section 1.2.2), namely "How to create a BIM model without a 3D model?" and "What metadata is required for implementing micro services in a Linked Data based BIM?". Throughout, a methodology has been developed to answer both questions. The methodology starts with the use of a BIMification process to initiate a BIM model from an analysis of the building, thus creating an inherently information or data only BIM model. Then to enable this BIMification an extensible BIM environment was needed. For this, Linked Data was used in combination with micro front-ends to implement additional functionalities into an overarching application, which leads to the second research question. To provide the required metadata of these micro front-ends, a revision of the Micro Front-End Store Ontology was then proposed. In this ontology, each micro front-end has a manifest that details how the module should be implemented, together with a human-readable description of the functionalities of said module.

This thesis doesn't provide all the answers within the research discourse on Linked Data in BIM (environments). Although, I hope it provides some starting points for other students to venture into the world of BIM development.

Bibliography

[1] Tim Berners-Lee. Semantic web roadmap. https://www.w3.org/DesignIssues/Semantic.html, 1998. Accessed on: 2023-05-06.

[2] Tim Berners-Lee. Linked data. https://www.w3.org/DesignIssues/LinkedData.html, 2009. Accessed on: 2023-05-06.

[3] BIM4Ren project. Bim4ren. https://bim4ren.eu/. Accessed: 2023-03-28.

[4] BIMForum. *Level of Development (LOD) Specification*, 12 2021.

[5] Mathias Bonduel. *A Framework for a Linked Data-based Heritage BIM*. PhD thesis, KULeuven, 2021.

[6] Mathias Bonduel, Jyrki Oraskari, Pieter Pauwels, Maarten Vergauwen, and Ralf Klein. The IFC to Linked Building Data Converter - Current Status. *6th Linked Data in Architecture and Construction Workshop*, 2018.

[7] Pierre Bourreau, Nathalie Charbel, Jeroen Werbrouck, Madhumitha Senthilvel, Pieter Pauwels, and Jakob Beetz. Multiple inheritance for a modular BIM. *BIM4Ren*, 2020.

[8] BuildingSMART International. Industry foundation classes (ifc). https://technical.buildingsmart.org/standards/ifc. Accessed: 2023-04-13.

[9] Richard Cyganiak et al. 1.1 graph-based data model. https://www.w3.org/TR/rdf11-concepts/#data-model, 2014. Accessed on: 2023-05-06.

[10] Richard Cyganiak et al. 1.4 rdf vocabularies and namespace iris. https://www.w3.org/TR/rdf11-concepts/#vocabularies, 2014. Accessed on: 2023-05-06.

[11] Richard Cyganiak et al. Rdf 1.1 concepts and abstract syntax. https://www.w3.org/TR/rdf11-concepts/, 2014. Accessed on: 2023-05-06.

[12] Antoine Dugué et al. B4R D7.4 Beta Testing. Technical report, BIM4Ren, 2022.

[13] Paula Gearon et al. Sparql 1.1 update. https://www.w3.org/TR/sparql11-update/, 2013. Accessed on: 2023-05-06.

[14] Al-Hakam Hamdan, Mathias Bonduel, and Raimar Joseph Scherer. An ontological model for the representation of damage to constructions, 2019-06-25.

[15] Al-Hakam Hamdan, Mathias Bonduel, and Raimar Joseph Scherer. An ontological model for the representation of damage to constructions, 2019-06-25.

[16] Steve Harris et al. Sparql 1.1 query language. https://www.w3.org/TR/sparql11-query/, 2013. Accessed on: 2023-05-06.

[17] Jack Herrington and Zack Jackson. Practical module federation. Sold at https://module-federation.github.io/, November 2020.

[18] KROQI platform. KROQI - la platforme de travail collaboratif. https://kroqi.fr/en/. Accessed: 2023-03-28.

[19] Lennard, Zia - R2M Solution. B4R D8.6 Replication Plan. Technical report, BIM4Ren, 2022.

[20] Pauwels, Pieter and Zhang, Sijie and Lee, Yong-Cheol. Semantic web technologies in AEC industry : a literature review. *AUTOMATION IN CONSTRUCTION*, 73:145–165, 2017.

[21] Severi Peltonen, Luca Mezzalira, and Davide Taibi. Motivations, benefits, and issues for adopting micro-frontends: A multivocal literature review. *Information and Software Technology*, 136:106571, 2021. https://doi.org/10.1016/j.infsof.2021.106571.

[22] Eric Prud'hommeaux et al. Rdf 1.1 turtle. https://www.w3.org/TR/turtle/, 2014. Accessed on: 2023-05-06.

[23] Mads Holten Rasmussen. The vision of a decentralized, distributed aec information infrastructure using lbd technologies, 2018. https://drive.google.com/file/d/1MWbwh3ihzhMtF_pj0G0t3XYP0glJ7v5o/view.

[24] Mads Holten Rasmussen, Maxime Lefrançois, Georg Ferdinand Schneider, and Pieter Pauwels. BOT: The building topology ontology of the W3C linked building data group. *Semantic Web*, 12(1):143–161, November 2020.

[25] Andrei Vlad Sambra, Essam Mansour, Sandro Hawke, Maged Zereba, Nicola Greco, Abdurrahman Ghanem, Dmitri Zagidulin, Ashraf Aboulnaga, and Tim Berners-Lee. Solid: A Platform for Decentralized Social Applications Based on Linked Data. Technical report, Decentralized Information Group of CSAIL at MIT, 2016.

[26] Raimar J. Scherer and Peter Katranuschkov. BIMification: How to create and use BIM for retrofitting. *Advanced Engineering Informatics*, 38:54–66, October 2018.

[27] Ruben Verborgh. Solid: innovation through personal data control. https://rubenverborgh.github.io/ECA-2021/, 2021. Accessed: 2023-05-05.

[28] Rebekka Volk, Julian Stengel, and Frank Schultmann. Building information modeling (bim) for existing buildings—literature review and future needs. *Automation in construction*, 38:109–127, 2014.

[29] Jeroen Werbrouck. ConSolid vocabulary. https://w3id.org/consolid#, 2022. Accessed: 2023-05-05.

[30] Jeroen Werbrouck, Pieter Pauwels, Jakob Beetz, and Mannens Erik. Consolid: a federated ecosystem for heterogeneous multi-stakeholder projects. (in press), 2023.

[31] Jeroen Werbrouck, Pieter Pauwels, Jakob Beetz, and Erik Mannens. Practical Semantic Enrichment of AEC Multi Models with Dynamic Micro Frontend Configurations. Manuscript under review, December 2022.

[32] Wikipedia Contributors. Resource Description Framework. https://en.wikipedia.org/w/index.php?title=Resource_Description_Framework&oldid=1152602887, 2023. Accessed on: 2023-05-06.

Appendix A

Revised Mifesto (*.ttl)

```
## the relevant prefixes
@prefix mifestoRM: <http://localhost:5000/Tester/ontology/mifestoRM#>.
## this prefix points towards this document and should be altered if the ontology would be published

@prefix owl: <http://www.w3.org/2002/07/owl#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix dcat: <http://www.w3.org/ns/dcat>.
@prefix dct: <http://purl.org/dc/terms/>.
@prefix dcterms: <http://purl.org/dc/terms/>.
@prefix foaf: <http://xmlns.com/foaf/0.1/>.
@prefix pav: <http://purl.org/pav/>.
@prefix doap: <http://usefulinc.com/ns/doap#>.
@prefix fno: <https://w3id.org/function/ontology#>.


## some metadata about this document
mifestoRM: a owl:Ontology;
    dcterms:title "MIcro FrontEnd Store Ontology";
    dcterms:description """
    *This is a draft document and ontology*
    The "MIcro FrontEnd Store Ontology" as revised by Robbe Michiels; originally described by Jeroen Werbrouck et al. (2022).
    The ontology is aimed at describing micro frontend applications which handle rdf structured data.
    The goal is to create collections of these applications in such a manner they can be searched and displayed in some sort of "app store".
    """@en.


```

```
## the classes for the store
mifestoRM:Store rdfs:subClassOf dcat:Catalog,
    rdfs:label "Store";
    rdfs:comment "The store is a container resource (dcat:Catalog) that
   references either mifesto Manifests or other Catalogs.".

## the store properties
## *use normal dcat properties, they already provide an intricate description
    for datasets and catalogs*


# the class for the manifest
mifestoRM:Manifest rdf:type rdfs:Class;
    rdfs:label "Manifest";
    rdfs:comment "A self-describing document regarding the necessary metadata
    to incorparate a module through a mifesto Store. It provides information
   on the inner-workings of modules and human readable descriptions in order
   to discover the module.".

## the manifest properties
mifestoRM:moduleName rdfs:subPropertyOf rdfs:label;
    rdfs:label "moduleName";
    rdfs:comment "Provides a human readable name of the module, intended to
   be displayed in the Store.";
    rdfs:domain mifestoRM:Manifest.

mifestoRM:moduleDescription rdfs:subPropertyOf rdfs:comment;
    rdfs:label "moduleDescription";
    rdfs:comment "Provides a human readable description of the module,
   intended to describe the module's functionality in the Store.";
    rdfs:domain mifestoRM:Manifest.

mifestoRM:hasDepiction rdfs:subPropertyOf foaf:depiction;
    rdfs:label "hasDepiction";
    rdfs:comment "Provides an image related to the module to be displayed in
   the store. This could be a logo or an image from a part of the module. It
   is the intention that this points towards a URL of the image. This
   property is defined as foaf:depiction.";
    rdfs:domain mifestoRM:Manifest.

mifestoRM:hasFunctionModule rdf:type rdf:Property;
    rdfs:label "hasFunctionModule";
    rdfs:comment "Describes that the manifest has a functionality module.";
```

```
    rdfs:domain mifestoRM:Manifest;
    rdfs:range mifestoRM:FunctionModule.

mifestoRM:hasViewerModule rdf:type rdf:Property;
    rdfs:label "hasViewerModule";
    rdfs:comment "Describes that the manifest has a visualisation module.";
    rdfs:domain mifestoRM:Manifest;
    rdfs:range mifestoRM:ViewerModule.


## the classes for either the functionality or visualisation modules
mifestoRM:FunctionModule rdf:type rdfs:Class;
    rdfs:label "FunctionModule";
    rdfs:comment "This Class describes a functionality module attached to the
    Manifest, e.g. a module containing primarily buttons with functions
   attached to them.".

mifestoRM:ViewerModule rdf:type rdfs:Class;
    rdfs:label "ViewerModule";
    rdfs:comment "This Class describes a visualisation module attached to the
    Manifest. This is meant for any module which visualises the (partial)
   graph or its items, e.g. a renderer which displays all geometry attached
   to items in a certain dataset.".

## the properties of the functionality or visualisation modules
mifestoRM:hasVersion rdfs:subPropertyOf pav:hasVersion;
    rdfs:label "hasVersion";
    rdfs:comment "Describes that the module has a pubic release or version
   and thus points to a mifesto Release object. This property has been
   defined as pav:hasVersion, with a range of mifesto Release.";
    rdfs:domain mifestoRM:FunctionModule, mifestoRM:ViewerModule;
    rdfs:range mifestoRM:Release.


## The class for the release of a module
mifestoRM:Release rdf:type rdfs:Class;
    rdfs:label "Release";
    rdfs:comment "This Class describes a public release or version of the
   module.".

## the Release properties
mifestoRM:versionNumber rdfs:subPropertyOf doap:revision;
    rdfs:label "versionNumber";
```

```
    rdfs:comment "Holds a literal describing the structured version number or label.".

mifestoRM:hasModule rdf:type rdf:Property;
    rdfs:label "hasModule";
    rdfs:comment "Describes that the Release has an actual Module which can be implemented.";
    rdfs:domain mifestoRM:Release;
    rdfs:range mifestoRM:Module.

mifestoRM:consumesInput rdf:type rdf:Property;
    rdfs:label "consumesInput";
    rdfs:comment "Describes that the released module needs a parameter it uses as input.";
    rdfs:range fno:Parameter.

mifestoRM:expectedOutput rdf:type rdf:Property;
    rdfs:label "expectedOutput";
    rdfs:comment "Describes that the released module will generate an output.";
    rdfs:range fno:Output.

mifestoRM:outputDescription rdfs:subPropertyOf rdf:comment;
    rdfs:label "outputDescription";
    rdfs:comment "Provides a human readable description of the output the module will generate. It is intended that this is a technical description of the alterations or additions the module will make to the provided graph.";
    rdfs:domain fno:Parameter.

mifestoRM:hasModule rdf:type rdf:Property;
    rdfs:label "hasModule";
    rdfs:comment "Describes that the Release has a Module with the injectable code.";
    rdfs:domain mifestoRM:Release;
    rdfs:range mifestoRM:Module.


## the class for the Module with the injectable code
mifestoRM:Module rdf:type rdfs:Class;
    rdfs:label "Module";
    rdfs:comment "This Class describes the actual module and the injectable code attached to the corresponding release and the manifest.".
```

```
130
131 ## the module properties
132 mifestoRM:isExposedOn rdf:type rdfs:Property;
133     rdfs:label "isExposedOn";
134     rdfs:comment "Provides the URL where the injectable code of the module is
    exposed. In the current iteration, the isExposedOn property only allows
    literal values.";
135     rdfs:domain mifestoRM:Module, mifestoRM:Release, mifestoRM:FunctionModule
    , mifestoRM:ViewerModule;
136     rdfs:range rdfs:Literal.
137
138
139 ## the origin meta-data classes
140 mifestoRM:Origin rdf:type rdfs:Class;
141     rdfs:label "Origin";
142     rdfs:comment "This class describes the origin of a resource.".
143
144 ## the meta-data properties
145 mifestoRM:hasOrigin rdf:type rdf:Property;
146     rdfs:label "hasOrigin";
147     rdfs:comment "Describes that a resource originates from a certain micro
    front-end or application.";
148     rdfs:range mifestoRM:Origin.
149
150 mifestoRM:originatesFrom rdf:type rdf:Property;
151     rdfs:label "originatesFrom";
152     rdfs:comment "Describes which micro front-end or application generated
    the resource, by referencing its Manifest.";
153     rdfs:domain mifestoRM:Origin;
154     rdfs:range mifestoRM:Manifest.
155
156 mifestoRM:addedBy rdf:type rdf:Property;
157     rdfs:label "addedBy";
158     rdfs:comment "Provides the person by whom the resource was added to the
    graph, by refering to their personal document.";
159     rdfs:range foaf:Person.
160
161 mifestoRM:addedOn rdfs:subPropertyOf dcterms:dateSubmitted;
162     rdfs:label "addedOn";
163     rdfs:comment "Provides when the resource was added to the graph.".
```

Appendix B

Example Manifest for a 3D rendering micro front-end (.ttl)

```
@prefix consolid: <https://w3id.org/consolid#>.
@prefix dcat: <http://www.w3.org/ns/dcat#>.
@prefix dct: <http://purl.org/dc/terms/>.
@prefix bot: <https://w3id.org/bot#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix mifestoRM: <http://localhost:5000/Tester/ontology/mifestoRM#>.

<#manifest> a mifesto:Manifest;
        mifesto:moduleName "3d_renderer";
        mifesto:moduleDescription "Renders the 3D geometries (specifically in
    gltf format) of the project. Also includes functions to add these
   geometry to any element in the graph.";
        mifesto:hasViewerModule <#viewerModule>;
        mifesto:hasFunctionModule <#functionModule>.

<#viewerModule> a mifesto:ViewerModule;
    mifesto:hasVersion <#release_viewerModule_1>.

<#release_viewerModule_1> a mifesto:Release;
    mifesto:versionNumber 0.1;
    mifesto:consumesInput <#_datasetParameter>,<#_itemParameter>;
    mifesto:hasModule <#module_viewerModule>.

<#module_viewerModule> a mifesto:Module;
    rdfs:label "./viewer";
    mifesto:isExposedOn "http://localhost:3100/remoteEntry.js".

<#functionModule> a mifesto:FunctionModule;
```

```
    mifesto:hasVersion <#release_functionModule_1>.

<#release_functionModule_1> a mifesto:Release;
    mifesto:versionNumber 0.1;
    mifesto:consumesInput <#_datasetParameter>,<#_itemParameter>;
    mifesto:hasModule <#module_functionModule>.

<#module_functionModule> a mifesto:Module;
    rdfs:label "./functions";
    mifesto:isExposedOn "http://localhost:3100/remoteEntry.js".

<#_datasetParameter> a fno:Parameter;
    rdfs:label "database";
    fno:type consolid:Project;
    fno:required "true"^^xsd:boolean.

<#_itemParameter> a fno:Parameter;
    rdfs:label "item";
    fon:type bot:Element;
    fno:required "true"^^xsd:boolean.
```