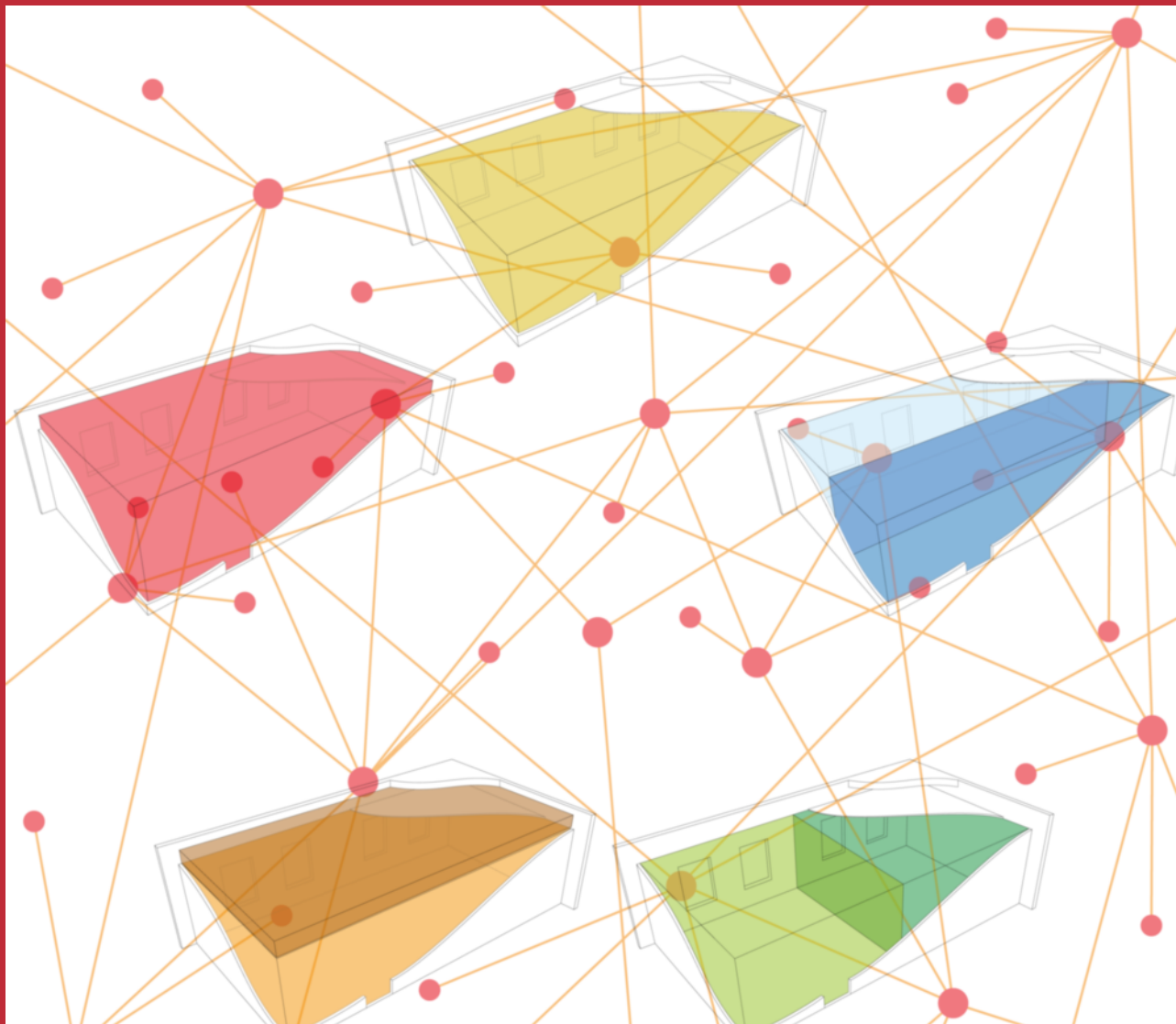
DTU Civil Engineering
PhD Thesis R-419

Digital Infrastructure and Building Information Modeling in the Design and Planning of Building Services

Mads Holten Rasmussen



Technical University of Denmark
Department of Civil Engineering
Brovej, building 118,
2800 Kongens Lyngby, Denmark
Phone +45 4525 1700
byg@byg.dtu.dk
www.byg.dtu.dk

Preface

This thesis is submitted to the Department of Civil Engineering at the Technical University of Denmark (DTU) in partial fulfilment of the requirements for the degree of Doctor of Philosophy. The study was conducted as an industrial PhD project* between January 2016 and January 2019. The main supervisor was Associate Professor Jan Karlshøj, DTU Civil Engineering; the co-supervisors were Associate Professor Christian Anker Hviid, DTU Civil Engineering, and Assistant Professor Pieter Pauwels, Ghent University (UGent) Department of Architecture and Urban Planning; company supervisor was Senior Engineer Morten Vammen Vendelboe, Niras. The dissertation is paper-based and consists of the present thesis and the papers prepared (see the List of Papers).

Kgs. Lyngby, January 14, 2019

Mads Holten Rasmussen
PhD candidate

*An Industrial PhD is a three-year industrially focused research project and PhD education which is carried out in a collaboration between a company, an Industrial PhD candidate and a university. For more information, see https://innovationsfonden.dk/en/application/erhvervsphd

Foreword

While doing project engineering, I often felt frustration over many of the work processes we do as engineers. Most processes are based on documents and generally there is no coupling between these documents even though the data they contain is highly interdependent. When working in such an environment the most noble job of an engineer simply consists in putting out fires. Changes occur rapidly, and the resulting amount of rework stacks up. Tasks were solved quickly from a short-term perspective, but the swiftness turns into a nightmare in the long run as the project progresses. Figure 1 by Alan O'Rourke illustrates the feeling I had quite well – a feeling that is probably shared by many of my fellow colleagues in the engineering business.

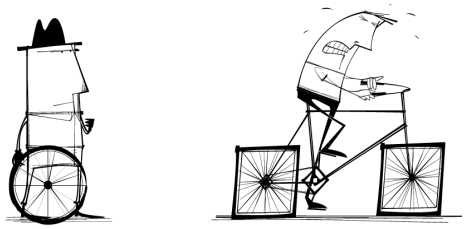


Figure 1: Too Busy To Improve by Alan O'Rourke.

At this point, I had some limited self-taught experience with web development, databases and scripting, so I had a basic technical understanding of what we were doing was not cleaver. However, my solutions to the problem were restricted to data transfers between AutoDesk Revit and Microsoft Excel – solutions that was proven to be vulnerable as I was the only one who could operate them. I

did not have time to develop a better linking between the tools, and when seen in hindsight, it would probably not have been the right approach anyway. The quote below indicates why in a quite straightforward manner:

> *"When you digitize a shitty process, what you get is a shitty digitized process" (Thorsten Dirks, CEO of Telefonica, 2016)*

It is a widespread practice to develop tools that focus on a specific task rather than zooming out and assess the full system of processes in which the particular task exists. Several software tools on the market promise to enhance Building Information Modelling (BIM) work flows, but most of them are, too narrow scoped to solve this complex challenge. Therefore, enabling interconnectivity becomes reduced to handling data transfers between proprietary software solutions rather than making sure that all the data which is generated is understandable at a global level.

When I had the opportunity to investigate the challenge in detail through a PhD-project, I therefore decided to go with it pursue my vision.

Summary (English)

The construction industry is a fragmented industry where several actors from different companies consume, process and further develop the common project material. The material forms the basis for the execution of a construction project and is created over the full course of the design. The design process is, however, challenged, partly as a result of the following characteristics:

1. Information storage is primarily document-centric, and Building Information Modelling (BIM) tools are mainly used for coordination and production of traditional design documentation in the form of 2D drawings and quantity lists.
2. Data exchanges are handled through the exchange of whole documents rather than specific data.
3. A substantial part of the knowledge built up is only stored in the minds of the project participants who become an indispensable information source over time.
4. It takes several years to carry out a construction project, and over this period of time there is inevitably a replacement of employees.

There are several derived consequences of these characteristics. Information retrieval from static documents is time consuming. It is difficult to establish dynamic links between derived information, and thus it is only possible to reuse work to a limited extent in the next project. Also, inconsistency in the design material is difficult to avoid over time, and it is hard to assess the consequences of a design change. Finally, the success of a project largely depends on the employee replacement and the enclosed loss of knowledge.

The purpose of this thesis is to investigate the possibilities of creating a web-based infrastructure that supports the need for interdisciplinary information exchange in the construction industry. This in a way that allows the distributed project material to grow organically with the construction project and be made available to any authorised actor. This way, insights and traceability are created as the project progresses. The intention is that increased transparency and accessibility of relevant project information will ensure that the design can proceed more smoothly. The overall reference case is the design of building installations.

Semantic web technologies make it possible to conceptually describe knowledge of objects from the real world, and in this project, it is investigated how they can constitute the requested infrastructure. Describing knowledge about an object is achieved partly by classifying it and partly by specifying properties and relationships with other objects. This is accomplished in an unambiguous manner ensuring that the stated knowledge is machine-readable. Thereby, accessibility is increased, and it becomes possible to reuse knowledge and automate the inference of new knowledge. Classes and relationships are described in ontologies that constitute vocabulary and terminology to describe a particular domain. The first contribution to an improved interdisciplinary information exchange using these technologies is a proposal for a minimal ontology, which describes the main topological principles of a building. This enables the description of the spatial and physically tangible components that constitute a building in a general manner, that can be extended with domain-specific terminology as needed. The second contribution is an ontology with terminology to handle complex design properties that change over time, have varying reliability and may be derived from other properties. Together, the two ontologies form the basic framework of the infrastructure that will drive the information exchange.

Through software architectural considerations and implementations, it is demonstrated how a decentralised web-based information exchange can be managed in the construction industry of the future. This is done, for example, by establishing data sets from BIM tools and Industry Foundation Classes (IFC) and demonstrating how this data set can be expanded through external tools. Furthermore, interaction with the model is investigated in more or less sophisticated ways that align with the workflows that lie in designing a building. Thereby, it is demonstrated what advantages can lie in the industry moving from the current document-centric practice to a more data-centric practice. This is the very core of the semantic web, which is also described as a "web of data". Potentially, the technologies can induce a paradigm shift in the design practice of today if they are implemented correctly. This practice is expected to result in buildings with fewer faults and with greater knowledge recycling in future projects. Smooth interaction with the data model is a precondition for the desired paradigm shift to be realised, and further research is needed in this area.

Summary (Danish)

Byggebranchen er en fragmenteret branche, hvor flere aktører fra forskellige virksomheder forbruger, bearbejder og videreudvikler et fælles projektmateriale. Materialet danner grundlag for udførelse af et byggeprojekt, og skabes over projekteringens fulde forløb. Projekteringsforløbet er imidlertid udfordret, hvor følgende karaktertræk blandt andet gør sig gældende:

1. Informationslagring er primært dokument-centreret og Building Information Modelling (BIM)-værktøjer bruges primært til koordinering og produktion af dokumentation i form af arbejdstegninger og mængdelister.
2. Dataudvekslinger håndteres gennem udvekslingen af hele dokumenter snarere end specifik data.
3. En væsentlig del af vidensopbygningen gemmes kun i hukommelsen på projektdeltagerne som derved bliver en uundværlig informationskilde over tid.
4. Det tager flere år at gennemføre et byggeprojekt, og over denne tidsperiode sker der uundgåeligt en udskiftning af medarbejdere.

Der er flere uhensigtsmæssige afledte konsekvenser af disse karakteristikker. Det er tidskrævende at fremskaffe informationer fra designmaterialet. Det er svært at etablere dynamiske links imellem afledte informationer, og dermed er det kun i begrænset omfang muligt at genbruge arbejde i det næste projekt. Desuden er inkonsistens i designmaterialet svært at undgå over tid, og det er vanskeligt at gennemskue konsekvenserne af en designændring. Til slut afhænger et projekts succes i høj grad af medarbejderudskiftningen og det medfølgende videntab.

Formålet med denne afhandling er at undersøge mulighederne for at skabe en web-baseret infrastruktur, som understøtter behovet for tværfaglig informationsudveksling i byggebranchen. Dette på en måde som tillader, at det distribuerede projektmateriale vokser organisk med byggeprojektet og gøres tilgængeligt for enhver autoriseret aktør. På denne måde skabes indsigt og sporbarhed i takt med projektets fremgang. Det er hensigten at øget transparens og tilgængelighed af relevante projektinformationer skal sikre, at projekteringen kan forløbe mere gnidningsfrit. I projektet tages udgangspunkt i projektering af bygningsinstallationer.

Semantiske web-teknologier gør det muligt at beskrive viden om objekter fra den virkelige verden konceptuelt, og i dette projekt undersøges det, hvordan de kan udgøre den efterspurgte infrastruktur. Beskrivelse af viden om et objekt opnås dels ved at klassificere dette og dels ved at angive egenskaber og relationer til andre objekter. Dette gøres på en utvetydig måde som sikrer, at den angivne viden er maskinlæsbar, og dermed øges tilgængeligheden og muligheden for genbrug og automatisering. Klasser og relationer beskrives i ontologier, som udgør vokabular og terminologi til at beskrive et særligt domæne. Det første bidrag til en forbedret tværfaglig informationsudveksling ved brug af disse teknologier, er et forslag til en minimal ontologi, som beskriver de væsentligste topologiske principper i en bygning. Denne skal muliggøre, at de rummelige og fysisk håndgribelige bestanddele som indgår i kontekst af en bygning kan beskrives på en generel måde, som kan udvides med mere fagspecifik terminologi efter behov. Det andet bidrag er en ontologi med terminologi til at håndtere komplekse egenskaber i byggeprojekter som ændres over tid, har forskellig pålidelighed og kan være afledt af andre egenskaber. Tilsammen danner de to ontologier grundrammen i den infrastruktur som skal drive informationsudvekslingen.

Gennem software-arkitektoniske overvejelser og implementeringer demonstreres det, hvordan en decentraliseret web-baseret informationsudveksling kan håndteres i fremtidens byggebranche. Dette gøres blandt andet ved at etablere datasæt fra BIM-værktøjer og Industry Foundation Classes (IFC) og demonstrere, hvordan dette datasæt kan udbygges gennem eksterne værktøjer. Ydermere undersøges det, hvordan der kan interageres med modellen på mere eller mindre komplekse måder, som er afstemt med de arbejdsgange, der ligger i at projektere en bygning. Dermed synliggøres det, hvilke fordele der kan ligge i, at branchen bevæger sig fra den nuværende dokument-centrerede praksis til en mere datacentreret praksis. Det er hele kernen i det semantiske web, som også beskrives som et "web of data". Potentielt kan teknologierne medføre et paradigmeskift i den projekteringspraksis, der i dag foregår såfremt disse implementeres korrekt. Denne praksis må forventes at medføre byggerier med færre fejl og med større genbrug af viden i fremtidige projekter. Gnidningsfri interaktion med datamodellen er en forudsætning for, at det ønskede paradigmeskift kan realiseres, og derfor er der behov for videre forskning på dette område.

List of Figures

List of Listings

List of Papers

The thesis is structured as a thesis of publications and consists of the peer reviewed papers listed below. The papers are grouped based on the primary topic which they cover, and for each paper, a brief summary is given. The identifications used for the papers are used throughout the thesis. All papers with Rasmussen, M. H. as main author are attached in Chapter 6.

A simple extendable ontology for a web-based BIM

BOT1 Rasmussen, M. H., Pauwels, P., Hviid, C. A., and Karlshøj, J. (2017b). Proposing a central AEC ontology that allows for domain specific extensions. In F. Bosché, I. Brilakis, and R. Sacks (Eds.), *Proceedings of the Joint Conference on Computing in Construction*, July 4–12, 2017 (Vol. 1). Heraklion, Crete, Greece: Heriot-Watt University. doi:10.24928/jc3-2017/0153

This paper investigates existing ontologies in the scope of buildings and finds that they repetitively describe the main components of a building (building, storeys, spaces). It suggests the Building Topology Ontology (Building Topology Ontology (BOT)) as a minimal, extendable ontology only covering the main concepts of a building and demonstrates linking approaches for domain specific extensions.

BOT2 Rasmussen, M. H., Pauwels, P., Lefrançois, M., Schneider, G. F., Hviid, C., and Karlshøj, J. (2017c). Recent changes in the Building Topology Ontology. *5th Linked Data in Architecture and Construction Workshop*, November 13–15, 2017. Univeristy of Burgundy, Dijon, France. doi:10.13140/RG.2.2.32365.28647

After bringing BOT to the World Wide Web Consortium (W3C) Linked Building Data (LBD) Community Group (W3C LBD Community Group (W3C LBD CG)), some shortcomings of the ontology were identified. This paper describes these in detail and accommodate the issue by adding a set of new classes and properties.

BOT3† Rasmussen, M., Pauwels, P., Hviid, C., and Karlshøj, J. (2019b). The BOT ontology: standards within a decentralised web-based AEC industry. *Automation in construction.* Under review

This paper describes a vision of having various interconnected Knowledge Graphs (KGs) described with BOT and other LBD ontologies. It uses BOT as a reference ontology and documents its latest developments as well as linking principles and methods for generating BOT-compliant data (LBD datasets). Through two use-cases it is demonstrated why a (semantic) web-based Building Information Modelling (BIM) can help bringing down the silos in Today's BIM implementations by allowing the dataset to be distributed and extended. Three BIM models are converted to LBD datasets and reasoning performance as well as file sizes is evaluated for these models. These are further used in a proof of concept where the decentralisation benefits are demonstrated by showing how the architect's dataset can be extended by Indoor Climate and Energy (ICE) and Heating, Ventilation and Air Conditioning (HVAC) engineers using simple queries to deduce new data.

Evolving, interdependent design properties

OPM1 Rasmussen, M. H., Lefrançois, M., Bonduel, M., Hviid, C. A., and Karlshøj, J. (2018c). OPM: an ontology for describing properties that evolve over time. M. Poveda-Villalón, P. Pauwels, and A. Roxin (Eds.), *Proceedings of the 6th Linked Data in Architecture and Construction Workshop*, June 19–21, 2018 (pp. 24–33). CEUR Workshop Proceedings. Accessed September 2018. UCL, London, UK: CEUR-WS.org. Retrieved from http://ceur-ws.org/Vol-2159/03paper.pdf

This paper presents an ontology for property management (Ontology for Property Management (OPM)). This includes terminology to describe multiple property states for a single property, thereby allowing it to change over time while keeping track of its full historical evolution.

REQ Rasmussen, M. H., Bonduel, M., Hviid, C. A., and Karlshøj, J. (2018a). Managing Space Requirements of New Buildings Using Linked Building Data Technologies. *eWork and eBusiness in Architecture, Engineering and Construction (ECPPM 2018)*, September 12–15, 2018 (pp. 399–406). Copenhagen, Denmark: CRC Press

This paper suggests how OPM can be used to describe a requirement for a future property. Specifically, it demonstrates how to describe different space requirements and comparing those to the actual design parameters. Space requirements are described at type level using BOT in combination with Web Ontology Language (OWL) property restrictions.

OPM2[†] Rasmussen, M., Lefrançois, M., Pauwels, P., Hviid, C., and Karlshøj, J. (2019a). Managing interrelated project information in AEC Knowledge Graphs. *Automation in construction.* Under review

This paper describes OPM in a larger perspective and extends the terminology initially presented in paper OPM1. The main contribution is the introduction of classification for 'property reliability' and 'calculations'. The latter provide a formal way to document the reasoning logic behind derived properties. Another contribution is the proposal of a standardised way to generate parametric queries for operating an OPM-compliant Architecture, Engineering and Construction (AEC) KG. A JavaScript based query generator (OPM-QG) that follows these principles is presented. With an implementation it is demonstrated how OPM

[†]Under review

can be used to accomplish a design task on a construction project. The specific implementation deals with calculation of space heating demands and demonstrates how OPM keeps track of the interdependencies so that consequences of a design change can be assessed.

Implementations and demos

FORGE Rasmussen, M. H., Hviid, C. A., and Karlshøj, J. (2017a). Web-based topology queries on a BIM model. *5th Linked Data in Architecture and Construction Workshop*, November 13–15, 2017. Univeristy of Burgundy, Dijon, France. doi:10.13140/RG.2.2.22298.95685

This paper presents the first software implementation of BOT. A custom exporter for the BIM authoring tool AutoDesk Revit was developed. The exporter assigns Uniform Resource Identifiers (URIs) to all elements in the model and exports BOT instances and relationships to a file. By loading the BIM model into a AutoDesk Forge based web viewer it is demonstrated how the 3D geometry can be queried and filtered based on the topological relationships.

SSN Rasmussen, M. H., Frausing, C. A., Hviid, C. A., and Karlshøj, J. (2018b). Demo: Integrating Building Information Modeling and Sensor Observations using Semantic Web. M. Lefrançois, R. García-Castro, A. Gyrard, and K. Taylor (Eds.), *Proceedings of the 9th International Semantic Sensor Networks Workshop, International Semantic Web Conference*, October 9, 2018 (pp. 48–55). CEUR Workshop Proceedings. Accessed September 2018. Monterey, CA, United States. Retrieved from http://ceur-ws.org/Vol-2213/paper4.pdf

This paper demonstrates an integration between an architectural BIM model and sensors installed in the actual building. The integration uses mapping between the LBD-dataset and the sensor observations described with the Semantic Sensor Networks Ontology (SSN)/Sensor, Observation, Sample, and Actuator Ontology (SOSA) ontology. The paper, however, describes a future vision of having the low voltage engineer describing these relationships as part of the project design. Thereby, the implementation outlines a design approach for describing sensors and actuators of a future building.

OSH[‡] Schneider, G. F., Rasmussen, M. H., Bonsma, P., Oraskari, J., and Pauwels, P. (2018). Linked Building Data for Modular Building Information Modelling of a Smart Home. *eWork and eBusiness in Architecture, Engineering and Construction (ECPPM 2018)*, September 12–15, 2018 (pp. 407–414). Copenhagen, Denmark: CRC Press

This paper presents the Open Smart Home dataset[4]*, and thereby demonstrates integration of disparate information sources in a modular BIM. The dataset is made available online with the intention of providing a basis to study a KG with smart home data.*

EXIST[‡] Bonduel, M., Rasmussen, M. H., Pauwels, P., Vergauwen, M., and Klein, R. (2018a). A novel workflow to combine BIM and Linked Data for existing buildings. *eWork and eBusiness in Architecture, Engineering and Construction (ECPPM 2018)*, September 12–15, 2018 (pp. 407–414). Copenhagen, Denmark: CRC Press

This paper demonstrates a modelling approach for existing buildings where the building topology is described using BOT without the use of traditional BIM authoring tools. Thereby the geometric and the semantic modelling phases are decoupled allowing the building topology to be described without having to model the geometry of each object. The paper presents an integration with the BIM authoring tool Revit that allows some automation for mapping BIM objects to their equivalent Linked Data entities.

[‡]Co-author

[4]https://github.com/TechnicalBuildingSystems/OpenSmartHomeData

Contents

Structure of the Thesis

Content

The thesis is divided into six chapters where the papers constitute the sixth. The first chapter introduces the practical point of departure by identifying characteristics and issues with current practices in the Architecture, Engineering and Construction (AEC) industry. In the next chapter, the theoretical point of departure starts by defining a research problem to uncover. In Chapter 3, a set of Research Questions (RQs) are defined based on the findings from the first two chapters. The methodology for investigating these questions is defined, and a set of Research Tasks (RTs) is laid out. Chapter 4 elaborates on the research tasks conducted as part of the present research and refers to the specific papers that describe the findings in detail. Finally, Chapter 5 discusses the implications and draws overall conclusions. Together, the chapters, in combination with the papers, represent the thesis. The overall structure is as follows:

Chapter 1 - Practical point of departure
This chapter describes the practical point of departure. It covers the motivational problem that initially led to taking the initiative to do research in the field and write the thesis at hand.

Chapter 2 - Theoretical point of departure
This chapter describes the theoretical point of departure. Initially, the overall research problem is formulated based on the observed problems from Chapter 1. Through a review of the state of the art literature on the topic, an overview of present research challenges is established, and thereby, this chapter sets the framing for how the research should be focused.

Chapter 3 - Research design
This chapter describes the research design. Based on the challenges identified in the previous two chapters, a set of research questions describing the research quality criteria are defined. The chapter further defines the overall research methodology and defines three research tasks to be accomplished through the research project.

Chapter 4 - Results
This chapter evaluates on the findings related to each of the three research tasks described in the research design. The evaluation represents a summary of what is described in detail in the associated papers.

Chapter 5 - Implications and conclusion
This chapter discusses and summarises the research findings. The significance of the research is clarified by revisiting the research questions and evaluate the general contributions made to the research community and the industry.

Chapter 6 - The papers
This chapter includes all the papers presented in the List of Papers. The papers are ordered based on publication date and those that were still under review at the time of submission are ordered based on submission date.

Reading guide

In Linked Data, everything is described using International Resource Identifiers (IRIs) (i.e. web addresses) that should return meaningful data when looked up in a web browser. In Section 2.5 it is described in detail why this is useful, but, initially, it is important to note that throughout the thesis such IRIs occur in a syntax where they are shortened by so-called prefixes. For example, if the prefix dbo is used to describe the namespace http://dbpedia.org/ontology/ then dbo:Person will refer to http://dbpedia.org/ontology/Person. In the Portable Document Format (PDF) version, the prefixed IRIs have hyperlinks embedded, and clicking this link will show the documentation of the particular thing. In the printed version, the full IRI is not visible, but the use of prefixes in this work is limited to the ones listed in Table 1.

Table 1: General prefixes used throughout the thesis.

Prefix	Name	Namespace
BOT	Building Topology Ontology	https://w3id.org/bot#
CDT	Custom DataTypes	http://w3id.org/lindt/custom_datatypes#
DBO	DBPedia Ontology	http://dbpedia.org/ontology/
EX	Example used for fictive ontologies	[fictive namespace]
FOAF	Friend Of A Friend ontology	http://xmlns.com/foaf/0.1/
FSO	Flow System Ontology	[fictive namespace]
GEO	GeoSPARQL	http://www.opengis.net/ont/geosparql#
ICE	Indoor Climate and Energy	[fictive namespace]
INST	Used to denote instances	[fictive namespace]
OPM	Ontology for Property Management	https://w3id.org/opm#
OWL	Web Ontology Language	http://www.w3.org/2002/07/owl#
PROV	The PROV Ontology	http://www.w3.org/ns/prov#
RDF	Resource Description Framework	http://www.w3.org/1999/02/22-rdf-syntax-ns#
RDFS	RDF Schema	http://www.w3.org/2000/01/rdf-schema#
schema	schema.org	https://schema.org/
SEAS	Smart Energy-Aware Systems	https://w3id.org/seas/
SOSA	Sensor, Observation, Sample, and Actuator Ontology	http://www.w3.org/ns/sosa/
XSD	XML Schema Definition	http://www.w3.org/2001/XMLSchema#

A full list of acronyms as well as a glossary with descriptions of subject-specific terms can be found in the end of the thesis. All items contained in the glossary are marked like this. In the electronic versions, there are links embedded in all acronyms and glossary entries that lead to the definition.

There are illustrations throughout the thesis where relationships between 'things' are shown as a directed graph. Solid arrows indicate that a relationship is explicitly stated in the dataset whereas dashed arrows indicate that the relationship is inferred. The technology behind this is explained in Section 2.5.

Chapter 1

Practical Point of Departure

This chapter describes the practical point of departure. It covers the motivational problem that initially led to taking the initiative to do research in the field and write the thesis at hand.

1.1 Introduction

When operating in the Architecture, Engineering and Construction (AEC) industry, one must understand that it is a cross-disciplinary field with several stakeholders involved. Each stakeholder, that being the client, the client's advisor, the architect, an engineer, an expert, or the contractor, each have their expectations, culture and points of view on the project, and this increases the overall complexity of the project. With each new project, a new temporary organisation is formed with stakeholders from different companies, so it is hard to transfer collaborative insights to the next project. Further, even though the different phases of a construction project are often represented as a linear process which goes from the planning phase over the design phase through the construction phase to commissioning and operation, the reality is not so. Designing a building is an iterative task which undergoes several changes along the way (Bertelsen, 2003; Kiviniemi, 2005b).

It is a common perception in the AEC industry that each new building is different from the previous, meaning that every new building is practically a prototype. In reality, though, Heating, Ventilation and Air Conditioning (HVAC) engineers, to a great extent, perform the same data processing in each new project, independently of the building's shape. Advanced geometrical shapes increase the complexity of geometrically dependent information take-offs, but the processes are similar. Space management and routing of mechanical installations, of course, require more creative solutions, but this is solved by initially establishing some guidelines like in any other project. Figure 1.1 shows a principle drawing, outlining guidelines for where the various mechanical installations should be located. It defines a general and secondary (branching) level for the different installations.

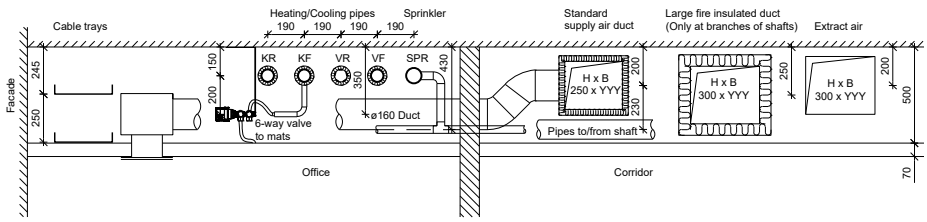


Figure 1.1: Principle drawing showing routing guidelines.

Data processing in HVAC engineering fits into a set of relatively structured systems. Each system consists of smaller sub-systems, and if there is a need for data exchange between two systems, there is an interface. Interfaces between sub-systems, and potentially between different design disciplines, constitute a potential conflict. In current workflows, most data exchanges are handled in a

predominantly manual fashion, and it is a complex task to maintain the overview and keep everything in sync. If computers could manage a greater extent of this work, the engineers would be better equipped for complying with the dynamic, iterative nature of the building design process.

1.2 A Challenged Industry

It is a known fact that productivity in the construction industry lags behind compared to other industries. A report by National Institute of Standards and Technology (NIST) (Gallaher et al., 2004) indicated an annual loss of $15.8B in the United States (US) Capital Facilities Industry as a result of poor interoperability, and in a recent report by McKinsey (2017), similar figures are presented. Figure 1.2 summarises the situation concerning productivity in the construction industry during the past two decades. Even though there have been productivity enhancements, they are minimal compared to the manufacturing industry and the total economy as a whole.

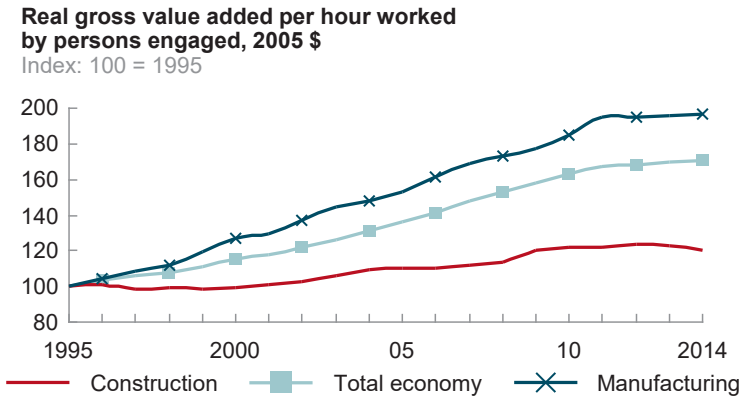


Figure 1.2: Global productivity growth trends (McKinsey, 2017).

The majority of the losses (68 %), from inadequate interoperability, identified in the NIST report, were incurred by building owners, and primarily in the operations and maintenance phase. The losses for architects and engineers, the disciplines particularly in scope with this work, have the highest losses (86 %) during the planning, engineering and design phase. These losses account for $1B of the $15.8B, which is quite significant when taking into account that these disciplines only include consultancy activities that constitute only a minimal share of the full cost figure.

McKinsey (2017) presents seven ways to improve the productivity of construction being:

1. Reshape regulation and raise transparency
2. Rewire the contractual framework
3. Rethink design and engineering processes
4. Improve procurement and supply-chain management
5. Improve on-site execution
6. Infuse digital technology, new materials, and advanced automation
7. Reskill the workforce

Roughly half of these seven items directly deal with information handling. *1.* encourages that transparency is enhanced and part of this includes mandating the use of technologies such as Building Information Modelling (BIM). *3.* deals with processes and encourages early collaboration from all parties involved in design as well as repeatability of design across projects. *6.* deals with investments in innovation offices and teams, the application of BIM and Three-Dimensional (3D) models, and the use of digital collaboration and mobility tools on portable devices.

1.3 HVAC Design as a System

This section uses a concrete construction project, a large-scale office building, to elaborate on the "systems of systems"-methodology described in Section 1.1. Here, the systems are denoted 'tasks', and the example demonstrates the overall task of 'designing a heating system for a building' along with the embedded sub-tasks. This particular design task is revisited in Chapter 4, where a software artefact for radiator sizing is described.

Designing a heating system includes several sub-tasks, of which the majority have interfaces with other sub-tasks within the overall design task or to other tasks performed within the super-task of designing the building as a whole. An interface, in this case, entails that the task either consumes information from or generates information to one or more other tasks.

Figure 1.3 shows the main sub-tasks involved in designing the heating system of the particular building. In general, each square indicates a task, manifested in a document, and the arrows between the squares indicate an interface. For example, a U-value calculation produces a resulting heat transmission coefficient of each building element. This result is necessary for performing heat loss calculations for the spaces in the building, as the building elements constitute the building's thermal envelope. The coloured squares in the figure symbolise an interface towards another super-task. For example, the heat loss calculation takes the geometrical properties such as surface areas and lengths of cold bridges as input from the architectural design. The design task of performing the architectural design belongs to the architect and hence another practitioner in the project.

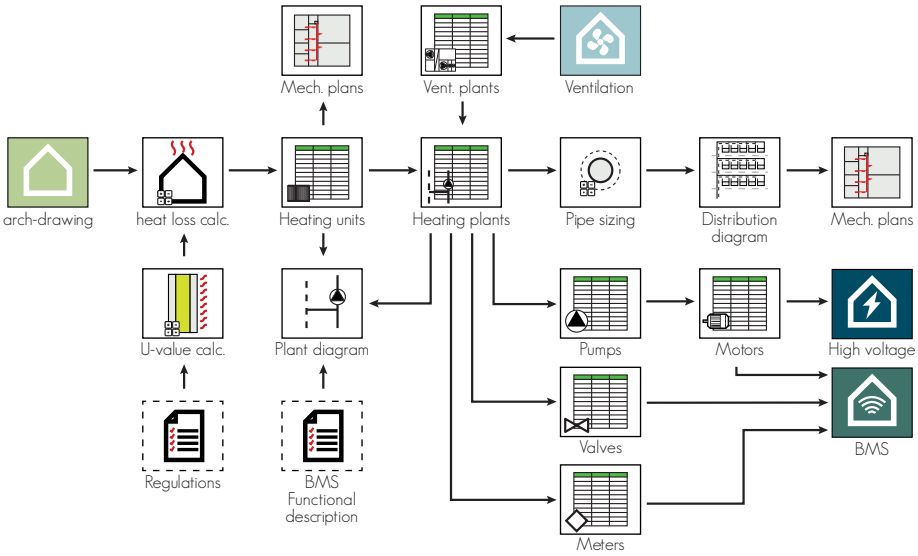


Figure 1.3: The HVAC design system.

Heat Loss Calculation

When designing a heating system, the first task is to calculate how much energy is needed to heat each space in the building. Accomplishing this requires the geometry of the building envelope segments facing each space, and the architect is the source for this information. In this particular project, the geometrical properties (i.e. heat transfer areas and lengths of cold bridges) were extracted manually from the Two-Dimensional (2D) plan drawings taking into account measuring rules defined in the Dansih Standard (DS) for calculation of buildings' heat loss (DS418, 2011). The thermal performance of the elements represented by each building envelope segment is necessary for calculating the transmission

heat loss through these segments. Heat transmission coefficients (U-values) for each separate construction type is a result of what materials they consist of, and at the early stages of a design, the architect has typically not considered this. Therefore, defining these is an iterative task which goes back and forth between the architect, the contractor, the engineers and the facade manufacturer. Initially, some assumptions were made to enable the design to make progress. Such assumptions are made based on the engineer's experiences with what is necessary to meet *(1)* the overall heating requirement of the building and *(2)* the heat transmission through the individual building elements, which are both regulated by the building regulations. All building envelope segments facing a space are summed up, and together they constitute the transmission heat loss of the space. The infiltration heat loss of a space is also accounted for, but this is typically applied as an even distribution of a general volume flow. The size of this volume flow depends on the air-tightness classification of the building which is restricted by the building regulations.

Heating Units

The heat sources supplying heat to a space must meet the heating demand of that space. In this particular project, heating is mainly provided by water-filled plastic tubes above the suspended ceilings. There is a limit to how much heat can be supplied by such a system, so some spaces need supplementary radiators. Spaces with double high glass facades further have support heating from convectors to prevent downdraft, and additional devices, such as air curtains and ramp heating for snow melting, were also required by the client. A list of heat sources was maintained, and additionally, these were shown on the 2D mechanical plan drawings. The central heating system is divided into sub-systems, and the full distribution system is illustrated in a 2D diagram. A subset of this diagram is illustrated in Figure 1.4. Most sub-systems contain a heating plant being either a heat exchanger or a shunt (mixing plant). The heating system and the plants, in particular, include controllable devices and sensors that are connected to the Building Management System (BMS), and hence the equipping of these involves a low-voltage engineer. Some Air Handling Units (AHUs) designed by the ventilation engineer include a heating coil to heat the ventilation air. Each of these coils also requires a separate mixing plant, which is part of the heating system. The equipping of plants and AHUs is illustrated in Piping & Instrumentation (PI)-diagrams like the one in Figure 1.6.

Pipe Sizing

Once the full layout of the heating system is known, it is possible to assess how much hot water is to be distributed throughout each pipe. This volume flow sets the main boundary conditions for the pipe sizing, and when further

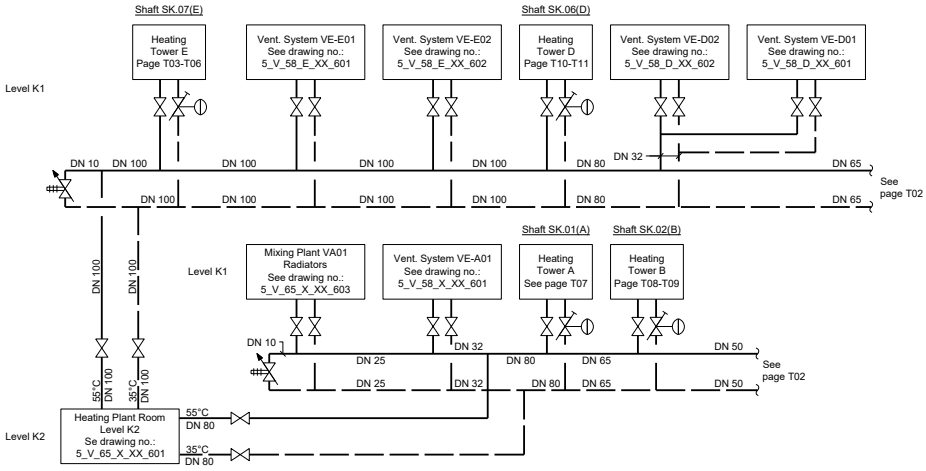


Figure 1.4: Subset of the distribution diagram.

taking into account the pipe material and its vulnerability to corrosion at high water velocities, as well a weighing between material cost, and electricity cost for circulating the water, the pipe size can be decided. In this particular project, the pipe sizing calculations resided in separate spreadsheets for each heating sub-system. The water flows and pipe sizes were manually transferred from the static documents (flow calculation, heating unit lists, heating plant lists) to the distribution diagrams, plan drawings and PI-diagrams.

Devices

Each heating plant contains a circulation pump, and the hydronic properties of these pumps, as well as pumps for circulation of domestic hot water, booster pumps for domestic water and so forth, were managed in a pump list. Valves are sized so that they have the necessary authority, and these sizes were stored in a separate list as well as indicated at the PI-diagrams. A third list contained energy- and mass flow meters. All these components are part of the BMS, and therefore the lists are shared documents that are processed by both the HVAC engineer and the low voltage engineer. Each pump contains a motor that needs a power supply. Therefore, these also existed in a fourth list belonging to the high voltage engineer with specifications relevant to the electrician.

Change Management

Designing a building is a dynamic process, and changes occur throughout all phases. Architectural design changes and functional space demand changes like

demonstrated in Figure 1.5 have consequences for the heating system design. Indoor climate demands typically follow the space function and the size of the building envelope related to one space directly impacts the heating demand. Changes must be expected, and therefore, it is typically considered bad practice to have the same information stated at more than one place. Instead, it should be attempted to refer to the lists wherever possible, and dimensions should be indicated only at the distribution diagram and not on the plans, sections and elevations. In this project, however, the contractor requested information such as dimensions and valve settings to be displayed at all drawings, which is rational from an execution perspective. This demand resulted in a situation where for example a pipe dimension could exist both in the initial calculation, on a plant drawing, on plan, section and elevation drawings as well as on the distribution diagram. Since there was no dynamic link between the pieces of information, it is not hard to imagine the extent of the workload and overview required by the engineers as changes occurred.

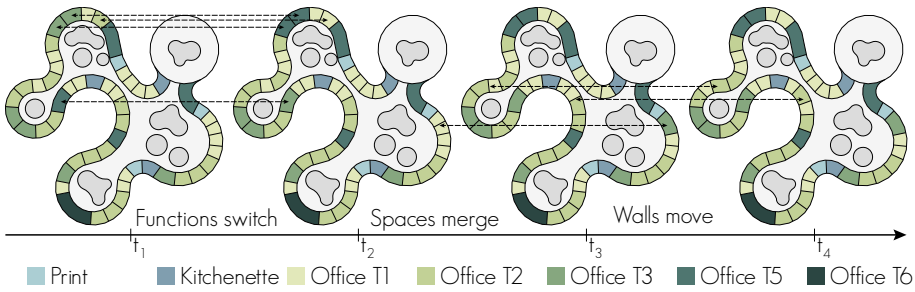


Figure 1.5: Reoccuring design changes in the reference project.

Figure 1.6 shows a PI-diagram. Even though a BIM model exists, these diagrams and the information they contain is managed separately. This image illustrates how changes are typically carried out. Most engineers do not have the software or the required skills to perform even simple changes in the diagram, so this is typically accomplished by marking the changes on a printout that is handed to a technical designer who digitises the hand drawing. This particular PI-diagram is practically a visualisation of the components that constitute an AHU, and it would be possible to auto-generate it from a data model. The concrete change indicated in the image concerns adding four sensors at particular locations in the AHU. If the drawing was generated from a data model, realising this change could be effected by adding the sensors to the data model. In a BIM workflow, this would further entail that the sensors would automatically be part of the low voltage engineer's project, and would hence show up on the respective sensor lists.

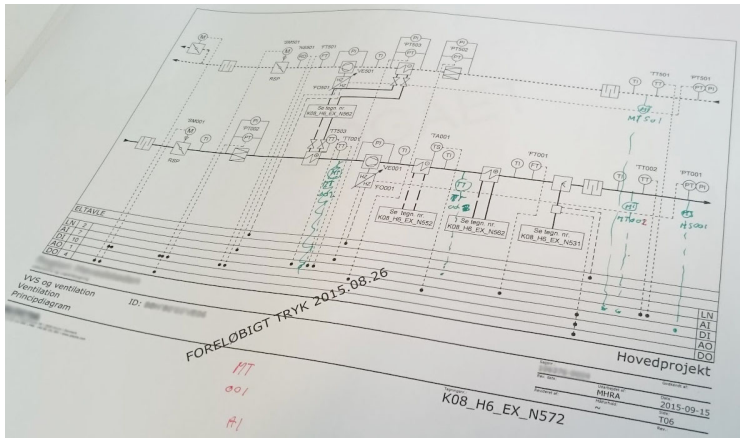


Figure 1.6: Change requests for a PI-diagram.

1.4 Observed Problems

This section summarises the problems identified in the HVAC case. Eppinger et al. (1989) describes how strategically decoupling major design tasks into subsystems can reduce the sizes of the working design groups, which can have a dramatic positive impact on development performance. Therefore, it is positive that the industry has developed methods for providing this sub-division. However, with the current working processes, there is still a challenge in the interfaces between each system. This challenge consists of the fact that most sub-tasks are performed in separate software tools with no dynamic linking between them. Even with the presence of BIM tools, data is trapped inside the native environments and is hard to access. Therefore, there is information redundancy, as it was demonstrated with the PI-diagram.

The general situation is that the documentation exists in the software in which it was also created. This is the case for calculations/lists living in spreadsheets and drawings generated in Computer Aided Design (CAD) or BIM software tools. Finding a piece of information is, therefore, a matter of either opening the software tool in which it was generated or browsing through the printouts (Portable Document Format (PDF) or paper). As there is not one single source for information, i.e. a database, it is a cumbersome task to find pieces of information on request. Further, when the right document containing the requested information is found, it requires human processing to extract the relevant pieces of information. Common questions from the contractor such as *"What is the size of this pump?"* or *"What is the total capacity demand on the district heating supply?"* are therefore not straightforward to answer. Therefore, information

browsing constitutes a significant share of an engineer's work day.

The HVAC design case along with the author's own experiences from working in the industry exposes a set of challenges in current design practices. Below, a set of characteristics is identified. Each of these characteristics consequently leads to one or more issues, that are also summarised. Together, these form the motivational problems that the research should explore in detail and investigate potential solutions for.

Main characteristics

1. Information storage is primarily document-centric, and BIM tools are mainly used for coordination and production of traditional design documentation in the form of 2D drawings and quantity lists.
2. Data exchanges are handled through the exchange of whole documents rather than specific data.
3. A substantial part of the knowledge built up is only stored in the minds of the project participants who become an indispensable information source over time.
4. It takes several years to carry out a construction project, and over this period of time there is inevitably a replacement of employees.

Main issues

1. Most data is unstructured, which makes it hard to access and reuse.
2. Exchanges in document form require human processing and make it impossible to establish dynamic links between the interrelated design tasks that the documents represent.
3. As the project evolves, changes occur. Comprehensive manual revision work must be accomplished and it becomes increasingly hard to avoid inconsistencies in the documentation.
4. Lacking transparency makes it hard to assess the consequence of a design change.
5. Only limited automation is achieved.
6. Information is lost when an employee leaves a project, and the success of a project is vulnerable to employee turnover.

1.5 Research Scope

In Section 1.2, three of the seven improvement proposals identified by McKinsey (2017) were concerned with information management. All these proposals are interrelated, and getting the full benefit from one item on the list might require that another improves as well. For example, contractual frameworks can provide an economic incentive for a proper interdisciplinary collaboration between the different stakeholders, and therefore, these are a precondition for harvesting the full potential of the technologies. The same accounts for point *7.* which is concerned with the skills of the workforce. Technically competent workers are a necessity if a more substantial part of knowledge management is to be accomplished with Information Technology (IT) tools. The software design, especially the User Experience (UX) design, is also an important factor in succeeding in the worker's adoption. All these fields are important, but not possible to solve in one PhD.

The imminent problem that the present research seeks to remedy is the poor interoperability between the software commonly used in HVAC design. As the title of the work indicates, the research investigates how to establish a "digital infrastructure" that will allow information to flow more seamlessly between the sub-systems involved in designing HVAC systems. Further, the goal is to investigate data storage solutions that will allow project information to be stored in one place, thereby avoiding redundancy that will potentially lead to inconsistencies over time.

The development of software artefacts and data models through prototyping is in scope, but this is solely for proof of concept purposes. Therefore, user interaction, UX design and associated investigations are not considered.

Chapter 2

Theoretical Point of Departure

This chapter describes the theoretical point of departure. Initially, the overall research problem is formulated based on the observed problems from Chapter 1. Through a review of the state of the art literature on the topic, an overview of present research challenges is established, and thereby, this chapter sets the framing for how the research should be focused.

2.1 Introduction

In Chapter 1, the motivation for the present research was laid out. Through experiences from a real design case, a set of characteristics and appertaining issues were identified. Based on these observations, the overall research problem below has been formulated:

> *"How to provide a web-based infrastructure for building data that allows a shared, distributed dataset to grow organically with the construction project, thereby providing insights and traceability as changes occur in a form that is consumable and extensible by any authorised agent*[1] *involved in the project?"*

According to Leiner et al. (2009), the first recorded description of online social interactions was a series of memos written by J.C.R. Licklider in 1962. Interesting enough, one of the social interactions he described, 'Computer-Aided Planning and Design', dealt with the design of a building. Using the design of a hospital as a reference case, he describes the potential of being able *"to permit several persons with various backgrounds and interests to look at tentative plans from their own differing points of view and to manipulate and transform the plans during the course of their discussion" (Licklider and Clark, 1962).* Comparing this to the current digitisation level of the construction industry, Licklider was very visionary.

The scope of this chapter is to first investigate general research in the field of digital communication in knowledge-intensive industries (Section 2.2). Building Information Modelling (BIM) was introduced to enhance digital interoperability in the Architecture, Engineering and Construction (AEC) industry, and in Section 2.3, an overview of the technologies and methodologies around BIM are given along with the current challenges that are yet to be solved. Tools exist that use cloud technologies to improve BIM interoperability and these are examined in Section 2.4. In Section 2.5, an introduction to the semantic web and linked data is given. These technologies are currently taking the World Wide Web (www) to the next level and could potentially do the same for BIM. The semantic web is used to construct a distributed, federated Knowledge Graph (KG) – a term used to denote a particular way of organising data, and in particular data relationships in the structure of a directed graph. Using formal, shared terminology it facilitates the description of real-world objects conceptually. In Section 2.6, existing efforts in using semantic web technologies in the context of the AEC industry are investigated.

[1] An agent is in this context a software that acts on the behalf of a user.

2.2 Knowledge Management in AEC

Managing knowledge is by Robertson et al. (2001) described as being particularly crucial for knowledge-intensive companies. This is also the case for companies operating in the AEC industry. Great amounts of data are accumulated, processed and evaluated by knowledge-workers employed by companies operating in this industry, and their main asset is, therefore, knowledge. The prominent challenge is that this knowledge is mainly materialised in the employees of the companies. As noted by Tserng and Lin (2004), *"most experience exists only in the minds of the individual participants"* and this means that critical knowledge is lost in periods with high employee turnover (O'Leary, 1998).

Definitions

To understand the challenges in Knowledge Management (KM), one must first understand the difference between data, information, and knowledge. The Data, Information, Knowledge, Wisdom (DIKW) hierarchy (Figure 2.1) (Ackoff, 1989) describes that data is only the raw facts. It does not require prior understanding and has no context assigned – A sensor observation is an example of data. When organising data and adding context, one gets information – The evaluated maximum temperature in a space over a period of time is information. When providing meaning with the information, the result is knowledge – The reasoning that the temperature was high at that particular time because the space was fully occupied and it was a warm summer day is knowledge. Wisdom is then considered as an evaluated understanding of that knowledge. With wisdom, it is possible to make well-founded predictions about the future.

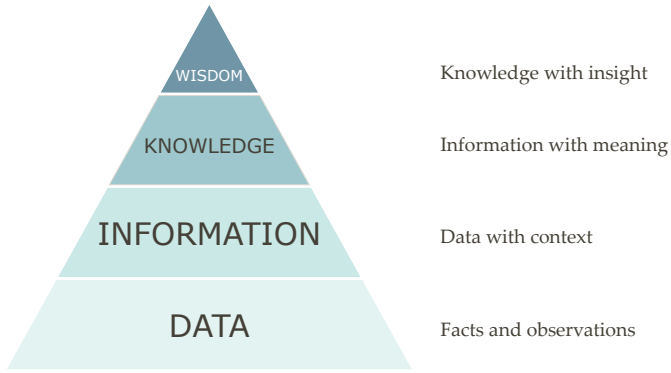


Figure 2.1: The DIKW hierachy (Ackoff, 1989).

The definition of KM is somewhat divergent. Girard and Girard (2015) conducted an extensive investigation across various disciplines and in conclusion defined it as either *"the process of creating, sharing, using and managing the knowledge and information of an organisation"* or *"the management process of creating, sharing and using organisational information and knowledge"*.

Project Websites

An extensive amount of data is accumulated as the design of a construction project progresses. The dilemma is that this data is mainly unstructured in formats such as building element descriptions in prose text, drawings, minutes of meetings and correspondences, in addition to what is solely stored in the memory of the project participants (Deshpande et al., 2014; Kiviniemi, 2005b).

The first step in elevating the documents from simple data to enriched knowledge is to add structure and make the documents available to other project participants. Through the last couple of decades, the industry has gradually made a shift towards online-based collaboration. The first technological advancements, after e-mails, were project websites. A project website is a document sharing system with functionality for adding metadata about each document, and thereby it provides a central platform which is commonly accessible for all project participants (O'Brien, 2000).

The shift from exchanging paper-based documents to sharing digital files online might not seem like a notable change, but O'Brien (2000) found significant adaption challenges in this transition. This observation supports the opinionated view of the construction industry being conservative and resistant to change. The stereotypes that form this view, are however not all true. For example, K. A. Davis and Songer (2009) found that there is no correlation between age and education and resistance to Information Technology (IT). General computer understanding, experience from past IT change and perceived future IT change are however factors that make a difference. Ease of use is a key driver for the acceptance of new technology, and Venkatesh (2000) supports the importance of an individual's general beliefs regarding computers in this regard. Fortunately, as stated by K. A. Davis and Songer (2009), engineers and architects have a high level of IT use. Therefore, the stereotype of the conservative industry might not be entirely correct after all.

Project websites add structure to documents and manage versioning. Knowing that a document has changed is useful, but if the document is a 50 pages report, the change can mean many different things. Documents are at the lowest level of the DIKW hierarchy and this means that *(1)* it requires a considerable

amount of human labour to interpret and extract the information and *(2)* the knowledge is hard to reuse in future projects. Therefore, there is a high demand for establishing a system that captures company-wide knowledge and makes it available (O'Leary, 1998).

2.3 Building Information Modelling

Section 2.2 defined what DIKW is, but the early systems for AEC KM: the project websites, only manage documents. BIM promises to add context to that data, and thereby bring it one step up in the DIKW hierarchy.

There exist several definitions of BIM, and generally, the focus varies depending on the professional background of the person defining it. The International Organization for Standardization (ISO) standard for BIM Information Delivery Manuals (IDMs) (ISO29481-2, 2012) describes that *"Building information modelling provides a concept for describing and displaying information required in the design, construction, and operation of constructed facilities"*. Another widely adopted definition of BIM is given by Chuck Eastman et al. (2008) in the BIM Handbook that defines it as *"a modelling technology and associated set of processes to produce, communicate, and analyze building models"*. The handbook further characterises Building Models as *(A)* consisting of building components represented by objects that carry graphics and data attributes as well as parametric rules to manipulate these, *(B)* including components describing how the objects behave, as needed for analyses and work processes, *(C)* having consistent and non-redundant data entailing that changes to component data are propagated throughout and *(D)* having coordinated data throughout all views. Thereby, BIM constitutes a set of technologies and methodologies that are thought to provide the AEC industry with the means to better manage the knowledge embedded in each project through its full life cycle.

History

Practitioners in the AEC industry have used computers to accomplish design tasks before the emergence of BIM. Computer Aided Design (CAD) systems, that date back to Sketchpad by Sutherland (1964), were primarily used for the production of Two-Dimensional (2D) construction drawings and the main interoperability these tools provided was the ability to do overlay drafting (Björk, 1989). The usage of computers for performing structural analyses using the Finite Element Method (FEM) has been done since 1956 (M. J. Turner et al.,

1956) and Building Performance Simulations (BPSs) have been used since 1963 (Brown, 1990). All these tasks were, however, performed in computer programs that did not communicate with the surrounding world, and interoperability was not addressed at this point.

Santos et al. (2017) suggest that the first mentioning of the term 'Building Information Model' was by van Nederveen and Tolman (1992), but the research in building models following the characteristics identified by Chuck Eastman et al. (2008) dates back to the Building Description System (BDS) by Charles Eastman (1975). At this time, however, there existed no universal standards for describing BIM models. With the establishment of the International Alliance for Interoperability (IAI) in 1996, the intention was to facilitate this interoperability by enabling full information exchange between the various software programs used in the industry. IAI, which in 2008 changed its name to buildingSMART, came out of a private alliance between 12 companies in the industry started by AutoDesk already the year before (buildingSMART, 2018).

One of the main contributions by IAI is the Industry Foundation Classes (IFC) standard by Liebich and Wix (1999), which is Today an ISO standard (ISO16739, 2013). The IFC Object Model was initially inspired by other efforts from the STandard for the Exchange of Product model data (STEP) standardisation project by ISO Technical Committee (TC) 184/Subcommittee (SC) 4 (ISO TC 184/SC 4). These efforts include the General AEC Reference Model (GARM) (Gielingh, 1988), the RATAS project (Björk, 1989), the building systems model (J. Turner et al., 1990) and the COmputer Models for the Building INdustry in Europe (COMBINE) project (Augenbroe, 1994). The IFC data model exists in different versions where the most recent at the time of writing is IFC4 Add2[2]. Each data model is expressed as a schema in EXPRESS, which is a data modeling language for product data formalised in the ISO Standard for STEP (ISO10303-11, 2004).

BIM Authoring Tools

BIM authoring tools are software applications that support object-based parametric modelling, meaning that geometric definitions and associated parameters and rules define the objects (Chuck Eastman et al., 2008). Not only the objects themselves but also the relationships between objects can be defined by rules that for example constrain the distance between two parallel walls or a pipe and a wall. Each object can be assigned with properties that fall into two categories: *(1)* geometrically derived properties such as height, area and volume

[2]http://www.buildingsmart-tech.org/ifc/IFC4/Add2/html/

and *(2)* attributes such as air flow, colour and classification code. The purpose of classification is described later in this section.

From the geometry model, it is possible to create different 2D views such as plans, sections and elevations. These 2D views can be annotated with information which is directly retrieved from the objects. This way, it is, for example, possible to indicate the airflow on a ventilation terminal or the elevation of a duct with concise information from the integrated data model. The use of an integrated data model further allows multiple views to show the same information consistently. All the air terminals in a project can, for example, be displayed in a list view that shows their id code and airflow. Changing an airflow in the list is immediately reflected on any annotated 2D view showing that information, and thereby, information in the model is defined non-redundantly in a manner that allows multiple visualisation possibilities. Geometry is also integrated non-redundantly, so if the location of a wall changes, all plan views showing this plan and any dimension annotations related to the geometry of the wall are updated consistently. All geometries associated with that wall are also modified accordingly. The change, therefore, influences any object hosted by the wall (e.g. windows, doors, radiators), or attached to the wall (e.g. other walls, roof, floor). No information or geometry can be represented redundantly internally in a BIM authoring tool, and therefore, there are no information inconsistencies.

There exist several BIM authoring tools on the market. Revit, ArchiCAD, Vectorworks, DDS-CAD and Tekla Structures are some of the most widely used. Some of them (namely Revit and Vectorworks) cover both architecture and engineering where DDS-CAD and Tekla Structures are specialised tools for Mechanical, Electrical and Plumbing (MEP) and structural engineering respectively.

BIM and Classification

Classification systems facilitate a more detailed specification of object types than the one used internally in the BIM authoring tool. For example, the Revit software uses a generic class 'Mechanical Equipment' for most objects in this category. It is not possible to specify that it is a radiator, but with a classification system such as the Danish Cuneco Classification System (CCS)[3] this can be achieved by assigning a classification code – in this case 'EPE'. CCS uses a rather coarse classification system and assigns additional properties to each class for further specification. Other systems, such as OmniClass[4], define more specific classes such as '21-03 10 20 20' which is the class for 'Interior Fixed Windows'. In CCS, such an object would be described as a 'QQA' with two boolean properties: 'interior' and 'openable' assigned.

[3]Cuneco Classification System: https://ccs.molio.dk/?sc_lang=en-gb
[4]Omniclass: http://www.omniclass.org/

BIM Data Exchange
Chuck Eastman et al. (2008) describes that objects can *"link to or receive, broadcast, or export sets of attributes"*. This mechanism allows reuse of object attributes in other software tools for specialised tasks such as simulation. ISO TC 184/SC 4 distinguishes between data exchange and data sharing. Data exchange is achieved by exchanging information between two software systems through flat files that represent the state of information at a single point in time (Isikdag et al., 2007). STEP Part 21 formatted IFC files are commonly used to exchange BIM data, although IFC also exists in other serialisation formats. Data sharing is an alternative to data exchange that has the benefit of using one single data source that all applications have access to. This technology is covered in Section 2.4.

With Model View Definitions (MVDs) it is possible to define a particular subset of the full IFC schema, a Model View, that is needed for a given task. Thereby a MVD can be applied by a BIM authoring tool to only write a subset of the model to a file. This is useful since different tasks such as BPS, cost scheduling or static analysis require different sub-models and specific object attributes. In IFC, properties are further grouped into property sets for different purposes. This simplifies the process of defining a MVD for a particular task.

Besides from IFC, there are also other Extensible Markup Language (XML) based formats for specific data exchange purposes such as energy analysis (green building XML (gbXML)) and Geographic Information System (GIS) (OpenGIS). These are developed by other standardisation organisations than buildingSMART but have the same purpose of allowing interoperability.

Specialised BIM Tools

There exist software tools that, at various levels, implement the BIM methodologies by making use of data from the building model. As a minimum, these tools support the import of IFC. As noted by Chuck Eastman et al. (2008), most applications do not need to create or edit geometry, and all the BIM tools listed here are merely consuming fixed geometry along with attributes from BIM authoring tools. In the following, a few examples of tools are given that to some extent support an integrated BIM workflow.

Performance Simulation
This category comprises tools that simulate different aspects of the design. Some BIM authoring tools have basic BPS capabilities like simple thermal simulation and shadow assessment built in, and dedicated tools like IDA ICE and IES Virtual Environment support the import of geometry through IFC or gbXML. Since only the geometry is exchanged, and since it often requires special modelling precautions (Maile et al., 2013), the tools do not provide a true BIM workflow. Structural analyses in an integrated BIM workflow using FEM software is also possible with tools like AutoDesk Robot and Strusoft FEM-Design.

Clash Detection and Model Validation
Tools in this category provide the user with the ability to load several models into the same environment in order to perform geometry clash detections and model validation through rules. In a case study of a hotel project, Azhar (2011) documented a significant saving from avoided clashes between mechanical installations and the building's structural systems due to preliminary clash detection assessment. Clash detection is, therefore, a widely used feature enabled by BIM. Examples of tools that facilitate clash detection include AutoDesk Navisworks and Solibri Model Checker. Solibri Model Checker, alongside with Datacubist SimpleBIM, provide a set of specialised tools for model validation. SimpleBIM has filtering and merging functionality to create new models for a given purpose, and Solibri Model Checker has functionality for rule compliance checking.

Cost and Time Planning
The last category included in this overview covers tools that are heavily used by contractors and Construction Managements (CMs) since they facilitate simulation of the execution process. These simulations are often denoted by the additional dimensions they add on top of the Three-Dimensional (3D) geometrical representation of the building. The fourth dimension (4D) is achieved by assigning a start time and a duration of the construction work involved in constructing each object. This added information is visualised in location-based schedules and animation videos, that both provide insights for optimising the construction work sequences. 4D simulations are used to optimise construction trades and identify clashes between trades that are supposed to conduct work at the same location simultaneously. The fifth dimension (5D) is achieved by further adding embedded construction costs to the objects. With this information added, it is possible to provide better cost estimates for tendering and making detailed analyses of cash flow during construction. Vico Office, Sigma Estimates, RIB iTwo, Synchro and Navisworks all provide some of these functionalities in a BIM-based workflow.

Adoption and Maturity

In several countries, providing incentives for companies operating in the construction industry to adopt BIM has been promoted at a political level (Smith, 2014). For example, with Directive (2014), it is by the European Union (EU) encouraged that each of the 28 member states may require the use of BIM and similar tools for public works contracts and design contests by the year 2016. In some countries; for example the Scandinavian countries, the governments have mandated the use of BIM even earlier (Wong et al., 2009). In Denmark, 'Det Digitale Byggeri' (The digital construction) made BIM mandatory on public projects as early as in 2007.

Even though BIM is gaining ground in the industry, the maturity of the methodology and tools is still in the early stages. Succar (2009) distinguishes between object-based modelling (BIM Stage 1), model-based collaboration (BIM Stage 2) and network-based integration (BIM Stage 3). Concerning technology, Stage 3 is characterised by semantically rich, integrated models that can be created and maintained collaboratively over a distributed network of servers. The models are described with non-proprietary formats. Also the 'wedge' diagram of BSI Standards Limited (2013) (Figure 2.2), which forms an important part of Publicly Available Specification (PAS) 1192 (BSI, 2013), describes BIM maturity levels. The diagram's 'Level 3' aligns well with Succar's 'Stage 3'. What is today referred to as BIM, is level 2 in the wedge diagram. This level was by the United Kingdom (UK) mandated on governmental projects with the EU Directive (2014). Dainty et al. (2017) convey a critical discourse towards this strategy with the concern that it could have negative effects for small construction companies who do not possess the necessary qualifications. Even for larger companies, there is a concern that many workers will not possess the necessary skills. They find that the construction industry as a whole is located somewhere between level 1 and 2. However, the study reveals that most, especially larger, companies see potential in the use of BIM.

Organisational Challenges

The success of a proper BIM implementation on a project not only depends on the maturity of the technology but is also highly dependent on the organisation around the project and the contract and delivery methods applied.

The most common project delivery method throughout the 20th century is the Design-Bid-Build (DBB) where the general contractor is engaged in the project through a tender process based on the final project material created by the ar-

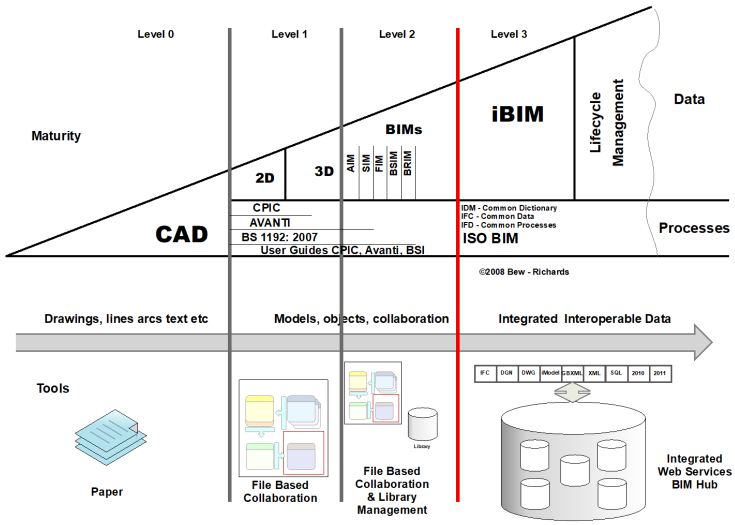


Figure 2.2: BIM Levels of Maturity (copyrighted image: Bew and Richards, 2008)

chitect or engineer (Chuck Eastman et al., 2008; Kent and Becerik-Gerber, 2010; Y. Liu et al., 2017). In a BIM environment there are some distinct challenges with this delivery method since all parties are not involved from the beginning of the project.

A more recent method, the Design-Bid (DB), which has initiated by the Design-Build Institute of America (DBIA) in the 1990s assembles the responsibility of both the design and the construction phase into one contract. Thereby, the contractors are brought to the table from the beginning of the project. An empirical study by Konchar and Sanvido (1998) found that this approach has improved the cost, schedule and quality of building projects. According to DBIA (2018), DB is gaining more popularity – especially in larger public projects. With this delivery method, the contractor can lay out prerequisites for how the designers are collaborating through BIM.

Relational Project Delivery Arrangements (RPDAs) also emerged in the 1990s. Lahdenperä (2012) compares three of these approaches: Project Alliancing (PA), Project Partnering (PP), and Integrated Project Delivery (IPD) that all incorporate *"Early involvement of key parties, transparent financials, shared risk and reward, joint decision-making, and a collaborative multi-party agreement"* (Lahdenperä, 2012). These define a form of cooperation and beside from PA, they cannot be considered as delivery methods like DB and DBB because they do

not provide the necessary contractual structure. That could, however, happen in the future. Especially with the arise of dedicated insurance products for joint liability projects like the British Integrated Project Insurance (IPI) described by Collinge and Connaughton (2017). The study by Kent and Becerik-Gerber (2010) also identifies that IPD can be applied with DB as the contractual agreement. Of the three RPDAs, only IPD specifically mentions the use of BIM, but this could be because the other two were developed before BIM. All three offer the necessary framework for an organisation to obtain the full collaborative potential of BIM.

Figure 2.3 by Chuck Eastman et al. (2008) is a conceptual diagram that illustrates the ideal information flow in a BIM based workflow supported by an RPDA. The diagram illustrates a traditional DBB, paper-based process (Ⓐ) in comparison with an ideal situation where the BIM model contains all project information (Ⓒ). Through the feasibility, design and construction stages, a continuous information buildup is achieved as a result of an improved integrated design process. In a traditional DBB contract the designers do not have an incentive to deliver more than what is demanded by the contract. Therefore, it is common practice to either only deliver the drawing material, lists and so forth, and not the source BIM model, or to strip the model for information in order not to incur liability.

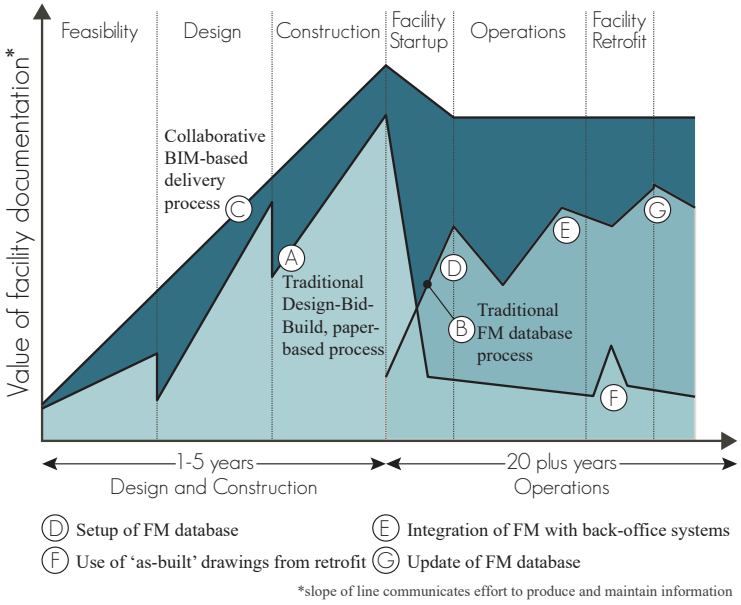


Figure 2.3: Less information is lost at project handovers with a BIM workflow (Chuck Eastman et al., 2008).

Figure 2.3 also shows the stages after construction completion. (B) shows a traditional process where the client starts from almost nothing (i.e. drawing printouts, building descriptions and so forth) and starts growing an Facility Management (FM) database. In an ideal collaborative BIM-based delivery process ((C)), the information is already available in the model.

Technological Challenges

Even though the IFC schema contains terminology to describe domain knowledge, the situation is that the model files are used for exchange and documentation only, and the potential of each domain extending the dataset is currently not practised if even possible. The storage and retrieval of domain-specific information is in practice based on proprietary vendor-specific solutions (Zhang et al., 2018).

As Pauwels et al. (2018b) points out, IFC has a strong focus on geometry while the demands for knowledge capture are not adequately met. Venugopal et al. (2012) on the other hand points out some of the shortcomings of the geometrical definitions. Based on findings by Sacks et al. (2010) they argue that they are highly redundant with multiple ways to define objects, relations and attributes which leads to unreliable, non-robust data exchanges. Therefore there is a need for embedding semantic meaning in exchange data.

In current implementations, IFC functions as a file-based format for exchanging a model from one BIM authoring tool to another. Internally, however, the tools operate on proprietary data structures and the IFC version of the model is therefore in practice read-only (Ma and Sacks, 2016; Fuchs and Scherer, 2017). The conversion roundtrip — exporting a native model to IFC and importing that back to a native format is also generally unsuccessful (Törmä, 2013). This means that full interoperability between BIM authoring tools is not yet achieved. In an early study by Zamanian and Pittman (1999) it is argued that because the AEC industry is comprised of multiple, disjointed, yet interdependent disciplines, it is desirable to have distributed project databases. An IFC file is a document and not a database and using it in a distributed manner is not immidiately possible. As described earlier in this section, there are BIM tools that allow multiple files to be viewed together and Datacubist SimpleBIM can merge IFC files. The interoperability is, however, still limited because of the file-based data exchange. Even though it is possible to view the geometrical models in a shared coordinate-system, no instance-interoperability is achievable. Therefore, it is, for example, not possible for a structural engineer to describe static-related properties of a wall defined in an architectural model (Törmä, 2013).

Zamanian and Pittman (1999) further points out that a document-based collaboration is not suitable for concurrent design and engineering because it generally focuses on 'static' data which does not allow for dealing with complex and evolving formations. This remark confirms several of the identified challenges from Chapter 1 regarding interdependencies.

Another technological challenge noted by Zhang et al. (2018) is the fact that there exists no standardised query language to retrieve the data contained in a BIM model. Proprietary tools like Solibri Model Checker provide functions for querying the data in an IFC file, but this is achieved using a closed system architecture, and the functionality is limited. BimQL by Mazairac and Beetz (2013) which is implemented in the BIMserver by Beetz et al. (2010) is an open-source alternative, but it is still not a standard or adopted by buildingSMART, the functionality is limited, and further development of it has stopped.

Since the EXPRESS language is not widely used outside the engineering domains there are not many general purpose tools that support it (Zhang et al., 2018). Everything needs to be developed from inside the industry, and therefore the profit from the general technology leaps in Information and Communications Technology (ICT) are not obtained.

2.4 Cloud Computing and BIM

Cloud technologies address the interoperability problem with file-based data exchange. Data sharing entails some significant benefits over data exchange since it uses one single source of information across all software applications. This development is mainly happening in closed proprietary environments, but there are also open source solutions available. This section first provides an overview of the history of the cloud and the technologies around it. Then it provides an overview of research efforts concerning BIM data sharing.

History and definitions

The 'cloud' is one of the buzz-words of the early 21st century. It was introduced in 2004 (Vouk, 2008) and is by National Institute of Standards and Technology (NIST) Mell and Grance (2011) defined as *"a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort or service*

provider interaction." Basically, the cloud is a computer that is rentable as a service, and the extent of the service enclosed with that computer depends on the service model. Infrastructure as a Service (IaaS) is the simplest model, where only the hardware is provided, and the user is free to deploy anything (e.g. Amazon EC2), Software as a Service (SaaS) allows users to use a software (e.g. Google Docs), and Platform as a Service (SaaS) is the in-between solution that allows users to deploy customised applications to a platform (e.g. Microsoft Azure).

RESTful Web-services

Interaction with the service offered with the cloud solution typically happens through a Representational State Transfer (REST) Application Programming Interface (API) (Fielding, 2000, Ch. 5). By conforming to the REST architecture, the cloud becomes a RESTful web service. Communication with a RESTful web service is handled through the HyperText Transfer Protocol (HTTP) which is the underlying protocol used by the www. Among other things, it defines different types of requests that browsers and servers can interpret and answer. Different HTTP methods can be handled by the REST API, and thereby it is possible for client applications to request the cloud computer to perform different tasks. Figure 2.4 illustrates how a RESTful web service handles Create, Read, Update and Delete (CRUD) HTTP requests.

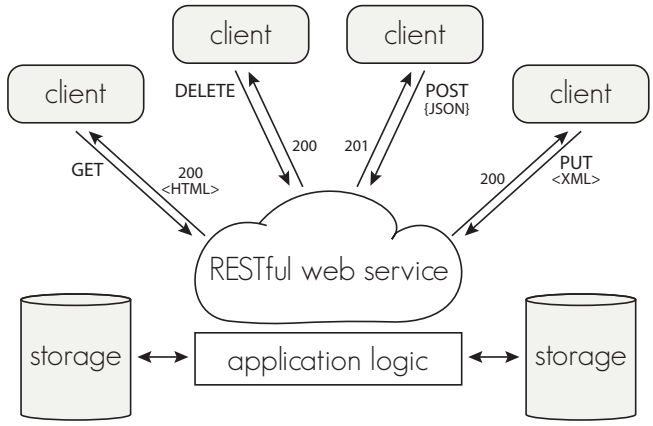


Figure 2.4: RESTful web service in the cloud.

Reading a resource stored on a RESTful web service is achieved by sending a GET request to the Uniform Resource Locator (URL) of that resource. With HTTP, it is possible to define a header stating explicitly, that the returned pay-

load should be in HyperText Markup Language (HTML) format. Other formats such as JavaScript Object Notation (JSON) could also have been requested, and the application logic of the web service takes care of this through a mechanism called content negotiation. The logic layer further handles communication with the databases that compose the storage. A successful request returns a 200 OK status code[5] in the header of the response message. Status codes provide a standardised way of communicating whether a request was successful and in cases where it is not, it provides an unambiguous reason such as for example 404 not found, 500 internal server error or 401 unauthorised. Creating data on the server requires that the client communicates what data should be created, and therefore, HTTP allows a payload to be sent with a POST request to a resource. Updating and deleting resources is handled in a similar way.

The application logic can also handle more complex tasks, and it can request data from other RESTful APIs or data processing services such as Natural Language Processing (NLP) and Business Intelligence (BI) tools. This is particularly beneficial when performing resource-intensive calculation tasks that can scale horizontally (i.e. add more computers instead of a bigger computer) for faster execution.

BIM in the cloud

The first effort in moving from file exchanges to data sharing in BIM was according to Kiviniemi (2005a) the IFC Model Server (IMSvr). After this followed the Simple Access to the Building Lifecycle Exchange (SABLE) initiative, which was developed in 2003-2005. This project aimed at providing a standardised API that would also allow federated storage of BIM on IFC model servers. Figure 2.5 illustrates the overall architecture. It uses a combination of a file storage system for physical files and a Database Management System (DBMS). A web service interface communicates with these two components through an API for IFC data processing and a database interface. Client applications can then use this web service to access the BIM data. The project was, unfortunately, discontinued and the website is no longer available.

Some more recent commercial and open source projects that qualify as cloud-BIM tools were reviewed by Chong et al. (2014), namely AutoDesk BIM360, Cadd Force, BIM9, BIMServer, BIMx and Onuma System. Ma and Sacks (2016) further adds Trimble Connect and GRAPHISOFT BIMcloud to that list. These systems have varying functionalities such as clash detection, visualisation in the browser, ability to develop applications on top and so forth. Only BIMServer

[5]Summary provided at https://httpstatuses.com/

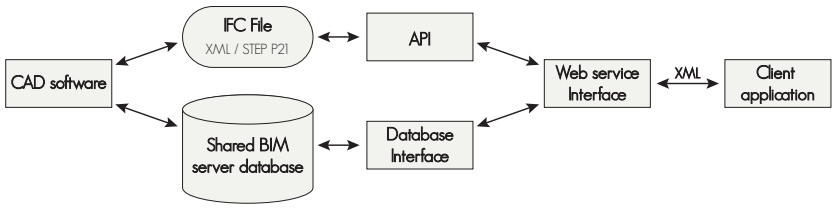


Figure 2.5: SABLE architecture is illustrated by Isikdag et al. (2007).

and Onuma System supports IFC, and therefore, the others only qualify as BIM tools because they are able to exchange information with a BIM authoring tool through its native API. The exchange, however, is mostly unidirectional and Ma and Sacks (2016) highlights that these applications mostly do not allow users to supplement the data. Also, since IFC is designed for file exchange, the internal data structure in the cloud solutions are proprietary. Typically, these store the data in relational databases following an internal schema. BIMServer, for example, uses a key-value-store database[6]. The fact that all the different cloud solutions parse the data to internal data structures means that interoperability is poor, and for users, this induces a significant dependency on the different software vendors. Ma and Sacks (2016), however, demonstrate a solution that replicates the IFC schema in the internal data structure. This is possible since they (like BIMServer) use a Not Only SQL (NoSQL) database. Such a database is fundamentally different from relational databases that use a Structured Query Language (SQL) for defining and manipulating data. Since SQL requires data schemas to determine the structure of the data before it is stored, the schema essentially needs to be known from the start. NoSQL, on the other hand, is designed to work with unstructured data and hence uses a dynamic schema. NoSQL databases depend on an Open World Assumption (OWA) which allows anyone to make statements about any resource (Klyne and Carroll, 2004), and this is valuable when describing a dataset that will evolve. Therefore, the database schema never needs to be definitive.

PAS 1192-2 (BSI, 2013) has a specification for a common BIM platform for project data – a 2.0 version of the project websites described in Section 2.2. This platform is denoted a Common Data Environment (CDE). The CDE has the concept of private and shared areas. Data in shared areas can be accessed and used by the other parties working on the project. This mechanism establishes clear ownership of data.

[6] http://bimserver.org/documentation/faq/

Concerns

Adopting any model of cloud computing entails that data will be stored on the service provider's server. Thereby, security management of the IT systems is implicitly also outsourced. This is also the case with project websites, and therefore the situation is not new. There are still, however, precautions to be taken. Clients that are particularly vulnerable to industry espionage might not be interested in having the material of their future facility stored in the cloud or might have restrictions to where the server must geographically be located.

With current BIM server implementations it is also a concern that most of them communicate with BIM authoring tools through their native API. Figure 2.6 by SABLE illustrates the problem with native APIs very well, and in the long run it is probably not a viable solution. Also, having data stored in native environments puts an increased reliance on the service provider since it is complicated to change to another provider. Kiviniemi (2005a) argues that data exchange is not feasible for AEC projects since content and structure of the domain-specific models are different. Therefore, it is not immediately clear how the SABLE project would ever have achieved standardised APIs. The next section looks into an alternative data modelling approach that might provide an answer to that question.

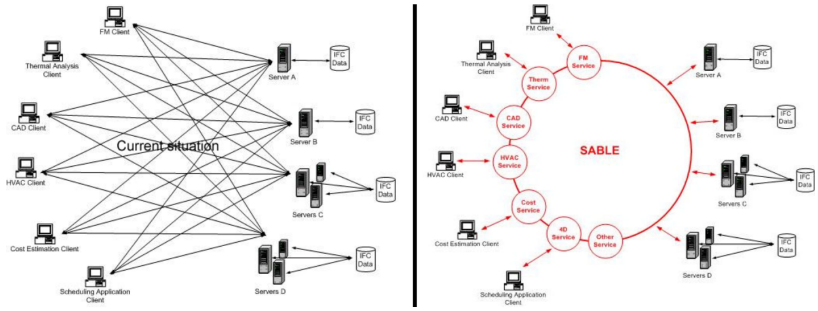


Figure 2.6: Standardised APIs in model server environment. Image from Kiviniemi (2005a) but originally by SABLE project 52.

2.5 Semantic Web and Linked Data

According to Santos et al. (2017), semantic web technologies and Ontologies are among the more recent research topics that are gaining momentum in the field of BIM. With these technologies, it is possible to add machine-understandable meaning to information. Thereby, it brings us one step further up in the DIKW hierachy.

This section examines the vision and technologies behind the Semantic Web and Linked Data. It is not the intention to provide a full guide in using the technology, but to provide the necessary foundation for understanding the research at hand.

History

Berners-Lee et al. (2001) is the first paper mentioning the vision of a semantic web. This vision consisted of using the existing web infrastructure to support KGs to be used by intelligent agents. Realising this vision should be accomplished by providing computers with access to structured collections of information and sets of inference rules for conducting automated reasoning. At this time, inference rules and systems for automated reasoning already existed in Artificial Intelligence (AI) research. Knowledge-Based Systems (KBSs) or Expert Systems (ESs), as they are also denominated, are described as a computer system that emulates the decision-making ability of a human expert (Jackson, 1998), and these are what inspired the semantic web. KBSs date back to the early 1960s (Engelbart, 1962; Licklider, 1965). However, as the situation was with hypertext before the web, KBSs were *(1)* not widely adopted and *(2)* dependent on proprietary data structures.

> *"Knowledge representation, as this technology is often called, is currently in a state comparable to that of hypertext before the advent of the Web: it is clearly a good idea, and some very nice demonstrations exist, but it has not yet changed the world." (Berners-Lee et al., 2001)*

With the semantic web, the vision was to provide a common language capable of expressing both data and rules for reasoning about the data. The Resource Description Framework (RDF) (Lassila and Swick, 1999) had already been developed, and this composes one of the main technologies for describing the KGs

that enable the semantic web. Furthermore, it was suggested to capture information in ontologies, that by Berners-Lee et al. (2001) are described as *"a document or file that formally defines the relations among terms."*. A more commonly used definition for an ontology is by Studer et al. (1998) who defines it as *"a formal, explicit specification of a shared conceptualization"*. By 'Formal', the authors refer to the fact that it must be machine-readable. 'Explicit' entails that the concepts used, and the constraints on their use, are explicitly defined, and 'Shared' implies that it describes consensual knowledge which is accepted by a group. An ontology should have taxonomy (classes of objects and relationships between them) and inference rules. Standard taxonomy for describing such ontologies was in 2002 released as World Wide Web Consortium (W3C) recommendations: The RDF Schema (RDFS) and the Web Ontology Language (OWL) (McGuinness, Van Harmelen, et al., 2004; Brickley et al., 2004).

Technology Stack

Where the vision of the semantic web encapsulates the overall vision of a web, discoverable by autonomous agents, *Linked Data*, or *Web of Data*, as it is also sometimes denoted, refers to a set of best practices for publishing and connecting structured data on the Web (Bizer et al., 2009). The full technology stack for the semantic web and linked data respectively are visualised in Figure 2.7.

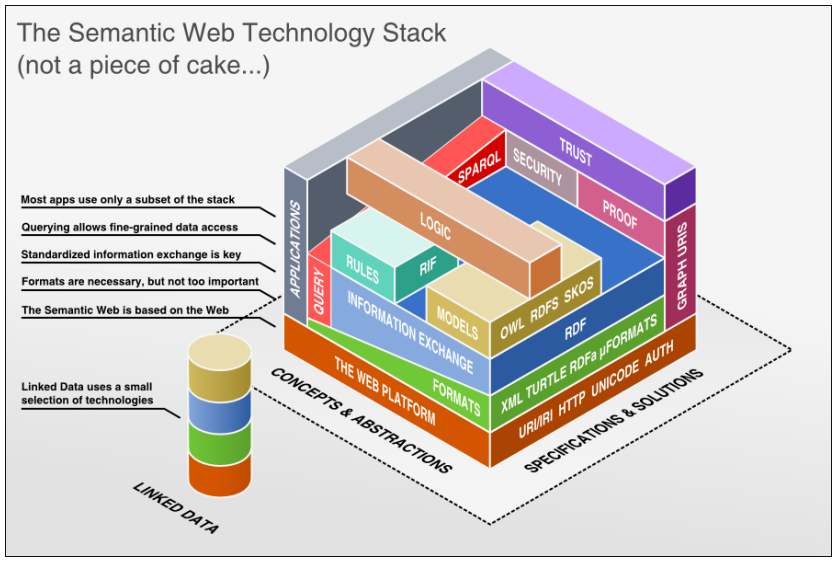


Figure 2.7: The semantic web technology stack[7].

Berners-Lee (2006) further describes the basic rules for Linked Data:

1. Use Uniform Resource Identifiers (URIs) as names for things
2. Use HTTP URIs so that people can look up those names
3. When someone looks up a URI, provide useful information, using the standards (RDF, SPARQL)
4. Include links to other URIs so that they can discover more things.

The SPARQL Protocol and RDF Query Language (SPARQL) is used to query an RDF KG, and it is described in detail later in this section.

The foundation of the stack is the web platform, which includes all the technologies on which the www is based.

URI and International Resource Identifier (IRI) are both unique identifiers for web resources – The only difference is that IRIs support international letters. URLs are a specific kind of IRI that identify document locations on the web. Since RDF 1.1, IRI is the preferred term (Cyganiak et al., 2014).

The SPARQL 1.1 Protocol (Ogbuji, 2013) is built on top of HTTP (Section 2.4) and defines specific requirements for communicating with a SPARQL Protocol service. Such a service listens for requests on a specific IRI called a SPARQL endpoint and answers the received queries. The endpoint provides an interface similar to the one offered by a RESTful web service (Section 2.4), but since the data is modelled using a standardised structure, the client can formulate more specialised requests than the provider of the RESTful web service could ever imagine there would be a need for. It does, however, come at a cost since the unfamiliar developer would not know how to formulate a SPARQL query. For simple CRUD operations, Schröder et al. (2018) therefore suggest a middleware that converts a SPARQL endpoint to a simple to use REST API. Taelman et al. (2018) further presents an approach to interact with an RDF KG through the popular GraphQL query language which was introduced by Facebook as an open standard in 2015 (Facebook, 2018). GraphQL has no notions of semantics or global identifiers and therefore, the authors suggest to extend it with JSON Linked Data (JSON Linked Data (JSON-LD)) context. JSON-LD is a Linked Data version of JSON (Sporny et al., 2014) and the context part is used to declare global identifiers and thereby add semantics. Thereby, they suggest

[7] Available at: http://bnode.org/blog/2009/07/08/the-semantic-web-not-a-piece-of-cake

GraphQL-LD – a more expressive version of GraphQL. These recent developments all aim at enhancing the accessibility of the technology and symmarise some of the promising research being conducted in the field.

The formats layer of the stack defines the different serializations of RDF. In RDF 1.0 the format was predominantly RDF/XML, which uses the eXtensible Markup Language (Gandon and Schreiber, 2015). However, with RDF 1.1 there is not one recommended serialisation format which underpins the fact that RDF is a data model and not a format. Figure 2.8 shows that there exists several serialisation formats, where some support multiple graphs (more on this later). RDF in Attributes (RDFa) allows RDF to be embedded into HTML documents and the Terse RDF Triple Language (Turtle) is a commonly used serialization since it is the triple pattern syntax used in SPARQL (Adida et al., 2015; Prud'hommeaux et al., 2014).

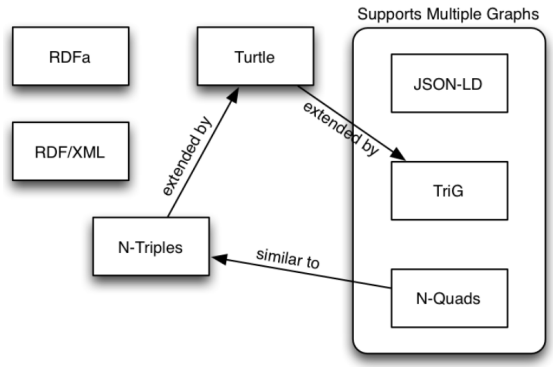


Figure 2.8: Serialisation formats for describing RDF 1.1[8].

The information exchange layer is constituted by RDF. It was first released in 1999, but at the time of writing, the latest revision is RDF 1.1 (Brickley and Guha, 2014). It is a data model consisting of triples in an Entity-Attribute Value (EAV) representation. Each triple consists of a subject (entity), a predicate (attribute) and an object (value) (See Figure 2.9).

The subject denotes the resource which the statement concerns, the predicate is the key to the statement and the object is the value. A triple can be viewed as a simple directed graph where the subject and predicate represent two nodes connected by an edge – the predicate. When multiple triples are defined, the

[8]Figure from https://www.w3.org/TR/rdf11-new/

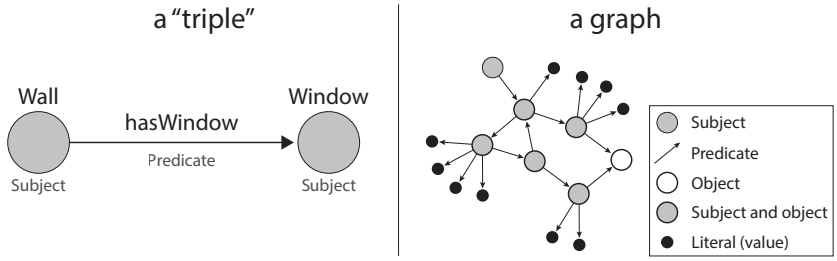


Figure 2.9: Any assertion about a resource in an RDF is described as a triple. Together, these triples form a directed graph.

object of one triple might be the subject of another triple, and together they form a larger directed graph like illustrated at the RHS of Figure 2.9. A predicate connecting two resources is called an object property (owl:ObjectProperty). In (2.1), a triple for assigning an object property is shown in N-Triples syntax. The triple is defined as three IRIs, each surrounded by <angle brackets> and the period indicates that the triple is finished.

<subject> <objectProperty> <object> . (2.1)

A predicate can also connect a resource with a literal property, and in this case it is called a datatype property (owl:DatatypeProperty). The value of an annotation property is always a string, but for datatype properties it is possible to specify a datatype. Common datatypes are often specified with XML Schema Definition (XSD). For example xsd:boolean or xsd:dateTime. More special datatypes to for example denote a mediatype are, however, also supported. Lefrançois and Zimmermann (2016) propose the use of Custom DataTypes (CDT) to describe physical units using the Unified Code for Units of Measure (UCUM) (eg. "0.25 W/(m2.K)"^^cdt:ucum). Their work includes a SPARQL implementation[9] that allows unit conversion as part of the query. For example, a distance can be requested in meters even though it is only stored in feet. In N-Triples, datatype properties are assigned as demonstrated in (2.2).

<subject> <datatypeProperty> "value"^^<datatype> . (2.2)

[9]https://ci.mines-stetienne.fr/lindt/playground.html

Besides from specifying a datatype, it is also possible to describe the language of a literal using ISO639-1 (2002) two-letter language codes. This allows for human-readable descriptions of a resource to be given in multiple languages. For example in Danish:

<subject> <datatypeProperty> "text in english"@en . (2.3)

<subject> <datatypeProperty> "tekst på dansk"@da . (2.4)

RDF must facilitate operation at Internet scale, and therefore, it builds on an OWA (Section 2.4).

The query layer is constituted by the SPARQL query language (Harris and Seaborne, 2013). Building a query is a matter of traversing the KG using triple patterns. In triple patterns, variables are indicated with a question mark as a prefix (e.g. '`?variable`'), and constants are stated explicitly. SPARQL uses the Turtle syntax and all namespaces are defined with `PREFIX` clauses. Describing the full functionality of SPARQL is not the scope of this section, but a few examples are given to demonstrate the basics.

The simplest triple pattern is to specify three variables '`?s ?p ?o`' which will return every single triple in the graph. Restricting the results is achieved by replacing a variable with a constant such as '`<http://someResource> ?p ?o`', which will only return triples having '`<http://someResource>`' as the subject. The pattern can be continued with more triples like it is illustrated in Listing 2.1. Here, the '`?child`'-subject and the '`?father`'-object variables are bound to anything that is connected with a '`ex:hasFather`'-predicate. Next, it is stated that the '`?father`' must also be connected to an '`?uncle`'-variable through a '`ex:hasBrother`'-predicate. The particular SELECT query returns the values of all children and their uncles but only if they have an uncle. The full pattern needs to be matched. Enclosing the second pattern with an '`OPTIONAL{}`'-clause would make the second pattern optional and, therefore, additionally return children with no uncle.

Listing 2.1: SPARQL SELECT query example.

```
PREFIX ex: <https://example.org/familyOntology#>
SELECT ?child ?uncle
WHERE {
    ?child ex:hasFather ?father .
    ?father ex:hasBrother ?uncle .
}
```

Listing 2.2 shows an example of another kind of query: a CONSTRUCT query. This query returns the sub-graph that matches the triple pattern. It can, however, also be used to re-format the existing triples. In the example, the two first patterns in the CONSTRUCT clause return the triple matches. The third

pattern creates new triples in the form '`?child ex:hasUncle ?uncle`' – triples that are not explicitly stated in the KG.

Listing 2.2: SPARQL CONSTRUCT query example.

```
PREFIX ex: <https://example.org/familyOntology#>
CONSTRUCT {
    ?child ex:hasFather ?father .
    ?father ex:hasBrother ?uncle .
    ?child ex:hasUncle ?uncle .
}
WHERE {
    ?child ex:hasFather ?father .
    ?father ex:hasBrother ?uncle .
}
```

Listing 2.3 shows an UPDATE query that deletes all existing '`ex:hasUncle`'-relationships and inserts new ones based on the triple pattern used in the previous examples.

Listing 2.3: SPARQL UPDATE query example.

```
PREFIX ex: <https://example.org/familyOntology#>
DELETE {
    ?a ex:hasUncle ?b .
}
INSERT {
    ?child ex:hasUncle ?uncle .
}
WHERE {
    ?child ex:hasFather ?father .
    ?father ex:hasBrother ?uncle .
    OPTIONAL{?a ex:hasUncle ?b}
}
```

The proof layer contains graph IRIs that describe so-called named graphs. A named graph is practically a container for triples belonging to this particular sub-graph. Thereby, it provides a mechanism to subdivide a KG into smaller KGs. Figure 2.8 shows the three serialisation formats that support multiple graphs. The most basic is N-Quads, which is similar to N-Triples but adds a graph IRI. Triple (2.1) as quad:

$$\text{<subject> <predicate> <object> <graph> .} \tag{2.5}$$

Queries on named graphs are also specified by triple pattern matches, but a `GRAPH` clause is used to specify a specific named graph to evaluate for matches. The graph name can also be specified as a variable, to allow the matching to be run against any of the named graphs.

Named graphs allow for expressing metadata about sub-graphs which according to Carroll et al. (2005) can be used for several purposes. For example to describe

a fine-grained access control which will prevent clients from accessing particular named graphs in a store. Also, the ability to sign a graph for validity or restrict the information usage with property rights are potential use-cases.

The logic layer rests on two legs that each constitutes an approach for reasoning about the facts contained in the KG. The first approach describes logic through procedural rules, and the other uses declarative rules. A procedural rule describes the processing step by step like a recipe and can be defined using the Rule Interchange Format (RIF) (Kifer and Boley, 2013), Shapes Constraint Language (SHACL) (Knublauch and Kontokostas, 2017), or simply as SPARQL CONSTRUCT queries as demonstrated in the *query layer* subsection. SHACL is intended mainly for validation and allows the expression of constraints (e.g. a person can only have one biological mother) in a Closed World Assumption (CWA), similar to the one used by traditional schema languages. It can, however, also be used to express procedural rules, described as SPARQL queries. This functionality does, however, not have the status of W3C recommendation (Knublauch et al., 2017). Declarative rules are defined using agreed upon terminology described in ontologies (Hitzler et al., 2012). Ontologies defining such rules include RDFS, OWL and Simple Knowledge Organization System (SKOS), and each of these provide different levels of expressivity. In the following, RDFS and OWL are described in detail as these are fundamental for describing an ontology.

The **RDF Schema (RDFS)** was first recommended by W3C in 2004, but at the time of writing, the latest revision is RDFS 1.1 (Brickley and Guha, 2014). It provides the properties rdfs:label and rdfs:comment that can be used to annotate classes and properties of an ontology with a human readable description (preferrably in multiple languages). Further, it includes taxonomy to describe a hierachy between classes and properties using rdfs:subClassOf and rdfs:subPropertyOf respectively. A resource which belongs to a certain class, also belongs to the super-classes of this class, and the same mechanism applies for properties.

$$\forall\ \mathsf{subClassOf(c1,c2)}\ \wedge\ \mathsf{type(s,c1)}\ \rightarrow\ \mathsf{type(s,c2)} \tag{2.6}$$

$$\forall\ \mathsf{subPropertyOf(p1,p2)}\ \wedge\ \mathsf{p1(s,o)}\ \rightarrow\ \mathsf{p2(s,o)} \tag{2.7}$$

For properties, it is possible to define an rdfs:domain and rdfs:range. The domain of a property specifies what type of resource this property is assigned to – i.e. what class it belongs to. For example, by defining the domain of ex:hasChild-property as ex:Parent, it is declared that anything that has a child is a parent. rdfs:range is similar, but instead describes the type of the range. For ex:hasChild it can, therefore, be used to assign an ex:Child class to the range of the property.

$$\forall\ \mathsf{domain(p,c)}\ \wedge\ \mathsf{p(s,o)}\ \rightarrow\ \mathsf{type(s,c)} \tag{2.8}$$

$$\forall\ \mathsf{range(p,c)}\ \wedge\ \mathsf{p(s,o)}\ \rightarrow\ \mathsf{type(o,c)} \tag{2.9}$$

RDFS also contains the rdfs:seeAlso predicate which is used to relate to another resource that might provide additional information about the subject, without entailing any inference.

The **Web Ontology Language (OWL)** was first recommeded by W3C in 2004, but at the time of writing, the latest revision is OWL 2 (W3C OWL Working Group, 2012). It has its origins in the DARPA Agent Markup Language (DAML) and accompanying Ontology Inference Layer (OIL) (DAML+OIL) funded by the United States (US) Defense Advanced Research Projects Agency (DARPA) (Mcguinness et al., 2002).

OWL 2 exists in three different profiles with different levels of expressivity: OWL 2 EL, OWL 2 QL, and OWL 2 RL. Each of these are less expressive than the full OWL 2 Description Logic (DL), and choosing one is a tradeoff between expressivity and performance. RDF and RDFS are covered by all OWL profiles.

Class membership is not exclusive, and hence a resource can belong to several classes. OWL includes axioms to describe equivalence (owl:equivalentClass) or disjointness (owl:AllDisjointClasses) between classes. OWL also has several means to describe restrictions, which is useful with an OWA entailing the default stance, that anything is possible until otherwise is stated.

OWL classes can be applied to properties in order to specify that they are for example *symmetric* (owl:SymmetricProperty), *asymmetric* (owl:AsymmetricProperty) or *transitive* (owl:TransitiveProperty). Again, due to the OWA it is a stronger notion to be asymmetric than being non-symmetric. It is also possible to describe that a property is owl:inverseOf another property.

OWL provides the means to create complex classes using constructors such as *intersection* (owl:intersectionOf), *union* (owl:unionOf) and *complement* (owl:complementOf). These can be used in combination with owl:equivalentClass to infer that a resource belongs to a certain class if it, at the same time, belongs to *all*, *either of*, or *neither of* another set of classes respectively. Combined with rdfs:subClassOf they can further be used to describe that a resource belonging to a certain class also belongs to the *intersection* of a set of other classes. Classes and properties can also be inferred by specifying property restrictions. Thereby, it is, for example, possible to infer a 'colour' property with value 'red' for all instances of 'RedCars'.

Also, it is possible to specify cardinality constraints on properties with owl:cardinality, owl:minCardinality and owl:maxCardinality and by classifying a property as an owl:FunctionalProperty or owl:InverseFunctionalProperty it is possible to restrict the number of unique property occurrences to/from a resource. The functional property 'has biological mother', for example, restricts that a child can only have one biological mother. This is, however, not a restriction in the traditional database fashion where it prevents the creation of an extra biological mother. The OWA will instead deduce that if a child has two biological mothers, 'Maria' and 'Molly', then 'Maria' and 'Molly' represent the same person. Therefore, precaution must be taken when creating new statements in the KG. This can either be handled at application level or by using procedural rules such as SHACL to temporarily "close the world".

Bringing it Together

To conclude this section, a short data modelling example is given. Listing 2.4 describes some assertions about the author of this work using the URL of his Researchgate profile as IRI. The same RDF triples are shown both in the primitive N-Triples syntax and the more sophisticated and compact Turtle syntax.

Listing 2.4: N-Triples and Turtle syntax.

```
# N-Triples syntax
<https://www.researchgate.net/profile/Mads_Holten_Rasmussen>
    <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    <http://xmlns.com/foaf/0.1/Person> .
<https://www.researchgate.net/profile/Mads_Holten_Rasmussen>
    <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    <http://www.w3.org/ns/prov#Person> .
<https://www.researchgate.net/profile/Mads_Holten_Rasmussen>
    <http://xmlns.com/foaf/0.1/knows>
    <https://www.researchgate.net/profile/Pieter_Pauwels> .
<https://www.researchgate.net/profile/Mads_Holten_Rasmussen>
    <http://xmlns.com/foaf/0.1/firstName>
    "Mads"^^<http://www.w3.org/2001/XMLSchema#string> .
<https://www.researchgate.net/profile/Mads_Holten_Rasmussen>
    <http://xmlns.com/foaf/0.1/birthday>
    "1988-03-19"^^<http://www.w3.org/2001/XMLSchema#date> .

# Turtle syntax
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix prov: <http://www.w3.org/ns/prov#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rg: <https://www.researchgate.net/profile/> .

rg:Mads_Holten_Rasmussen a foaf:Person , prov:Person ;
    foaf:knows rg:Pieter_Pauwels ;
    foaf:firstName "Mads"^^xsd:string ;
    foaf:birthday "1988-03-19"^^xsd:date .
```

Dataset and syntax

Two widely adopted ontologies in the semantic web are used to describe assertions about Mads. The Friend Of A Friend ontology (FOAF) is used to describe social relations and the Provenance Ontology (PROV-O) is used to describe provenance, ie. *"information about entities, activities, and people involved in producing a piece of data or thing, which can be used to form assessments about its quality, reliability or trustworthiness."*[10]

The first assertions define that Mads is an instance of foaf:Person and prov:Person. This is stated using the rdf:type predicate. In N-Triples, the full IRI is used, and the subject and predicate are repeated for both triples. In Turtle, *'a'* is an abbreviation for rdf:type. Using a comma instead of a period at the end of the triple further states that the subject and predicate are repeated in the next triple. A semicolon indicates that only the subject is repeated.

Two datatype properties, foaf:firstName and foaf:birthday are assigned to Mads. The name has datatype xsd:string and the birthday is a xsd:date. Since all namespaces are declared with prefixes in Turtle, the syntax is significantly more readable and compact.

Reasoning

Explicitly stating that Mads is an instance of foaf:Person was not necessary since this is already defined by the rdfs:domain of both foaf:firstName and foaf:knows (not shown in the listing). Since foaf:Person is a rdfs:subClassOf foaf:Agent it can also be inferred that Mads is a foaf:Agent. Also, since the rdfs:range of foaf:knows is a foaf:Person, it can be inferred that Pieter is a foaf:Person and a foaf:Agent. The foaf:knows predicate is defined as a owl:SymmetricProperty meaning that it can also be inferred that '`rg:Pieter_Pauwels foaf:knows rg:Mads_Holten_Rasmussen`'.

Querying

Listing 2.5 shows an example of querying the dataset with SPARQL. The SELECT query returns data in table form. SPARQL endpoints will typically accept content negotiation and return either JSON or XML following a standard formatting (Seaborne, 2013; Hawke et al., 2013). The asterisk simply means that all variables should be returned, but since the query only has one variable, `?class`, all values bound to this will be returned. In this case prov:Person, foaf:Person and foaf:Agent.

[10] https://www.w3.org/TR/2013/REC-prov-dm-20130430/

Listing 2.5: SPARQL SELECT query.

```
PREFIX rg: <https://www.researchgate.net/profile/>
SELECT *
WHERE {
    rg:Mads_Holten_Rasmussen a ?class
}
# RESULT:
# |-------------|
# | class       |
# |-------------|
# | foaf:Person |
# | prov:Person |
# | foaf:Agent  |
# |-------------|
```

Listing 2.6 shows a CONSTRUCT query, which returns the subset of the dataset matching the triple pattern given in the `WHERE` clause. In this case, this means any assertions that have '`rg:Mads_Holten_Rasmussen`' as subject, except for the associated classes since these are filtered out by omitting all rdf:type predicates.

Listing 2.6: SPARQL CONSTRUCT query.

```
PREFIX rg: <https://www.researchgate.net/profile/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
CONSTRUCT
WHERE {
    rg:Mads_Holten_Rasmussen ?p ?o
    FILTER(?p != rdf:type)
}
# RESULT:
# rg:Mads_Holten_Rasmussen foaf:knows rg:Pieter_Pauwels ;
#       foaf:firstName "Mads"^^xsd:string ;
#       foaf:birthday "1988-03-19"^^xsd:date .
```

Considerations

Since the semantic web relies on an OWA, both ontologies and assertions (schemata and data) are open-ended. Linked data prescribes that IRIs are used for naming resources, so the existing web infrastructure can be utilised for distributing the dataset. Thereby, these technologies can enable the 'distributed databases' requested by Zamanian and Pittman (1999).

As demonstrated in the example, shown in Listing 2.4, it is common practice to combine different ontologies to express assertions of a certain topic of interest. Since it is possible to pick terminology that suits a specific assertion, this allows for describing data models in a modular, distributed manner that has not been possible before.

In order to increase interoperability and reduce redundancies, thereby encourag-

ing reuse of the assertions, it is by Lóscio et al. (2012, Sec. 8.9) prescribed that ontologies, already in use by others, are reused. It is further encouraged that codes and terms from agreed-upon standards are used, in order to avoid ambiguity. This entails using ontologies developed and maintained by a standardisation organisations such as Internet Engineering Task Force (IETF), Open Geospatial Consortium (OGC), W3C and so forth. Browsing existing ontologies is possible through repositories such as Linked Open Vocabularies[11].

2.6 AEC Knowledge Graphs

Chapter Recap

At the beginning of this chapter, it was described how project websites provide structure to documents. The level of KM provided with these systems is limited, and human involvement for highlighting changes, and preparing and uploading documents for revision, is still required. Information retrieval requires human processing of a document.

Level 2 BIM implementations (Figure 2.2) provide context with the data, and are therefore capable of representing information. Accessing that information is, however, still a challenge since it is mostly trapped inside proprietary BIM environments. Interoperability is essentially limited to directly accessing the internal data model through application-specific APIs or through exchange of whole models in IFC files.

Lately, software vendors have started to offer cloud services, but these primarily communicate with BIM authoring tools through their APIs, and if they support IFC, the data is parsed to a closed internal data structure. Open source initiatives like BIMServer also do not expose the local schema, and this makes them hard to extend. With NoSQL it is possible to copy the IFC data structure internally, but extending the dataset is still restricted to using the terminology provided by IFC or extending with private terminology, that is not directly interpretable by third-party applications.

With semantic web, a dataset is rarely described by one schema only. Instead, assertions are stated using ontologies from different sources, and thereby, it allows the schema to be modular. Also, ontologies go beyond the capability of database schemata, as they support the description of logic through declarative and procedural rules. Applying these technologies in the context of BIM could

[11] https://lov.linkeddata.es/dataset/lov/

enable machines to interpret the information trapped inside the silo models, and thereby improve the availability by magnitude. This is possible if the AEC industry advances from a document-centric attitude to a data-centric attitude.

> *"Just as the World Wide Web has revolutionized the way we connect and consume documents, so can it revolutionize the way we discover, access, integrate and use data."* (Heath and Bizer, 2011)

This section describes state of the art, in the use of semantic web technologies, in the context of BIM. Thereby, it provides a general understanding of what has been accomplished so far, and what is needed in order to procure comprehensive AEC Knowledge Graphs (AEC-KGs). An AEC-KG is an RDF graph for AEC knowledge representation. Such a graph is capable of capturing knowledge relevant to AEC practitioners, and provides answers to any competency questions these practitioners might have. The term is not described in the literature, but the notion of KGs is widely adopted.

An Ontology for BIM

Fuchs and Scherer (2017) describes that linked data is in principle an integrated system – i.e. using an internal common format for data operations. The data models can be distributed on the www, but still it is possible to query them based on a normalised data format. This has major benefits when operating in a cross-disciplinary domain. Rezgui et al. (2011) argues that ontologies can address some of the CWA-related issues of IFC by providing a richer conceptualisation. They describe that an ontology is to be considered as a living system, and a similar conclusions have been reflected in earlier research projects:

> *"systems for the AEC industry will most likely need to offer dynamic schema definition in the future thus allowing a schema to evolve over time, perhaps in a distributed fashion, such that the semantics and organisation of information satisfies the needs of a diverse, expanding group of end-users and third-party developers."* (Zamanian and Pittman, 1999)

Beetz et al. (2009) also highlights lack of formal rigidness, limited reuse and interoperability, and lack of built-in distribution as reasons for converting the IFC schema to an OWL DL ontology called ifcOWL. Later efforts in formalising

ifcOWL were conducted by Pauwels and Terkaj (2016). The resulting ontologies exist in a version for each release of the IFC schema, and the latest at the time of writing was IFC4 Addendum 2[12]. Each ontology is a direct translation of the EXPRESS schema converted by the EXPRESStoOWL[13] converter and hence is a quite large ontology with more than 1300 classes and 1500 properties[14]. A Linked Data Working Group (LDWG) exists under buildingSMART International (bSI)[15], and the focus of this group is the development of ifcOWL. The official IFC documentation did at the time of writing not include the OWL specifications but they can be generated with the ifcDoc tool[16].

Simplification and Modularisation Efforts

The complexity of ifcOWL complicates query writing, and therefore it is not so straightforward to retrieve data in an ifcOWL AEC-KG. Therefore, there have been proposals for providing a simplified version of it. Both IFC Web of Data (ifcWoD) and SimpleBIM (not to be confused with Datacubist SimpleBIM) are such proposals, and both use post-processing of an ifcOWL-compliant RDF dataset in order to simplify the representation (Mendes de Farias et al., 2015; Pauwels and Roxin, 2016). In ifcOWL, geometry represents the greater part of the file size, and therefore SimpleBIM omits the geometry. Their argument is that geometry is seldom used in a linked data context. They do, however, mention RDF representation of geometry as future research topic of interest. Lastly, Zhang et al. (2018) suggests an approach where procedural rules are applied to infer shortcut-predicates that makes ifcOWL datasets easier to query. They also generate simplified, read-only mesh geometry and deduce geometrically derived properties from the IFC geometry such that it can be referred to directly in queries.

The issue with the non-dynamic schema is not entirely addressed by ifcOWL. Converting IFC to RDF makes it extendable with things from outside the IFC world, but the schema itself is still large and non-dynamic by nature. Terkaj and Pauwels (2017) deal with this particular problem by suggesting a method to make a modular ifcOWL ontology.

[12] http://ifcowl.openbimstandards.org/IFC4_ADD2/index.html#

[13] Source code available at https://github.com/pipauwel/EXPRESStoOWL

[14] According to the WebVOWL online tool: http://www.visualdataweb.de/webvowl/#iri=http://ifcowl.openbimstandards.org/IFC4_ADD2.ttl

[15] LDWG: http://www.buildingsmart-tech.org/future/linked-data

[16] ifcDoc: http://www.buildingsmart-tech.org/specifications/specification-tools/ifcdoc-tool

Transitional approaches

Beetz et al. (2014) takes a different approach by simply extending legacy IFC instance models with linked data. The reasoning behind this is that the implementation of ifcOWL requires a considerable shift in technologies and that the STEP Part 21 format is more suitable for geometry representation than RDF. The hybrid solution involves enritching an IFC file with RDF triples as illustrated in Figure 2.10. Even though this work demonstrates a hybrid approach, it is emphasised that it is meant as a transitional approach towards a full RDF representation.

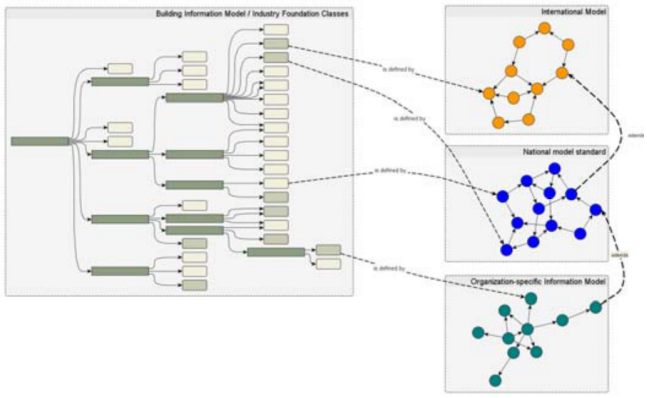


Figure 2.10: Extending legacy IFC instance models (Left Hand Side (LHS)) with linked data. (Beetz et al., 2014)

Fuchs and Scherer (2017) present another transitional approach with Multimodels that combine heterogenious models of different domains. The Multimodel approach simply uses the linked data infrastructure to establish links between data elements in the different models. The links only comprise a loose coupling, and therefore, the idea is that domain models are not affected. What is still achieved, however, is access to a federated dataset about any element in the different domain models. The approach is focused on data exchange of task specific federated datasets in containers, and is not capable of representing a full construction project or integrating with live data.

Other Ontologies for BIM

The BIM Shared Ontology (BIMSO) and BIM Design Ontology (BIMDO) together constitute another project focusing on modularity. The ontologies are designed from scratch and have therefore no connection to IFC. BIMSO has a minimal core and builds on the UNIFORMAT II classification system (Niknam and Karshenas, 2017). BIMDO provides specific terminology for design. Unfortunately, these ontologies are not publicly available, and hence per definition

not to be regarded as an ontology. The idea of having a minimal core (except for the dependency of the entire UNIFORMAT II) aligns well with the best practices described in Section 2.5.

Other AEC Ontologies

There exist several ontologies that deal with a specific subset of a building. These are often focused on a specific domain such as *(1)* building automation and smart homes, *(2)* building energy performance, or *(3)* buildings in the context of GIS and smart cities. The project haystack by Charpenay et al. (2015), the brick schema by Balaji et al. (2016), the DogOnt by Bonino and Corno (2008), and the Smart Appliances REFerence (SAREF) ontology (Daniele et al., 2015) all belong in the first category. The Smart Energy-Aware Systems (SEAS) ontologies (Lefrançois et al., 2017) are more in the second category, but SEAS and SAREF both partly cover all categories. In the third category is the cityGeography Markup Language (GML), which also exists in an OWL version[17] as proposed by Métral et al. (2010).

A semantic web mediated BIM can make use of SPARQL for querying the dataset and therefore, there is no need to define a query language for the construction industry alone from scratch. There might, however, be a point in developing a specific extension of SPARQL for BIM as it has also been done with the GIS-specific geoSPARQL[18] by OGC. Zhang et al. (2018) suggests such extension with BimSPARQL.

Making the Transition

It has been highlighted that IFC is not well implemented in BIM authoring tools because of the multiple ways to describe the same geometry (Venugopal et al., 2012) but still, it will inevitably require a substantial shift in technologies to realise a full Level 3 BIM (Figure 2.2) mediated by these technologies (Beetz et al., 2014). The open question is whether a gradual shift can be obtained by enabling interoperability with existing BIM authoring tools during the transition.

A common research concern in the topic of semantic web mediated BIM is how to establish and manage links between sub-models and other data sets (Pauwels et al., 2017). A lot of the research is, however, based on existing BIM models,

[17] http://vgibox.eu/repository/index.php/CityGML_in_OWL
[18] https://www.opengeospatial.org/standards/geosparql

and the situation might be different if the interconnected models were built organically altogether from scratch.

Change management in a framework of interlinked datasets is another open challenge which is mentioned by Törmä (2013) as a future research topic of interest.

Chapter 3

Research design

This chapter describes the research design. Based on the challenges identified in the previous two chapters, a set of research questions describing the research quality criteria are defined. The chapter further defines the overall research methodology and defines three research tasks to be accomplished through the research project.

3.1 Research Questions

The literature study conducted as part of Chapter 2 acknowledges the characteristics and derived issues identified in Chapter 1. It is a recognised perception that the interoperability is inadequate which is likely influenced by the current technological maturity and adoption of the available Building Information Modelling (BIM) systems (BIM level 2). The concern of information being trapped mainly inside the heads of the knowledge workers rather than made digitally accessible is also recognised (Kiviniemi, 2005b; Deshpande et al., 2014). The fragmented, document-centric nature of the information handling constitutes an obstacle for interoperability, and by adopting state of the art technology, it is possible to evolve to a data-centric nature.

BIM provides a data structure that encapsulates semantics about the construction project, but accessing the data is a challenge. One solution is to exchange proprietary files, but then the selection of tools is limited to those provided by that single software vendor. Most BIM authoring tools also provide an Application Programming Interface (API) that allows access to the internal proprietary data model. This approach, however, requires that the developer understands its architecture. Further, if translations between every proprietary software should be enabled through their APIs, it would require n^2 translations. The common BIM exchange format Industry Foundation Classes (IFC) standardises the description of semantics, and thereby provides a common language, but accessing and extending the information is still not straightforward (Beetz et al., 2009; Ma and Sacks, 2016; Zhang et al., 2018). The main reason for this is that IFC is an exchange format, meaning that the internal data model in the software implementations is still proprietary.

When comparing the BIM exchange practice to the one used by web applications, there is a big difference. Web applications have a clear separation between the frontend application and the backend, which contains all the application intelligence and the data model. A photo gallery on a social media platform is generated from a subset of images returned from the billions of photos stored on the servers, and the transaction happens almost instantly. In contrast, current implementations of IFC requires the exchange of the gross file to which a filtering mechanism is afterwards applied. Alternatively, a specialised filter is applied beforehand using an Model View Definition (MVD), but the flexibility with this approach is limited, and only a few MVDs are currently implemented in the software tools. Therefore, the reality is that getting a simple dataset, such as the areas of all spaces, usually entails the exchange of a full discipline model. Further, the model must be opened in special software to extract specific pieces of information. This is necessary because the STandard for the Exchange of Product model data (STEP) format has only been adopted by engineering

disciplines and tools to access the data are therefore not widely available (Zhang et al., 2018). For those reasons, lately, there has been a growing interest in the research field of BIM in the context of (semantic) web technologies (Santos et al., 2017).

The research should ultimately seek to remedy the observed problems that were identified in Chapter 1 and summarised to one research problem in Chapter 2.

> *"How to provide a web-based infrastructure for building data that allows a shared, distributed dataset to grow organically with the construction project, thereby providing insights and traceability as changes occur in a form that is consumable and extensible by any authorised agent[1] involved in the project?"*

The literature study conducted as part of Chapter 2 indicates that semantic web technologies could conceivably be the right fit for overcoming the research problem. Based on this hypothesis, a Primary Research Question (PRQ) and a set of sub-Research Questions (RQs) were defined.

PRQ "How to utilise semantic web technologies to make assertions about a building, its properties and the interdependencies between these in a manner that allows for change management and enhances transparency using widely adopted ontologies in the extent that these are available, and what additional terminology is needed?"

RQ 1 "What existing ontologies can be used to establish a common platform for interdisciplinary information exchange in the Architecture, Engineering and Construction (AEC) industry and is there a need for extending these?"

RQ 2 "How is it possible to get information from the BIM authoring tools used today to the new platform?"

RQ 3 "When describing a project using the common platform, how is it possible to capture the design information as well as interdependencies while allowing it to change and evolve?"

[1] An agent is in this context a software that acts on the behalf of a user.

3.2 Methodology

The primary goal of this research is to provide viable answers to all the RQs. Ultimately, this will demonstrate the advantages in a glssemantic web-based BIM where the data model is in the centre instead of the proprietary models used by today's BIM authoring tools. Also, it will illustrate that the advantages in BIM goes beyond geometry. There are numerous semantic relationships in a building and the processes around the design of it that can be captured in data models. Ultimately, this will enable the industry to overcome the observed problems from Chapter 1. In order to structure the research work, it first needs to be defined what particular body of knowledge is missing for the semantic web-based BIM to be obtainable.

The Concept
Figure 3.1 illustrates the overall web-based BIM vision. The figure eases the division of the overall vision into smaller bits that can be investigated as separate tasks. The central element is the AEC Knowledge Graph (AEC-KG) in which the knowledge is captured in the form of Resource Description Framework (RDF) triples. This Knowledge Graph (KG) should be capable of capturing knowledge and providing answers to design questions and thereby ultimately provide a solution to the overall research problem. The colours of the nodes (classes) and edges (properties) in the AEC-KG indicate to what terminology cloud they belong.

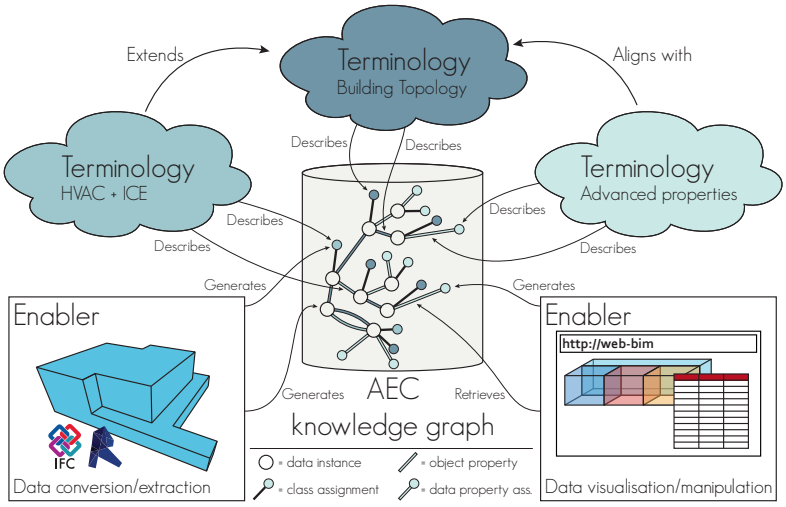


Figure 3.1: The overall concept.

Terminology

To establish the AEC-KG, the necessary terminology and enablers must first be made available. The three terminology clouds each cover a particular domain of interest which needs to be captured in the AEC-KG. The middle contains terminology to describe the core elements of a building and their internal relationships. The specific terminology layer for the Heating, Ventilation and Air Conditioning (HVAC) and Indoor Climate and Energy (ICE) domains extends the general building layer. Hence, the building layer: the core layer, must be flexible enough to allow extensions. Further, terminology to describe evolving, interdependent design properties is needed in order to describe property provenance data. Thereby, insights and traceability, as requested in the research problem, can be achieved.

Enablers

The two enablers are practical software implementations that operate with the assertion layer of the AEC-KG. In order to connect to legacy software, data needs to either be extracted directly from the BIM authoring tools, or generated by converting an IFC file. A third option is to generate the assertions from scratch, which is an approach investigated with the second enabler, that consists of tools for data visualisation and manipulation. The second enabler should mediate the communication between the engineer and the AEC-KG by visualising the content as well as enabling the creation of new assertions and updating existing ones. Thereby, it also serves as a demonstration of how future BIM authoring tools can interact with an AEC-KG.

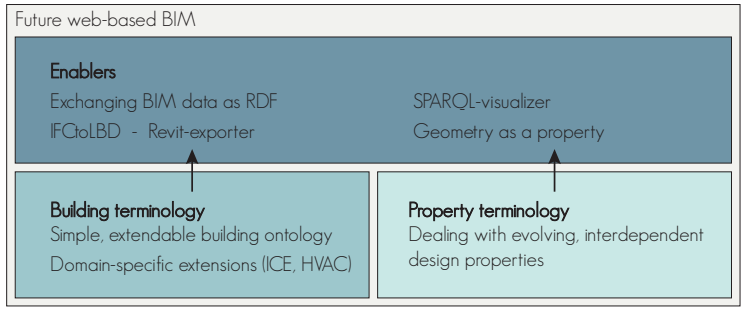


Figure 3.2: Research tasks.

Figure 3.2 simplifies the main content of Figure 3.1. It shows the research in three layers and indicates how they are related. The two bottom layers constitute the terminology research, and are defined from the clouds in Figure 3.1. Research in these layers is focused on providing answers to RQ 1 and 3. The terminology layers constitute the foundation of the enabler layer. Without this

foundation, it is not possible to describe the AEC-KG. The enablers are software artefacts that allow data to flow from BIM authoring tools to open knowledge representations as well as representing and consuming this data. Thereby, the top layer focuses on RQ 2. Without the enablers, it would not be possible to establish or interact with the AEC-KG or demonstrate the capability of the architecture to a wider audience. Therefore, this layer is especially important from dissemination and industrial perspectives.

Community and Industry Involvement

A research community group exists under the World Wide Web Consortium (W3C)[2]. This group focuses on the application of semantic web technologies in the context of buildings. In order to establish a solid foundation, early involvement in this group was procured. Thereby, all the work conducted as part of the research could be aligned with ongoing work by other researchers. Further, the contributions could be discussed and evaluated in a wider forum where feedback could be taken into the next design loop. Thereby, each Research Task (RT) became an iterative task which was continuously revisited and refined during the research in a prototyping approach (Alavi, 1984). Especially in regard to ontology development, it was valuable to have discussions with the community group. As described by Hoekstra (2009), ontology specification is about *"reaching consensus on the meaning of terms"*, and hence becomes a *"social activity"*. Noy, McGuinness, et al. (2001) supports this view by describing the process of developing an ontology as an iterative process that will likely continue through the lifecycle of the ontology.

According to H. Liu et al. (2017) it requires the involvement of domain experts to build a comprehensive domain ontology. Since this research was conducted as an industrial PhD with the involvement of a hosting company: Niras, in which the author was already employed as a consulting engineer, a wide board of business professionals were available. Also, the professional background of the author in Architectural Engineering and not Computer Science, meant that the challenge did not consist in knowing the domain, but rather in knowing the technology. Having direct contact with domain experts in a large consulting engineering company throughout the research meant that the different hypotheses could be tested, validated and discussed.

[2]https://www.w3.org/community/lbd/

Knowledge Engineering Methodology

The terminology research involves development of ontologies. There exists several methodologies for developing ontologies, and Casellas (2011) lists 15 of these. Some of them are purpose-specific and others are more general. Since the author is already familiar with the domain, and given the ontologies's limited scope, the Skeletal Methodology by (Uschold and Gruninger, 1996), illustrated in Figure 3.3, was chosen.

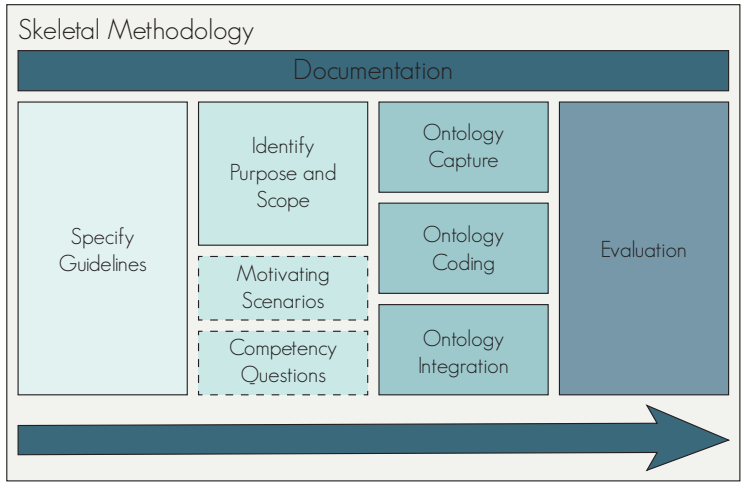


Figure 3.3: Uschold and Gruninger (1996)'s Skeleton Methodology as illustrated by Hoekstra (2009).

The Skeletal Methodology defines four overall steps of ontology development. Part 1 is about considering the intended use and the reason for building the ontology. Part 2 is about identifying the purpose and scope of the ontology, thereby providing the framing for the ontology to be built. This part is the specification part which involves capturing the motivating scenarios and defining a set of informal competency questions. Informal competency questions set the expressiveness requirements of the captured knowledge in the form of questions, and describing these informally means that they are captured in human language rather than in the formal language of the ontology. Part 3 is where the ontology is built, and with RDF, RDF Schema (RDFS) and Web Ontology Language (OWL), the capturing, coding and integration parts can be integrated in one work flow.

Other methodologies such as the Ontology Development 101 by Noy, McGuinness, et al. (2001) enforce the importance of having early considerations of what ontologies already exist, but with the Skeletal Methodology, this is not considered until part 3. With the emergence of OWL, the alignment process has changed, and as described by Lóscio et al. (2012), integration with existing ontologies should be considered earlier. Therefore, this part was in practice considered as early as part 1 but formalised in part 3. Part 4 is the evaluation part in which the competency questions are formalised and tested as queries against the captured knowledge. The ontology, including guidelines, purpose, scope and content should be documented. According to the second and third rule for linked data by Berners-Lee (2006), this should be implemented so that looking up the International Resource Identifier (IRI) of a resource in a web browser returns relevant information.

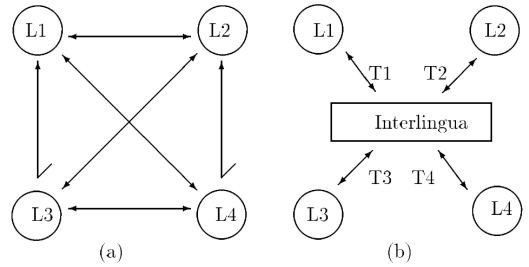
3.3 Research Tasks

The RTs were briefly described in Section 3.2. In this section, the methodologies behind the RTs are described in detail. Each RT is structured so that it examines one or more of the RQs that were formulated in Section 3.1.

RT 1: Ontologies for Building Information

This task covers the work concerned with the lower LHS of the diagram in Figure 3.2. The work originates from the existing efforts in providing an ifcOWL ontology and other efforts in ontology development for the building domain. The goal is to provide core terminology for describing a building as a whole and in particular in the domains of ICE and HVAC. The terminology should be defined in an extendable, modular fashion that satisfies the Data on the Web Best Practices (Lóscio et al., 2012).

Initially, during the guideline specification part of the ontology development, existing ontologies in the AEC domain were investigated. The hypothesis was that the individual ontologies would all redundantly define the same terminology for describing the overall topology of a building. The purpose was then to establish a minimal ontology that just covered the core terminology necessary to describe the topology of a building – a Building Topology Ontology (BOT). Thereby, it would constitute an interlingua for communication as illustrated in Figure 3.4 by Uschold and Gruninger (1996). Using and extending this interlingua with existing and future domain-specific ontologies was also part of the scope.



To translate from language L_i to L_j and vice versa, a translator is required between L_i and the inter-lingua and another between the inter-lingua and L_j. Thus, given n languages, only $O(n)$ translators are required, not $O(n^2)$.

Figure 3.4: Ontology as inter-lingua by Uschold and Gruninger (1996).

The minimal ontology was formalised as RDF triples in a middle-out approach as described by Uschold and Gruninger (1996) where the most fundamental terms are defined first. It was brought to the W3C LBD Community Group (W3C LBD CG) who adopted it and subsequently the terminology has been revisited and refined to meet the requirements of its users.

For dissemination purposes, this RT also includes the development of examples demonstrating how the ontology is extended with domain specific ontology.

RT 2: Connecting to Legacy Software

This task covers the work constituted by the top layer of the diagram in Figure 3.2 and concerns the procurement of assertions from legacy BIM models as well as making the knowledge accessible for users. In general, this task investigates how the users can interact with the AEC-KG.

Connecting to legacy software was a key deliverable of this RT. In order to deliver on this, however, a solid programming and application development foundation was needed. This was procured through a blended learning process where a mix of online courses, tutorials and classroom courses were attended (Bersin, 2004).

Each software artefact was developed with a prototyping approach (Alavi, 1984). All prototypes were made publicly available on GitHub, and feedback from users was used in continuous development. Some prototypes were used internally in the hosting company, and through different workshops, the prototypes were optimised.

RT 3: Dealing With Evolving, Interdependent Design Properties

This task covers the work concerned with the bottom RHS of the diagram in Figure 3.2. This work should investigate modelling approaches for properties that evolve and the interdependencies between these properties.

The initial discussions for the guidelines of this Ontology for Property Management (OPM) were conducted in a workshop in May 2017 with Maxime Lefrançois at the École des Mines de Saint-Étienne. The intention of the workshop was to outline the guidelines for a Flow System Ontology (FSO)[3], but the focus shifted with the realisation that flow systems are highly dependent on the inheritance of properties from the context in which they operate. Therefore, the work on a more general ontology for describing evolving, interdependent properties of design projects was initiated.

During the workshop, the first two parts of the Skeletal Methodology (Figure 3.3) were discussed: the guidelines were specified and a set of informal competency questions were defined. This work was highly inspired by the existing Smart Energy-Aware Systems (SEAS) ontologies.

The concepts were discussed in the W3C LBD CG, and the approach of assigning property states for evolving properties was well received. This was documented in a workshop paper for Linked Data in Architecture and Construction (LDAC), and the feedback given at this workshop was valuable for the further development. The audience of this workshop is evenly distributed between academia and the industry, and the practitioners acknowleged the problem in scope.

[3]Repository available at https://github.com/thesmartenergy/seas/pull/21

Chapter 4

Results

This chapter evaluates on the findings related to each of the three research tasks described in the research design. The evaluation represents a summary of what is described in detail in the associated papers.

In this chapter, the main results of the Research Tasks (RTs) from Chapter 3 are presented. Each RT constitutes a work package that was conducted in order to provide an answer to the Research Questions (RQs). The main outcome of the RTs is the papers included in Chapter 6. The full documentation of the problem in scope and the proposed solutions are described in detail in the respective papers.

4.1 RT 1: Ontology for Building Information

The Building Topology Ontology (BOT)

The task of defining a Building Topology Ontology (BOT) has been revisited several times during the overall research project. The argumentation for a minimal, extendable Architecture, Engineering and Construction (AEC) ontology was initially described in paper BOT1 (M. H. Rasmussen et al., 2017b). It is intended to provide a common core vocabulary for describing a building in a minimal, extendable way. As noted by Uschold and Gruninger (1996), *"due to different needs and background contexts, there can be widely varying viewpoints and assumptions regarding what is essentially the same subject matter."* – i.e. different practitioners have different needs for knowledge capture. Therefore, each practitioner should be able to extend the generic terminology with domain-specific terminology such that an item can be described at different levels of detail with the most general being well established by BOT.

In paper FORGE (M. H. Rasmussen et al., 2017a), it was later demonstrated how a web-based Building Information Modelling (BIM) model can be queried based on BOT terminology. This demo was effective for dissemination purposes, as it eased the understanding of the topological relationships that BOT contains.

The initial version of BOT, illustrated in Figure 4.1, described a building as consisting of the building, storeys, spaces and elements. This terminology describes two relationships between a space and an element: adjacency or containment. A building can have storeys that can have spaces, and axioms of the ontology infer that buildings and storeys also implicitly have a relationship to the elements contained in a space of that building/storey.

Schneider (2017) identified some challenges in defining alignments between BOT and ontologies in the smart homes domain, and this, in combination with plenum discussions in a sub-group of the W3C LBD Community Group (W3C LBD CG) lead to some changes in the initial ontology which were documented in paper

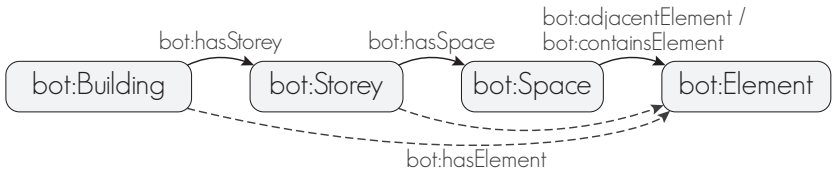


Figure 4.1: The initial version of BOT described in (M. H. Rasmussen et al., 2017b).

BOT2 (M. H. Rasmussen et al., 2017c). This revision, illustrated as *v0.2* in Figure 4.3, included some major changes. The introduction of the general class bot:Zone as a super-class of bot:Building, bot:Storey and bot:Space and adding a new sub-zone bot:Site along with the ability to express adjacency and containment between zones constituted the most significant change. The existing relationships between zones: bot:hasSpace, bot:hasStorey and a new bot:hasBuilding were defined as sub-properties of the generic, transitive bot:containsZone. The rdfs:range of these properties was further relaxed to the new bot:Zone class, thereby allowing spaces to be assigned directly to a building using bot:hasSpace. The concept of bot:Interfaces was also introduced. Interfaces enable quantification of a relationship between zones, elements or a zone and an element. These are excluded from Figure 4.3, but Figure 4.2 demonstrates the use of interfaces for describing the relationships between a pipe segment and the zones and elements it intersects. This can, for example, be used for automatic creation of thermal insulation or fire sealing based on properties of the zone, the pipe and the medium carried by the pipe.

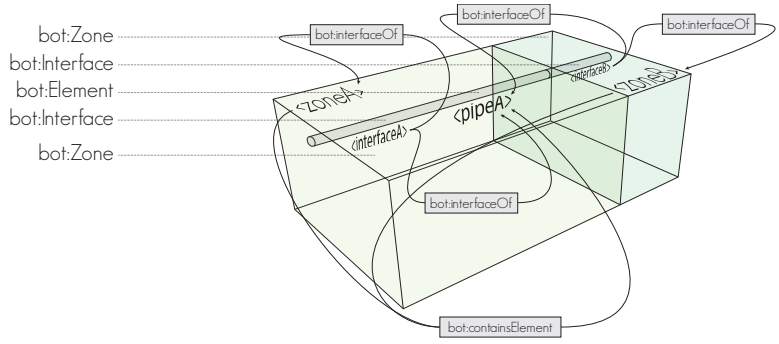


Figure 4.2: An example application of BOT — quantifying the interface between spaces/walls and an intersecting pipe.

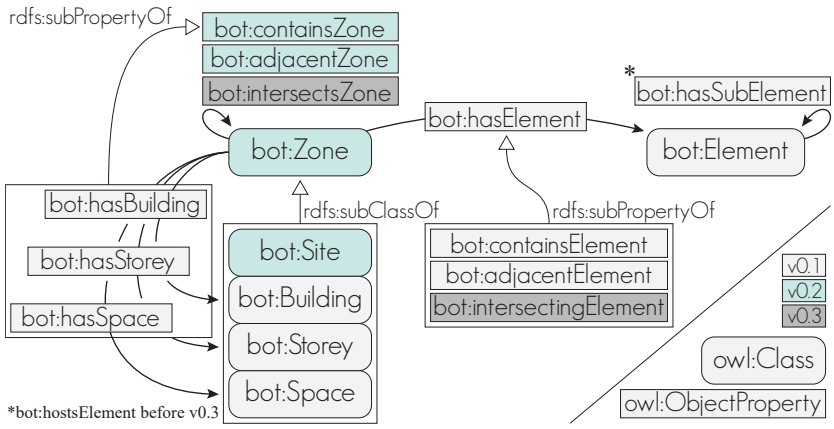


Figure 4.3: BOT has evolved throughout the project.

There have been minor subsequent changes in BOT. bot:hasElement is reintroduced from *v0.1* as a super-property of bot:adjacentElement, bot:containsElement and a new bot:intersectsZone. The intended use of the new property is for describing stairwells that intersect multiple storeys. Predicates have further been added to assign Three-Dimensional (3D) geometry to zones and elements. These additions are described in detail in paper BOT3 (M. Rasmussen et al., 2019b).

BOT has been used throughout the research project as a core ontology. Figure 4.4 shows a demonstration from paper SSN (M. H. Rasmussen et al., 2018b) that shows how BOT can be used together with the Semantic Sensor Networks Ontology (SSN) and Sensor, Observation, Sample, and Actuator Ontology (SOSA) to describe observations from sensors and actuators in the devices' context of a building. This example demonstrates the power of linked data that allows a model to be described with multiple schemas. The model in Figure 4.4 is described with classes, property types from five different namespaces (i.e. BOT, props[1], XML Schema Definition (XSD), GEO[2] and Custom DataTypes (CDT)[3].

In paper REQ (M. H. Rasmussen et al., 2018a), BOT was used to describe space requirements for a future building. Schneider et al. (2018) uses BOT in the context of the Open Smart Home (OSH) dataset and Bonduel et al. (2018a) uses it in a novel approach where the topology of an existing building is captured before a BIM model exists.

[1]W3C LBD CG ontology for properties: https://github.com/w3c-lbd-cg/props
[2]OGC GeoSPARQL: http://www.opengis.net/ont/geosparql#
[3]CDT documentation: https://ci.mines-stetienne.fr/lindt/v2/custom_datatypes

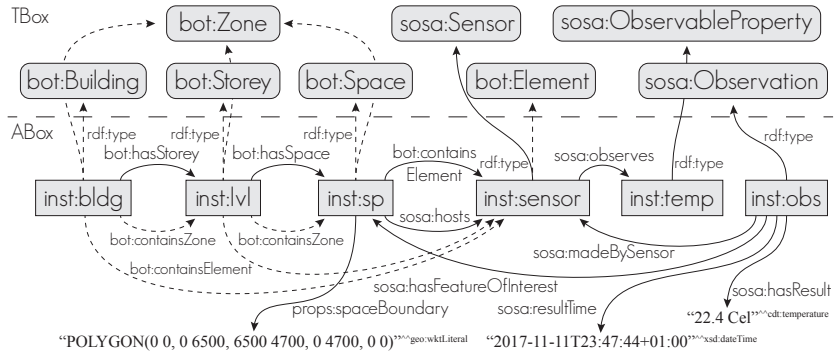


Figure 4.4: BOT and SSN/SOSA for modelling sensors and observations in their context of a building's spaces.

Part of this task concerns providing proper documentation so the ontology can be understood and adopted by others. BOT[4], including documentation[5] is shared publicly online.

Domain-Specific BOT Extensions

BOT has been extended with domain- and project-specific extensions, but no published ontologies were developed as part of this research. Hence, what is presented in this section does not constitute a major research contribution. Instead it demonstrates how two specific domain extensions related to the Heating, Ventilation and Air Conditioning (HVAC) design task described in Chapter 1 can be described based on BOT terminology. Both extensions were used in the proof of concept radiator sizing application from paper OPM2 (M. Rasmussen et al., 2019a), which is also described in Section 4.2. Since the ontologies are not documented in papers, a detailed description of their concepts is provided here.

Flow Systems Ontology (FSO)

In paper BOT3, concepts from a fictive Flow System Ontology (FSO) were defined to demonstrate linking approaches for extending BOT with domain specific terminology. A property: fso:heatedBy ($\sqsubseteq$ bot:containsElement), and a class: fso:Heater ($\sqsubseteq$ bot:Element) is defined for this purpose that examplifies how vari-

[4]Ontology available at https://w3id.org/bot/bot.ttl
[5]Documentation available at https://w3id.org/bot

ous viewpoints as described by Uschold and Gruninger (1996) can coexist in an AEC Knowledge Graph (AEC-KG) based on BOT.

The development of an actual ontology, used for the radiator sizing application (Section 4.2), was initiated based on preliminary work on extending the Smart Energy-Aware Systems (SEAS) ontology[6]. In the following, the principal design thoughts behind the ontology are described in detail.

Overall, a flow system consists of consumers, sources and the distribution system. A consumer either consumes energy, as it is the case for a closed system, consumes mass, as it is the case for an open system, or consumes a mix of mass and energy. Except for evaporation and leaks, a closed system does not consume mass – it simply circulates the fluid between the source and the consumers as energy is consumed. Heating and cooling systems are examples of closed systems like the one illustrated in Figure 4.5. Domestic water systems and ventilation systems where the air is supplied isothermally, are examples of open systems, that consume mass. Ventilation systems can also be used for heating and cooling, and in this case, they are a mixed system.

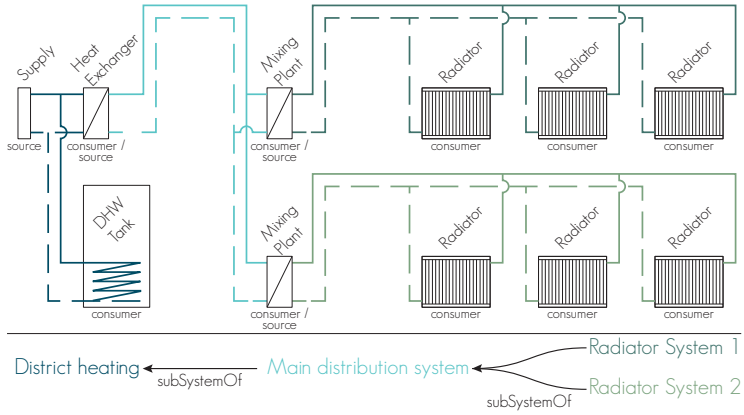


Figure 4.5: HVAC systems

A closed system typically consists of several smaller sub-systems as illustrated in Figure 4.5. The figure shows two radiator systems where each radiator consumes heat. The source in both radiator systems is a mixing plant, which constitutes an interface to another system – the main distribution system. The mixing plants are the energy sources in the radiator systems and the consumers in the main distribution system. The source of the main distribution system is then the main heat exchanger, which constitutes an interface to the main supply

[6] https://github.com/thesmartenergy/seas/blob/505d7136a725d5d40e145d42c9f473065483167d/src/main/ontop/1.0/FlowSystemOntology-1.0.ttl

from the district heating. The Domestic Hot Water (DHW) tank is connected directly to the main supply. Looking at the building isolated, this is the main source of energy. In the district heating distribution system, however, it is a consumer.

All systems are instances of the fso:FlowSystem class that has fso:HeatingSystem as a specific sub-class. Consumers are instances of the fso:FlowTerminal class and a fso:HeatEmitter is a specific sub-class of this class. A fso:FlowTerminal can be a fso:MassConsumer or an fso:EnergyConsumer and since the two classes are not disjoint it can also be both. The fso:servedBy property ($\sqsubseteq$ bot:containsElement) defines a relationship between a space and a flow terminal located in the space.

FSO provides explicit terminology to describe the anatomy of a flow system. In combination with procedural rules, it can be extended to define derived technical characteristics of a flow system explicitly. Figure 4.6 illustrates the different interdependencies that can automatically be derived by such rules. The water flow through a radiator is dependent on the desired heat output of the radiator and the temperature and thermal properties of the circulated water. This flow sets the boundary condition for the distribution system supplying the radiator with water. At any point in the distribution system, the total flow can be inferred, and if the pipe size is known, the water velocity can be calculated. High velocity means noise and risk of corrosion dependent on pipe material and rules can be set up to monitor these conditions. If the pipe material and its roughness are known, the friction pressure drop can further be calculated. , and higher pressure drop means bigger pump and higher energy consumption. All these interdependencies can be modelled in a consistent way if standard terminology is available.

Indoor Climate and Energy (ICE)
The ICE extension of BOT is currently limited to modelling information related to heat loss assessment of the spaces of a building. It contains class: ice:ThermalEnvironment ($\sqsubseteq$ bot:Zone) which is an environment that has some relevance to the indoor climate. Such environments include climatised zones of a building and their adjacent zones which they exchange energy with. Instances of this class hold information such as the ambient design temperature for heat loss calculations, weather data, infiltration rates, indoor climate requirements and so forth.

Another class, ice:ThermalEnvelope ($\sqsubseteq$ bot:Interface), is used for capturing the heat transmission between a climatised zone and its adjacent ice:Thermal Environment. ice:ThermalEnvelope instances can represent both heat transfer surfaces (One-Dimensional (1D) heat losses) and cold bridges such as linear (Two-

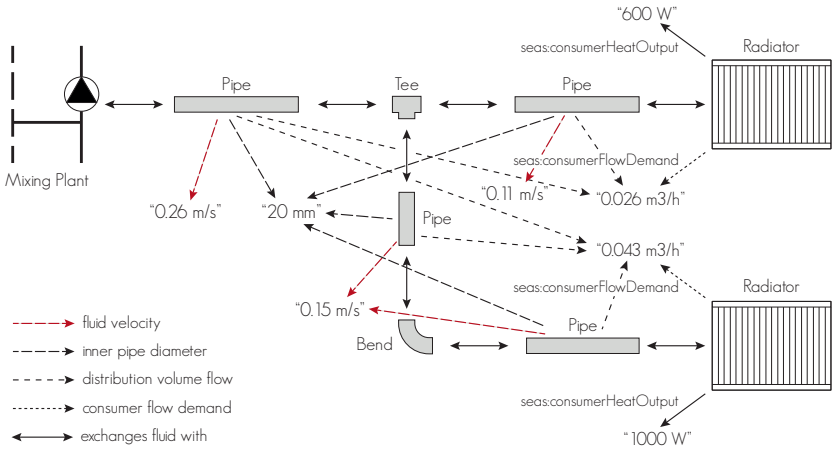


Figure 4.6: Elements in a flow distribution system.

Dimensional (2D)) and point (3D) heat losses. Figure 4.7 illustrates one- and two-dimensional heat losses in a plan section. The heat transfer through a plane homogeneous wall is one-dimensional, but in the corner where the two walls meet as well as in the assembly between the wall and the window there are 2D heat transfers. The Danish code, DS418 (2011) uses a simplification where the corner losses are accounted for by using a gross heat transfer area of the wall as indicated with measurement A_1 which indicates a need for even more specific terms such as a ds:ThermalEnvelope ($\sqsubseteq$ ice:ThermalEnvelope). With this, the ICE ontology demonstrates an application example for bot:Interfaces. Other applications such as acoustics could be modelled with similar BOT extensions.

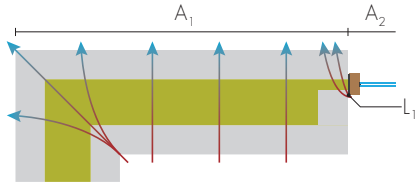


Figure 4.7: Plan section of a wall and a window showing 1D and 2D heat losses.

Figure 4.8 shows how the heat transfer between two zones and the outdoor environment can be modelled using the ICE ontology. The two ice:ThermalEnvelope instances each have relationships to *(1)* the climatised thermal environment described with the ice:surfaceInterior ($\sqsubseteq$ bot:interfaceOf) predicate, *(2)* the outdoor environment described with the ice:surfaceExterior ($\sqsubseteq$ bot:interfaceOf) predicate and *(3)* the element represented by the surface described using the

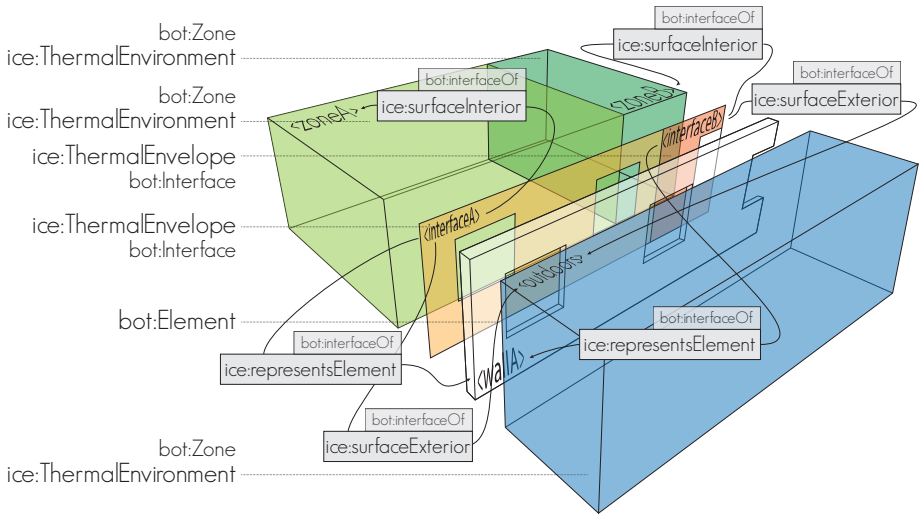


Figure 4.8: ICE specific extensions of BOT

ice:representsElement ($\sqsubseteq$ bot:interfaceOf) predicate. A cold bridge has an additional relationship since it represents two elements and point losses represent three elements.

Key Findings

The preliminary research in existing ontologies in the scope of buildings rendered a good overview of the research that has already been conducted by other actors. Also, this was a good introduction to ontology engineering and best practices for linking approaches. Therefore, it provided a good foundation for exploring the subject. It was discovered that several ontologies redundantly define concepts such as building, storey and space as expected. Therefore, the simple extendable BOT ontology was suggested.

The benefit of the middle-out ontology design approach (Uschold and Gruninger, 1996) was experienced in the continuous development of the ontology. Some concepts needed more general additions whereas other needed further specification. This acknowledges middle-out as a good approach to ontology modelling.

4.2 RT 2: Connecting to Legacy Software

With the terminology in place, this work consisted in *(1)* establishing datasets that follow the principles described in Section 4.1 and *(2)* demonstrating how this data can be used. This work involved development of various software artefacts. A full list of these artefacts is available in Appendix A.

Freeing Data from Native BIM Authoring Tools

Establishing an AEC-KG can be achieved by generating it from scratch or by using an application designed for that. However, since BIM models from legacy systems already exist, as a first step, it is more valuable to make use of these models. This section describes software artefacts for generating AEC-KGs from legacy software.

Revit Exporter
This artefact is a Resource Description Framework (RDF) exporter for the BIM authoring tool AutoDesk Revit. The revit-bot-exporter[7] was developed as part of the work for paper FORGE. It has later been extended to export more topological relationships, 2D space geometry described as Well-Known Text (WKT) formatted literals and 3D mesh geometry for elements and spaces described as OBJ formatted literals. Sample files exported with the tool can be found in the BOT-Duplex-house repository[8]. This repository was generated as part of the work for paper BOT3.

A version that sends the triples directly to a Representational State Transfer (REST)ful web-server was also develped. Figure 4.9 shows operations performed on the server for doing so. Basically, it performs a comparison between the received triples and the triples in the AEC-KG. The AEC-KG uses Ontology for Property Management (OPM) (Described in Section 4.3) to model properties, and with this architecture, it is possible to only save properties where the value has changed. The Revit-web-server implementation is documented in paper OPM2 and demonstrated in a video available online[9].

[7]Repository available at https://github.com/MadsHolten/revit-bot-exporter
[8]Repository available at https://github.com/MadsHolten/BOT-Duplex-house
[9]Video available at https://youtu.be/6Ohuw16bZrQ

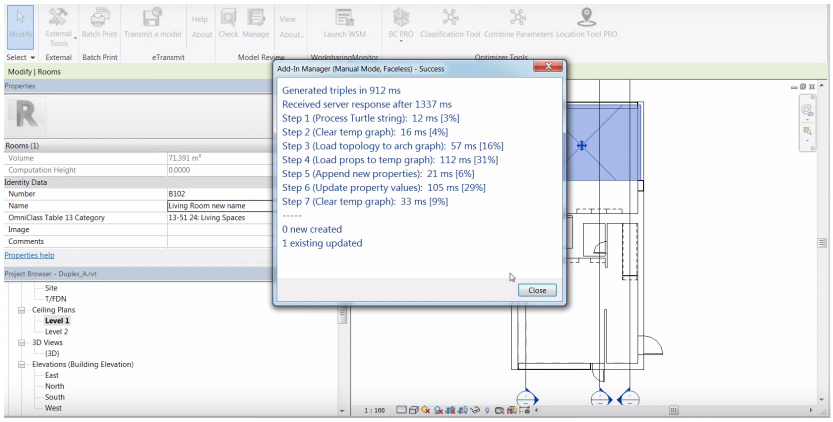


Figure 4.9: Revit synchronisation with OPM compliant AEC-KG. The particular change only affects one property to which the server automatically assigns a new property state.

Geometry as a Property

This artefact was developed while investigating the possibility of exporting 3D geometry as OBJ-formatted literals. There exist exporters that convert a full BIM model to mesh geometry in this format, but the approach investigated here defines mesh geometry for each object (zone or element) individually. Thereby, it is possible to assign one or more 2D or 3D representations of the same resource to, for example, capture different Levels of Detail (LoDs). Also, it is possible to make use of the element and zone aggregation mechanisms of BOT to attach geometry both to individuals and aggregations of these. For example, a building or a storey, which only exist as aggregations of all the contained spaces and elements, can have geometry itself (possibly generated from all the sub-geometries). An Air Handling Unit (AHU), which is an aggregated device consisting of fans, filters, a heat exchanger and so forth, is another example of an element with multiple geometrical representations.

The artefact demonstrates a proof of concept for rendering only the particular geometry that is requested instead of using a filtering mechanism in post-processing. When geometry is just a property, a user can request specific elements' geometry (e.g. columns and walls at a particular level) from the server and render exactly that. Figure 4.10, for example, shows only the walls that are adjacent to a specific space. Linked Data allows fragmented datasets, and hence the query can even request data from different domain models. This opens new opportunities that are not possible with today's BIM implementations and cloud solutions. Several prototype applications were developed and two libraries for the popular JavaScript (JS) framework Angular are made publicly available at

Node.js Package Manager (NPM). The ng-plan[10] (Figure 4.11) and the ng-mesh-viewer[11] (Figure 4.10) libraries provide a 2D and 3D viewer respectively. Both are implemented in the demo application in the BOT-Duplex-house repository[8].

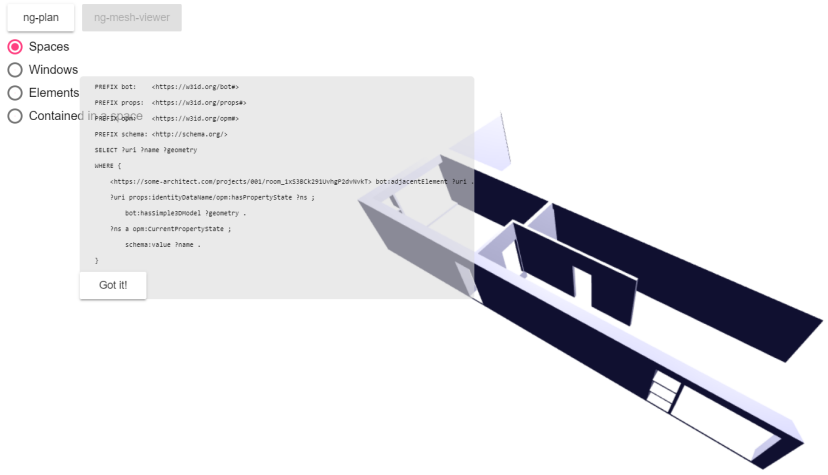


Figure 4.10: 3D model built upon request from a SPARQL Protocol and RDF Query Language (SPARQL) query. Demo available online[12].

In paper SSN, the two modules are used together with an AEC-KG consisting of BOT and SSN/SOSA compliant triples describing sensors and their observations in the context of the spaces and the building in which they are located (Figure 4.11). An online video demonstration[13] shows how the sensor observations can be queried from a User Interface (UI) showing the geometry of the model. Schneider et al. (2018) also uses the 2D plan viewer.

Enabling User Interaction

The artefacts presented in this section are focused on enabling user interaction with an AEC-KG. This includes exploration of the graph's content by traversing, communication of its content and methods for expanding it with new knowledge.

[10] Module available at: https://www.npmjs.com/package/ng-plan
[11] Module available at: https://www.npmjs.com/package/ng-mesh-viewer
[12] Web-app available at https://madsholten.github.io/BOT-Duplex-house/
[13] Video available at https://youtu.be/P_38gIvrbmg
[14] Web-app available at https://madsholten.github.io/BOT-Duplex-house/

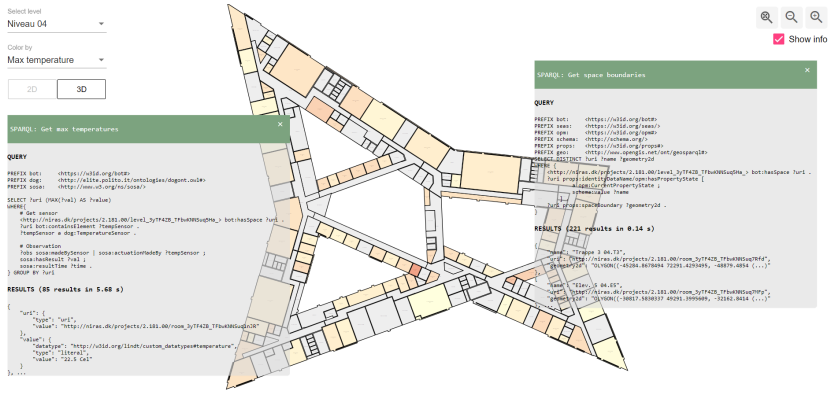


Figure 4.11: 2D plan showing maximum temperature observations over a period of three months.[14].

3D Geometry Viewer

The artefacts developed in relation to the *"Geometry as a Property"* study were not the first step in the approaches to visualising 3D geometry. The first artefact in this regard was built on AutoDesk's Forge viewer[15] and required the use of AutoDesk's cloud service for model conversion and storage. The purpose of the study was to investigate how a 3D BIM model can be queried and filtered directly from a web-browser using SPARQL. The Ng-Forge App[16] (Figure 4.12) was first demonstrated at the buildingSMART International (bSI) Summit in London in October 2017. Later, the approach was documented in paper FORGE. The dependency on the model conversion service and the fact that it was simply a filtering mechanism that filtered out elements based on International Resource Identifier (IRI) matches of a query result meant that this was a hybrid between linked data and a proprietary closed data model. It did, however, help in communicating the terminology that BOT describes.

SPARQL Visualizer

The SPARQL-visualizer[17], is a web application that lets users query an RDF dataset with SPARQL and add a description of what is contained in the dataset and what is returned by the query. The application was intended for communicating data modelling approaches, but it has also proved valuable to explore RDF datasets. When querying a dataset, the application can return results either as a graph (Figure 4.13) or in a table view (Figure 5.1).

[15]https://forge.autodesk.com/en/docs/viewer/v2/developers_guide/overview/
[16]Web-app available at https://forge-sparql.herokuapp.com/
[17]Web-app available at https://madsholten.github.io/sparql-visualizer/
[18]BOT modeling example for stairwells available at https://bit.ly/2PXUBfz

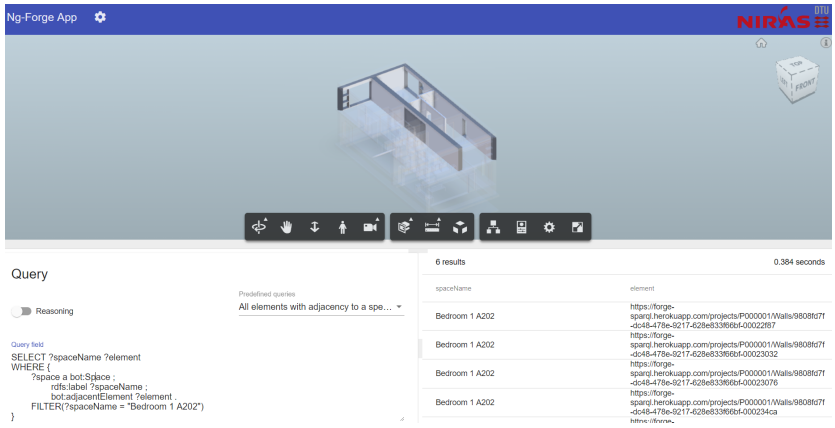


Figure 4.12: The Ng-Forge App based on the AutoDesk Forge viewer[16].

The main communication feature is that the application allows a user to save an interactive presentation like the one illustrated in Figure 4.13 for sharing with others. It us useful for communicating data modelling approaches *(1)* to people who are unfamiliar with RDF and *(2)* inside communities working with data modelling. For category *(1)* it has, for educational purposes, been useful to provide examples of how to query the graph including hints on how to further explore the content of the graph.

Paper BOT3 uses the tool to demonstrate two use-cases and a proof of concept.

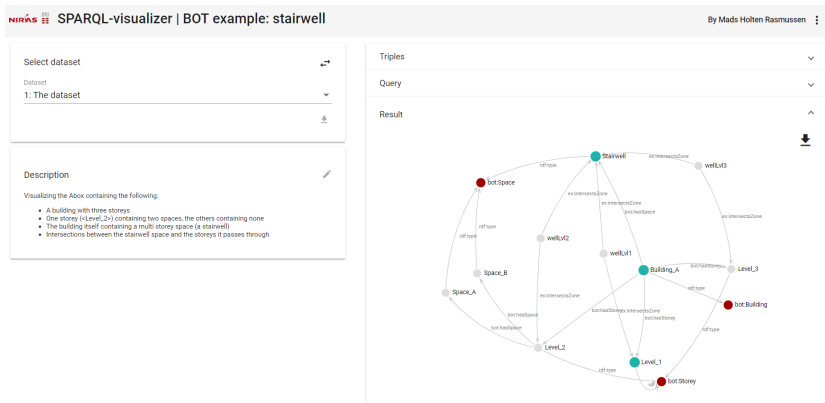


Figure 4.13: SPARQL-visualizer used to communicate a modelling approach for a stairwell spanning multiple storeys with BOT[18].

An Application for Radiator Sizing

This prototype was developed as part of the work for paper OPM2. It demonstrates how data from the architect's model can be used and further enriched with information from the ICE and HVAC engineering disciplines. Data from the architect's model is extracted with the Revit-web-server approach[9] documented in the beginning of this section. Information from the architect's Knowledge Graph (KG) is gradually extended with information relevant from an ICE engineer's perspective and later this is further extended with HVAC information.

The application demonstrates how an AEC-KG can be extended from outside a BIM authoring tool. Through the UI (Figure 4.14), the ICE engineer is able to define project specific classes with property restrictions (similar to how requirements are defined in M. H. Rasmussen et al. (2018a)), and assign these to the architect's bot:Zone and bot:Element instances. Thereby, the class property restrictions are inherited as properties by all the instances belonging to that class. The user never sees the data model behind, but gradually builds an OPM-compliant AEC-KG (Section 4.3).

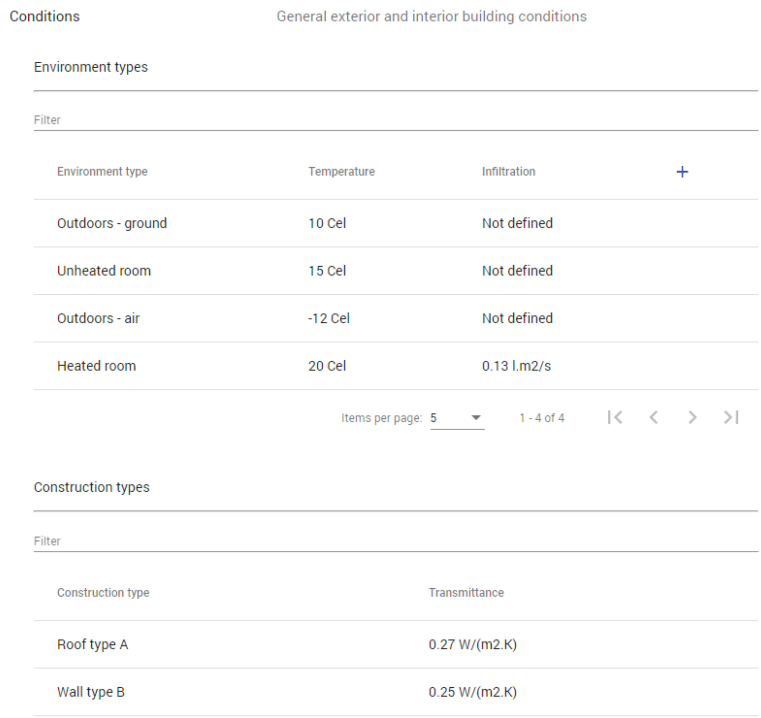


Figure 4.14: Predefined thermal environments and project specific construction types in the radiator sizing application.

The prototype is a demonstration of how the shared AEC-KG can be gradually grown as the different engineers assign new classes to the resources of interest, thereby defining their individual interpretations (i.e. technical assessments) of them.

Key Findings

The software artefacts proved to be an efficient way of conveying research findings and conceptual ideas — especially when communicating with individuals that do not have substantial knowledge in ontology engineering or data modelling in general.

During prototype development, it was occasionally discovered that the data modelling decisions were not sufficiently suitable for the intended information retrieval. Therefore, switching between ontology development and software prototyping resulted in a valuable iterative two-way feedback loop.

4.3 RT 3: Dealing with Evolving, Interdependent Design Properties

When working in iterative design processes changes occur rapidly. Being able to adapt to new changes quickly as they occur, or at least to know where they occurred, is crucial for the success of the project. The purpose of this RT is to provide terminology to model evolving, interdependent properties in construction projects.

Evolving Properties

The first paper describing this work is paper OPM1 (M. H. Rasmussen et al., 2018c), and this paper deals solely with property assignment of evolving properties. The ontology was designed to answer the following competency questions:

1. How to semantically describe a property such that its value is changeable while its historical record is maintained?
2. How to revise a property value?

3. How to delete a property while still being able to retrieve the history of it and not break all the links to derived properties that depend on it?
4. How to restore a deleted property?
5. How to retrieve the full history of how the value of a property has evolved over time?
6. How to retrieve only the latest value of a property?
7. How to simplify a complex OPM property (using states) for easier and faster querying?

Allowing properties' values to change over time is achieved by assigning at least one property state to each property. The property state classified as opm:CurrentPropertyState is the most recent, and hence the valid property state. Figure 4.15 illustrates how a property's value is changed by appending a new property state and changing the class of the one that was previously defined as the current state.

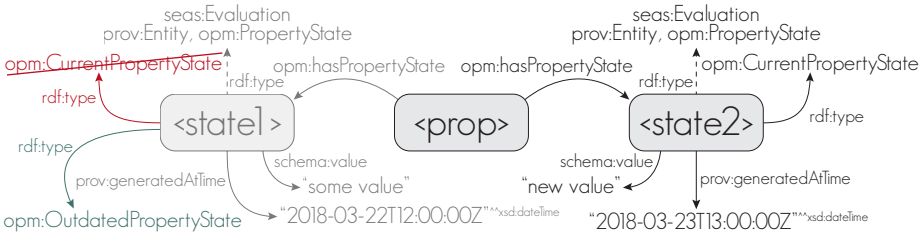


Figure 4.15: OPM modelling approach for state assignment to properties.

This modelling approach saves the full history of property states and can further hold metadata such as provenance data about who changed the property as part of what activity for what reason and so forth.

Property Reliability

Work described in paper OPM2 touches the concept of property reliability which, for example, allows engineers to work with assumed (opm:Assumed) property values in the absence of actual design values. This is common practice as it allows the project to progress until the design values are available. A reliability class can also be assigned to describe that some property is opm:Confirmed (will not be subject to change) or opm:Required. The latter was in paper REQ

used to describe requirements for spaces in a future building and comparing requirements to actual design values. The reliability classes were added in order to make the KG capable of answering the following competency questions:

1. How to describe that the value of some property is defined temporarily until the actual value is known?
2. How to document that some property has been confirmed and can hence be trusted not to change in the future?
3. How to describe that some property is derived from, and hence dependent on the value of some other property?
4. How to describe a property requirement?

Property Interdependency and Calculations

Paper OPM2 deals with properties that are derived from other properties. It further introduces the concept of an opm:Calculation to formalise the specification of the reasoning logic behind a derived property. Calculations answer the following competency questions:

1. How to associate a derived property to the properties from which it was derived?
2. How to identify that a derived property is outdated?
3. How to formally describe a calculation that can be applied to infer derived properties?
4. How to associate a derived property to the calculation or algorithm that formalises how it was derived?
5. How to check for circular dependencies in derived properties?
6. How to define the reliability of a derived property?
7. How to check which derived properties will be affected if a specific property is changed?

When working with derived properties, it is possible to model precisely what state of a particular property the result was derived from. This approach was used in the proof of concept demonstrated in paper BOT3. The demonstration,

which is illustrated in Figure 4.16, simulated a concept where three different project practitioners, an architect, an ICE engineer and an HVAC engineer work on separate sub-graphs of a federated AEC-KG. The concept is that parts of the graph are constructed from the content of the other practitioner's graphs. The content of the HVAC engineer's dataset in named graph 3 (NG3) is generated based on features in the ICE engineer's dataset in NG2 which is based on the content of features in the architect's dataset in NG1.

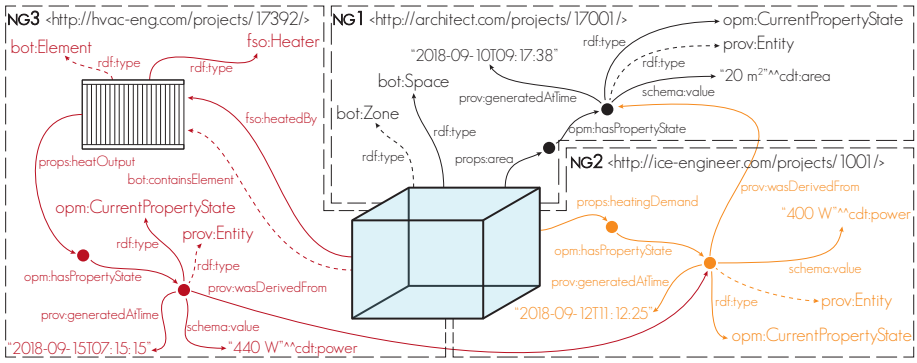


Figure 4.16: The concept of a distributed dataset from (M. Rasmussen et al., 2019b)[19].

Generating new data based on existing data is achieved using procedural inference rules formalised in SPARQL INSERT queries. The queries are constructed so that specific patterns are matched in the graph. Thereby, the arguments for each simple calculation are retrieved, and the result can be inserted as new triples. In the case illustrated in Figure 4.16, a simplified heating demand was simulated as 20 W/m^2 floor area. Thereby, the content of the ICE engineer's KG is constructed from floor areas of spaces in the architect's dataset. The HVAC engineer's KG is then constructed based on these heating demands. This is achieved by generating a query that adds a space heater with a capacity of 110 % of the heating demand for all spaces that have a heating demand. Had the information been there, a similar approach could have been used to infer the heating demand from the infiltration and transmission heat losses of the space calculated in a steady state winter condition. This approach was used in the radiator sizing application (Section 4.2), which is described in paper OPM2.

Paper OPM2 further specifies an approach to manage OPM properties and calculations using parametrised queries. A JS based tool for generating such queries: OPM-QG[20] was developed as part of this work.

[19]Video available at: https://youtu.be/61nwZ7tTcnM
[20]Library available at https://www.npmjs.com/package/opm-qg

Providing proper documentation[21] was also part of this RT.

Key Findings

In the W3C LBD CG, there have been discussions on what level of complexity to use for describing properties of a building. A takeoff from this study is that exchanging simple datatype properties, denoted L1 properties, is often sufficient. In the Revit-web-server approach[9], L1 properties were extracted from the architect's model, but internally on the server, new property states were added as L3. L3 property states provide valuable insights when used internally in one organisation, but when exchanging data with other actors, L1 is sufficient. Exchanging L3 provides higher transparency, but this transparency might not be desired.

The capabilities of this ontology were discussed with engineers at Niras and hence, it aligns with the processes and demands from practice. Maintaining control was expressed as an important factor. The system must not automatically recalculate the full dependency trees but shall instead provide the engineer with insights for decision making. Transparency is essential, and with the tractability that can be modelled with OPM, it should be capable of providing the requested insights.

[21] Documentation available at https://w3id.org/opm

Chapter 5

Discussion and Conclusion

This chapter discusses and summarises the research findings. The significance of the research is clarified by revisiting the research questions and evaluate the general contributions made to the research community and the industry.

5.1 Research Questions Revisited

In this section, the research questions are revisited. For each question it is discussed how it has been addressed with present research. The Primary Research Question accumulates all the Research Questions (RQs) in one, and therefore, the findings described below should together embrace this question.

> "*How to utilise semantic web technologies to make assertions about a building, its properties and the interdependencies between these in a manner that allows for change management and enhances transparency using widely adopted ontologies in the extent that these are available, and what additional terminology is needed?*"

RQ 1

"*What existing ontologies can be used to establish a common platform for interdisciplinary information exchange in the Architecture, Engineering and Construction (AEC) industry and is there a need for extending these?*"

Answers to RQ 1 are mainly provided through the work conducted as part of Research Task (RT) 1. The work presented in paper BOT1 (M. H. Rasmussen et al., 2017b) provided a list of existing ontologies in the AEC industry and as a result, a new, narrow-scoped core AEC ontology was proposed. This ontology should function as a mediator between practitioners of different specialist domains, and it is demonstrated with the FSO and ICE domain ontologies how it can be extended. Paper BOT3 (M. Rasmussen et al., 2019b) also demonstrates use cases and a proof of concept that extend the minimal ontology.

There are existing ontologies for building automation and energy monitoring, but for the particular field of Heating, Ventilation and Air Conditioning (HVAC) engineering, the available terminology is still limited.

RQ 2

"*How is it possible to get information from the Building Information Modelling (BIM) authoring tools used today to the new platform?*"

Some concrete tools to answer the Research Question were provided with the work conducted as part of RT 2. The revit-bot-exporter[7] initially presented in paper FORGE (M. H. Rasmussen et al., 2017a) allows BOT-compliant RDF

triples to be exported directly from the proprietary BIM authoring tool. Furthermore, it was demonstrated with paper OPM2 (M. Rasmussen et al., 2019a) how communication between the BIM authoring tool and a Knowledge Graph (KG) can be established through a RESTful web API.

The IFCtoLBD converter[1] by Jyrki Oraskari is fully compliant with the Building Topology Ontology (BOT), and constitutes another enabler for answering this Research Question.

RQ 3

"*When describing a project using the common platform, how is it possible to capture the design information as well as interdependencies while allowing it to change and evolve?*"

RT 3 provides some answers to RQ 3. The work conducted as part of this task provided terminology to describe evolving properties with paper OPM1 (M. H. Rasmussen et al., 2018c) and more was added with papers REQ (M. H. Rasmussen et al., 2018a) and OPM2 (M. Rasmussen et al., 2019a). Paper REQ demonstrated a use-case where property requirements were compared to actual design properties and a JavaScript (JS)-based API (Ontology for Property Management (OPM)-QG) was introduced in paper OPM2 to simplify the communication with an OPM-compliant AEC Knowledge Graph (AEC-KG).

Working with interdependent properties is a complex task, and there is probably not one single solution. OPM provides the necessary terminology to describe evolving properties and includes some modelling approaches for derived properties. How to infer the derived properties is not restricted by OPM, and therefore it is quite flexible. With paper OPM2 a RESTful web API implementing the OPM-QG[4.3] was proposed.

[1] Repository available at https://github.com/w3c-lbd-cg/IFCtoLBD

5.2 Implications

This section summarises the implications to academia and the industry and discusses these in comparison with other efforts and approaches. A future outlook is presented to outline potential future research projects, and finally, some concluding remarks are given.

Contributions to Academia

The main contributions to academia are the two ontologies, BOT and OPM. This includes the considerations on what problems they are supposed to solve, how they are integrated with existing work, and how they can be adopted by the AEC industry.

BOT
BOT was proposed as a *central AEC ontology that allows for domain-specific extensions* and a secondary contribution of this research has been to demonstrate how it can be used in combination with existing ontologies. This was demonstrated in relation to sensor observations and assignment of client space requirements. The integration with legacy BIM software also constitutes a secondary contribution of this work.

BOT has been reused in various research projects in different sub-domains working with buildings. The fact that it is being adopted by the community indicates that it does fill a gap. Bus et al. (2018) suggests using it for compliance checking. The Building Automation and Control System (BACS) ontology by Terkaj et al. (2017) uses the building topology concepts in the context of Building Management Systems (BMSs), Qiu et al. (2018) for capturing human experiences of indoor environments, and the Energy Efficiency Prediction Semantic Assistant (EEPSA) extends BOT in the context of energy efficiency. IFCtoLBD[1], which converts an Industry Foundation Classes (IFC) file to BOT-compliant RDF triples is another example of its use. BOT is further mentioned in the Linked Data chapter of Pauwels et al. (2018a).

OPM
OPM was proposed as an ontology for managing evolving, interdependent design properties with varying reliability. A secondary research contribution of this work is the specification of a standardised way to generate parametric queries

for managing an OPM-compliant AEC-KG. To enhance ease of use, the latter was implemented in a software library that is publicly available[2].

Existing research is mainly focused on generating an RDF graph from a finalised BIM model and using linked data to extend with external knowledge. The focus of this research has been on the design stages, and therefore, the approach has been to establish the graph from scratch. Connections to legacy software were generated to generate baseline datasets, but the ontology design encourages a fresh start. From the perspective of an HVAC engineer, it has been demonstrated how the HVAC design can be designed based on knowledge from an architect's model, and it is possible with OPM to use this knowledge in an insightful way.

Dissemination

Semantic Web and Linked Data integration with BIM are research fields in growth. At the European Conference on Product and Process Modelling 2018 in Copenhagen, Dr. Arto Kiviniemi, who has been involved in BIM since 1996[3], mentioned that he had never seen so many BIM research projects on one shared topic. There has been great feedback and valuable follow up discussions on presentations given at conferences. Especially within the buildingSMART International (bSI) Summits in London 2017 and Paris 2018.

SPARQL-visualizer has been well received by the research community. In the W3C LBD Community Group (W3C LBD CG), it has been used by Mathias Bonduel to communicate different approaches to property assignment[4]. It has further been used in education at both Technical University of Denmark (DTU), Eidgenössische Technische Hochschule Zürich (ETH), Ghent University (UGent) and Aalborg University Denmark (AAU) and has been used for live demonstrations at several conference presentations by various researchers. For education, it has in particular been valuable that it allows the exploration of a dataset. In the context of BOT, the tool is used to communicate the content of the ontology[5] and to demonstrate modelling approaches[18].

Two lectures on BOT and the possibilities enabled with data modelling in graphs using semantic web technologies, were conducted in the MSc course 11034 Advanced BIM at DTU. The students were mainly from the Architectural Engineering programme and had little to no programming experience. Nonetheless, they left an impression of having understood the technologies, and the application of them. This was reflected in the lecture exercise where they had to model

[2]https://www.npmjs.com/package/opm-qg
[3]From biography: https://www.liverpool.ac.uk/architecture/staff/arto-kiviniemi/
[4]See discussion at: https://github.com/w3c-lbd-cg/props/issues/2
[5]BOT-introduction available at https://bit.ly/2Ql5XtD

a KG of the auditorium from their professional perspective. Knowledge representations in the domains of statics, indoor climate, acoustics, fire evacuation and other building related disciplines were submitted. The students afterwards used SPARQL-visualiser to query the KG.

Contributions to the Industry

The use of the terminology provided with BOT and OPM is not limited to academia. The developed tools demonstrated that the technology is mature, especially considering combinations with existing commercial triplestores. Neither OPM or BOT are standardised by any standardisation organ, but the schematics that they provide can still be used internally in a company. Both ontologies were brought to the W3C LBD CG, and efforts in providing proper online documentation[6] was conducted as part of the research. The technologies were also demonstrated at industry conferences in order to increase the awareness.

AEC Practitioners as Software Developers
The construction industry is heavily influenced by local culture and tradition that varies significantly from country to country. Even the different companies that operate in the industry have individual approaches to solving the same problem. Therefore, it is hard to buy off the shelf software that meets the demand. The technologies presented in this work can be used to capture specific project information in a universally accessible way, and it was further demonstrated how this information can be accessed, processed and evolved through a proof of concept tool for radiator sizing. Engineers already develop custom tools in advanced spreadsheets. Integrating these tools with the AEC-KGs could take them to the next level by providing them with access to the single project truth represented in a (federated) data model. This connection is already possible since communication with the AEC-KG is enabled with the HyperText Transfer Protocol. Also, as it was demonstrated, in some of the software artefacts developed as part of the research, it is not too resource-intensive to develop custom applications for different design tasks. Especially not when new engineer graduates already possess some programming knowledge. This option is probably reserved for medium to large size companies in the near future, but as the technology evolves, we already see an increase in accessibility.

[6]BOT: https://w3id.org/bot, OPM: https://w3id.org/opm

The software artefacts opm-qg[7], ng-plan[8] and ng-mesh-viewer[9] are all openly available for download from Node.js Package Manager (NPM) and the source codes are on GitHub. These are building blocks that can be used to build custom software tools for processing an AEC-KG. The BOT-Duplex-house[10] demonstrates a minimal application that uses ng-plan and ng-mesh-viewer and communicates with a small AEC-KG of the common BIM model, the Duplex Appartment[11]. The source code is openly available on GitHub[12] and it is built on open source JS libraries such as Angular, ThreeJS and rdfstore[13].

Freeing the Data

An immediate achievement by parsing proprietary BIM models into RDF graphs is that data can be made quickly available to the project participants. Even a simple dashboard-like application can provide valuable insights that will instantly bring value to a construction project. When demonstrating to an engineer that all rooms, room numbers and areas could be retrieved in a table format within a second (Figure 5.1), his response was: *"are you aware that it currently takes me half an hour to open that model in the BIM authoring tool to retrieve such data?"* This comment exposes the benefits in extracting the data from the tools. The data store does not necessarily need to be a triplestore but could also be achieved using conventional database solutions. In an industry where practitioners generally play by the rules laid out by the software at hand, it is an eye-opener to see such a demonstration.

Implementation Considerations

Niras managers and employees have reflected on how a transition to using AEC-KGs could happen. The vision is to develop small applications suited for particular sub-tasks performed regularly by the engineers, and these applications should integrate with the AEC-KG of each project. To realise this, Niras needs to educate employees at various levels, and the discussions have revolved about education at four levels. An employee at the entry level, *Level 1*, can use the tools. *Level 2* includes employees who understand the tools and can make feature requests. At *Level 3*, the employee can build front-end applications that communicate with the AEC-KGs through a backend (REST API), and the highest level, *Level 4*, includes the few employees who understand the capability and design of the KG in detail and can maintain and extend the backend. Most of

[7] https://www.npmjs.com/package/opm-qg
[8] https://www.npmjs.com/package/ng-plan
[9] https://www.npmjs.com/package/ng-mesh-viewer
[10] https://madsholten.github.io/BOT-Duplex-house/
[11] https://www.nibs.org/page/bsa_commonbimfiles#project1
[12] https://github.com/MadsHolten/BOT-Duplex-house
[13] http://npmjs.com/package/rdfstore

2639 results in 0.492 seconds Show datatypes

s	name	area
http://172.16.10.122:8080/projects/1010/rooms_1hsZsuUnb3vgd2Lu1_eaMJ	Forrum, toiletter 80.01.041	3.70
http://172.16.10.122:8080/projects/1010/rooms_1hsZsuUnb3vgd2Lu1_eaMS	Garderobe 10.03.042	6.33
http://172.16.10.122:8080/projects/1010/rooms_1hsZsuUnb3vgd2Lu1_eaPN	Garderobe 10.03.043	6.53
http://172.16.10.122:8080/projects/1010/rooms_1hsZsuUnb3vgd2Lu1_eaPG	Forrum, toiletter 80.01.046	3.82
http://172.16.10.122:8080/projects/1010/rooms_1hsZsuUnb3vgd2Lu1_eaPR	Toilet 01.01.050	3.42
http://172.16.10.122:8080/projects/1010/rooms_1hsZsuUnb3vgd2Lu1_eaOa	Toilet 01.01.049	2.94
http://172.16.10.122:8080/projects/1010/rooms_1hsZsuUnb3vgd2Lu1_eaOe	Forrum, toiletter 80.01.045	6.42

Items per page: 10 11 - 20 of 2639

Figure 5.1: Room areas queried using SPARQL Protocol and RDF Query Language (SPARQL)-visualizer.

these employees up to level 3 are primarily engaged in regular projects where they apply their particular engineering knowledge, and contributing to the tools is a secondary task. This way, the development becomes an integrated part of the consultancy, and this is the main goal. Some of the engineers in the company already possess programming skills at varying levels from visual programming in tools like Grasshopper for Rhino and Dynamo for Revit (Figure 5.2) to C# programming and intermediate understanding of Revit's API. These engineers should be educated to level two and higher.

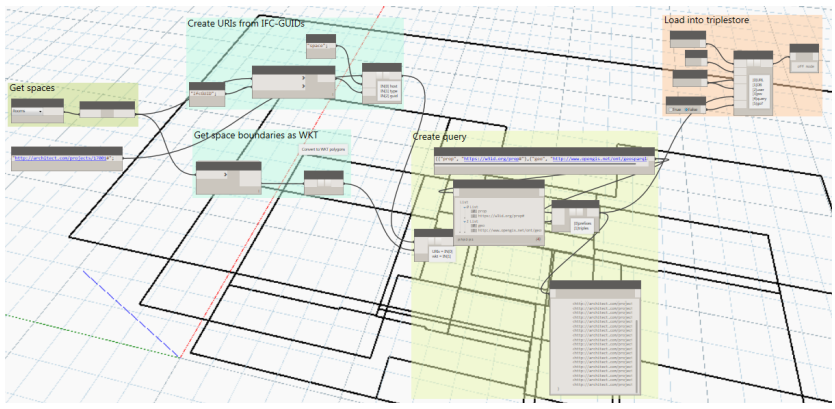


Figure 5.2: Communication between Revit and an AEC-KG through Dynamo.

Maturity of the Technology
There are several commercial triplestore solutions on the market, and Modoni et al. (2014) concluded that they meet the demand of the industry in regard to security, speed and scalability. Since then, new promising products like Ontotext's GraphDB and Amazon's Neptune have arrived, and it seems like the technology is gaining momentum. Gartner's 2018 Hype cycle for emerging technologies[14] lists KGs as an 'innovation trigger' that is believed to reach maturity within the next five to 10 years.

The great vision of a semantic web as described by Berners-Lee et al. (2001) has still not *"changed the world"*, as the authors envisioned it would. The semantic web vision was about a decentralised web of structured collections of information that could be interpreted and interacted with by computer agents. Instead, we have seen the web becoming increasingly centralised where big players like Google and Facebook grow and leave little room for competitors to thrive. The vision is not dead, however, and proponents like Sir Tim Berners-Lee, Ruben Verborgh and the Decentralized Semantic Web (DeSemWeb) community are still working on realising it. One of the realisations by this community is that it must not be too academic, and it is crucial for its propagation that it is easy to use for frontend developers (Verborgh, 2018). One result of this realisation is Solid[15] and the whole ecosystem around it that makes it easy to integrate with popular JS frameworks like React and Angular. The development of JSON Linked Data (JSON-LD) (Sporny et al., 2014) is also an important accomplishment in this regard since the JavaScript Object Notation (JSON) data structure is what web developers are used to.

Adoption in Other Industries
The semantic web technology stack (Figure 2.7) provides a toolset for describing knowledge, and the fact that RDF is well-suited for describing domain knowledge is supported by its adoption in other industries. The Linked Open Data (LOD) cloud[16] (Figure 5.3) is a visualisation of publicly available linked data, and it contained 1,231 datasets and 16,132 links in June 2018. The cloud is divided into domains, and Life Science is the dominating one. In this field, semantic web technologies are widely used to integrate datasets in fields such as neurosciences, cancer research, and drug discovery. The Gene Ontology[17] by Ashburner et al. (2000) and Bio2RDF[18] by Belleau et al. (2008) are examples of initiatives that have been well adopted by this community.

[14] https://www.gartner.com/en/newsroom/press-releases/2018-08-20-gartner-identifies-five-emerging-technology-trends-that-will-blur-the-lines-between-human-and-machine
[15] https://github.com/solid/solid
[16] https://lod-cloud.net/
[17] http://www.geneontology.org/
[18] http://bio2rdf.org/

Another example application more related to the construction industry is the European Road Object Type Library (OTL) for asset information management for European Roads life-cycle management (Luiten et al., 2018). This is a development of the INTERLINK project[19], which is partly funded by the Conference of European Directors of Roads (CEDR). Although the European Road OTL is in its infancy, a road map for a bottom-up adoption by National Road Authoritiess (NRAs) is laid out. This approach has also been discussed in relation to the implementation at Niras. It will be necessary to develop domain specific extensions for BOT as it was demonstrated in Chapter 4 with Flow System Ontology (FSO) and Indoor Climate and Energy (ICE). Following a bottom-up approach like the one suggested by Luiten et al. (2018), the next step would be to make the ontologies available to the national authorities, and subsequently to a global standardisation organ like buildingSMART.

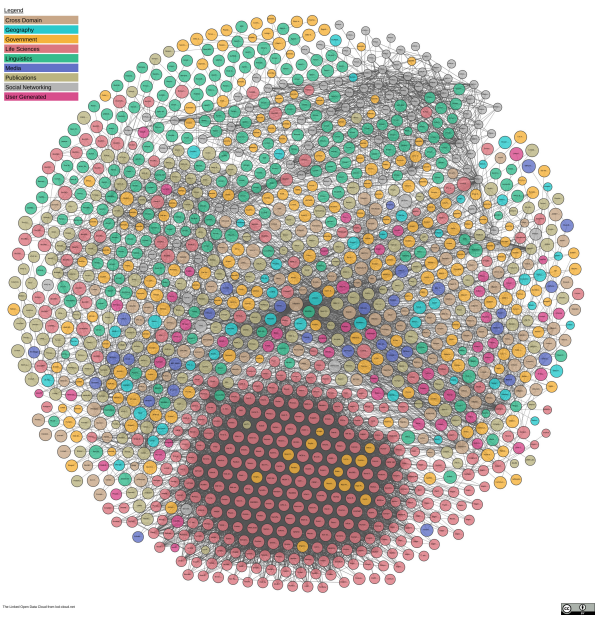


Figure 5.3: The LOD Cloud, Accessed January 2019.

Alternative Approaches

Property Graphs (PGs) are an alternative to RDF for modelling KGs, and the most widely adopted NoSQL graph database according to DB-engines[20] is Neo4J which is based on Labeled Property Graphs (LPGs). In LPGs edges and

[19] https://www.roadotl.eu/
[20] https://db-engines.com/en/ranking/graph+dbms

nodes act as containers that can hold properties internally. This architecture entails that edges can have properties (which requires an intermediate node in RDF), and since only relationships between objects are defined as edges, the data model is more compact. LPGs and PGs do not support the open formats and publication standards of the semantic web stack. They do not provide reasoning capabilities and do not use International Resource Identifiers (IRIs) to identify resources. Therefore, while they might be a good solution for a closed dataset organised internally in a company, they do not bring the integration benefits provided with RDF and the semantic web stack. Triplestore solutions like Stardog have lately added support for mapping so-called virtual graphs to relational databases and NoSQL databases like MongoDB, and with this addition, SPARQL becomes a quite powerful language for querying heterogeneous datasets in a semantically meaningful way.

With AutoDesk Forge, it is also possible to develop tools that consume information from native and open BIM models, and integrate it in customised ways. This solution does, however, require all the data to reside on AutoDesk's servers, and therefore it implies a significant reliance. Questions like *"What happens with our data if the company goes bankrupt?"* and *"What if we wish to transfer our data to another software tool?"* are relevant to ask. Further, with a situation where the majority of AEC practitioners have based their entire toolset on one vendor, a monopoly situation emerges allowing this vendor to increase the prices of their products and practically stop the development. The vision behind Forge is similar to the one presented with this research as it emphasises the benefit in putting the data model in the centre, thereby separating it from the software solutions. The major difference, however, is that Forge's data model is proprietary. With a data model described using open ontologies it is possible to switch from one vendor to another with the dataset intact, and this significantly reduces the vulnerability of the AEC practitioners. For the software vendors, however, there is an incentive in making the customers' reliant on their products. This situation probably means that open standards need to be enforced by the AEC practitioners. Alternatively, as it is already seen with initiatives like Grasshopper for Rhino, Dynamo for Revit and Speckle for communication between these, the development of open source tools must be driven by the AEC community itself.

Future Outlook

This research opens an extensive list of open challenges for future research projects. In the following subsections, some of these are briefly listed.

A Distributed AEC Knowledge Graph
In paper BOT3 (M. Rasmussen et al., 2019b), the foundation for a decentralised web-based AEC industry is demonstrated. Future research should examine this approach in a practical environment with running tools, processes, and a real ongoing construction project. The data model could be put in a single repository – a Common Data Environment (CDE). It would, however, be even more interesting to investigate the potential of a truly distributed model using technologies such as Linked Data Fragments by Verborgh et al. (2016).

The Human Factor
As it was mentioned in the Research Scope in Chapter 1, user interaction, User Experience (UX) design and related studies are important factors for the success of an Information Technology (IT) system. The literature reviewed as part of Chapter 2 also backed up the importance of the perceived ease of use. Bhatt (2000) argues that only 10 % of the effort in Knowledge Management (KM) is constituted by technology, whereas 70 % depends on people as illustrated in the Venn diagram in Figure 5.4. Therefore, a natural next step is to research how people can interact with an AEC-KG.

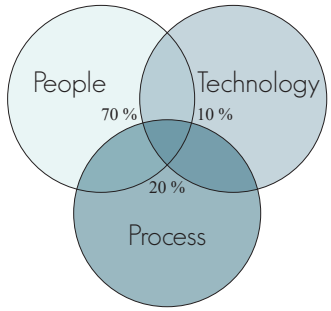


Figure 5.4: KM components by Bhatt (2000).

Collaborative Perspectives
van Berlo and Krijnen (2014) describes the use of BIM Collaboration Format (BCF) in a server based workflow. BCF supports collaboration in a BIM environment by allowing any user to create a task and describe this task in relation

to the objects which it concerns. Treldal et al. (2016) suggests that this scope is expanded to also include contract management methodology and other process-related activities. OPM on the other hand deals with interdependent properties and provides methods to assess the full history of a property's evolution over time. It would be interesting to investigate how BCF and OPM could be used to relate the state changes of the involved properties to the specific task. The use of BCF is currently practically limited to objects that have geometrical representations in a BIM model. In the proof of concept demonstration in paper BOT3 (M. Rasmussen et al., 2019b), heat sources are defined in the data model before they are modelled geometrically. This allows for BIM objects to exist at an abstract level, and it would be interesting to investigate how to include such abstract objects in BCF tasks – potentially in combination with the process-management approach outlined by Treldal et al. (2016).

Smart Contracts
With OPM, property changes are assigned as new property states and the reliability classes can describe a property as assumed, required or confirmed. OPM does, however, not deal with responsibility or breach of contract in the case where a confirmed property is changed or a requirement is not fulfilled. The legal aspects of these situations could be addressed if the property states were put in a distributed ledger instead of residing on the servers of the different stakeholders. Therefore, it would be interesting to look into the use of OPM in a blockchain architecture.

Knowledge Capture from heterogeneous formats
Natural Language Processing (NLP) technologies could enable extraction of assertions from prose text such as meeting memos, building element descriptions and e-mails. Thereby, the knowledge workers would not have to change their practice significantly, and there would be no need to manually type information into forms. It is possible with OPM to describe why a certain change was made, and this kind of information is valuable for later assessment of what initial decisions led to the resulting solution. This, however, requires the information to be captured, and with automatic knowledge extraction from prose text, the likelihood of the information being captured is higher. An integration between NLP technologies and AEC-KGs is, therefore, a future research topic of interest.

Design Properties
OPM addresses challenges related to modelling of complex properties and their interdependencies, but the actual properties are not part of this work. The

W3C LBD CG is addressing this challenge with the PROPS[21] ontology. It might also be worthwile to investigate an architecture similar to the one proposed with BOT for properties. International Organization for Standardization (ISO) standards, Eurocodes and national annexes all define specific physical properties, and things such as measurement method and application purpose is very important for interpreting a property. A heat transmission area varies from country to country as it was demonstrated with the ICE ontology in Section 4.1.

buildingSMART Data Dictionary (bSDD) is one approach to dealing with properties, and it addresses the demand for being able to define a property to be commonly shared and understood. It achieves this by generating unique identifiers for different properties and allowing for attaching labels in different languages. The reality, however, is that many properties are not defined, some are defined redundantly and very few languages are supported.

As part of this work, it was briefly investigated how Wikidata[22] could be used for dynamically generating an ontology for physical properties based on Wikipedia. Wikidata provides meaning (e.g. *area* is a scalar physical quantity abbreviated with symbol A with International System of Quantities (ISQ) dimension L^2.) and labels in different languages. For example, there are 148 labels available for *area* compared to six in bSDD[23]. The prototype[24] developed as part of this work is described in Appendix A. An important outstanding challenge that was not addressed is responsibility and trust when generating an ontology dynamically from an encyclopedia edited by the public. Also, if it should constitute the core like BOT, it should be demonstrated how the generic Wikidata concepts are extended with code-specific terms. Proper guidelines for how the standardisation organs should structure and publish their vocabularies in RDF are also needed from a dissemination perspective.

Concluding Remarks

This work uncovered that a "digital infrastructure" that will support the engineer in the design and planning of building services already exists in the form of the internet and all the web standards that are continuously being developed by the World Wide Web Consortium (W3C). From a cloud computing perspective, Infrastructure as a Service (IaaS) only includes hardware, but with this work, the infrastructure was thought of as a term that also covers software. With

[21] Repository available at: https://github.com/w3c-lbd-cg/props

[22] Wikidatadefinitionofarea:https://www.wikidata.org/wiki/Q11500

[23] bSDD definition of area: http://bsdd.buildingsmart.org/#concept/details/0BiMY5Q4P8heAm8_4CQYUF

[24] Dynamic props ontology: https://objprops-gen.herokuapp.com/id/area

semantic web technologies and the proposed ontologies: Building Topology Ontology (BOT) and Ontology for Property Management (OPM), the foundation for establishing an infrastructure for information capture and retrieval is established. None of these are restricted to building installations, but can be used in a wider perspective. The scope of BOT is buildings, and it is intended to be extended for specific domains like installations, Facility Management (FM), life Cycle Assessment (LCA), Building Performance Simulation (BPS), acoustics and so forth, which was demonstrated with the Flow System Ontology (FSO) and Indoor Climate and Energy (ICE) extensions. OPM is designed to capture any complex design property and can also be applied in a wide range of domains that deal with properties that change over time and have interdependencies.

The hypothesis is that enhanced transparency will enhance the interoperability between the different stakeholders collaborating on a project. This is conflicting with the current practice where transparency means that everything can be found and used in case of dispute. In Chapter 1, it was briefly mentioned that new contracts must be composed in order to avoid this contradiction, and Integrated Project Delivery (IPD) is a contract that accommodates this problem. There are, however, also other reasons why full transparency might not be desirable. For example, it might not be for the better of the project to communicate all design changes in a creative phase where everything changes rapidly, and therefore there will be a demand for working in both private and open environments. When using OPM to manage evolving properties it might therefore also be desired to track all internal changes and interdependencies in private and only exchange the most recent property state with other practitioners. As such, a clear interface between the design tasks or "systems" as denoted by Eppinger et al. (1989) is maintained, while it is still possible to establish dynamic links to the information.

With the contributions of this work alone, it is not possible to assert that the observed problems with knowledge capture have been solved. The two ontologies together provide the necessary terminology to make meaningful assertions about a construction project that will enable knowledge retrieval. This, however, presupposes that the knowledge has been captured by the knowledge workers. In order for this to happen, accessible, meaningful interfaces to the Knowledge Graph are an absolute necessity.

In order for the proposed ontologies to be adopted by the industry, it is important that they are backed up by a standardisation organisation such as buildingSMART or W3C. The ontologies serve a similar purpose as schemata, but since the Resource Description Framework (RDF) builds on an Open World Assumption, a dataset is described with terminology from multiple sources. This architecture opens new opportunities for liaison between standardisation bodies that should be embraced. BOT demonstrates how buildingSMART could struc-

ture the successor of Industry Foundation Classes (IFC) in a modular fashion that reuses work by other organisations – NASA has, for example, developed an ontology for Quantities, Units, Dimensions, and Data Types (QUDT), that could be reused for describing physical units. In such a setup, the role of buildingSMART is not to provide a conclusive schema for Building Information Modelling (BIM) but to provide a solid core and documentation that allows other standardisation bodies and product manufacturers to align.

Chapter 6

Papers

This chapter includes all the papers presented in the List of Papers. The papers are ordered based on publication date and those that were still under review at the time of submission are ordered based on submission date.

6.1 BOT1: Proposing a Central AEC Ontology That Allows for Domain-Specific Extensions

This paper investigates existing ontologies in the scope of buildings and finds that they repetitively describe the main components of a building (building, storeys, spaces). It suggests the Building Topology Ontology (Building Topology Ontology (BOT)) as a minimal, extendable ontology only covering the main concepts of a building and demonstrates linking approaches for domain specific extensions.

Rasmussen M.H., Karlshøj J., Hviid C.A. and Pauwels P. (2017). "Proposal of a central AEC ontology for domain specific extensions" In: *LC3 2017: Volume I – Proceedings of the Joint Conference on Computing in Construction (JC3)*, July 4-7, 2017, Heraklion, Greece, pp. 237-244. DOI: https://doi.org/10.24928/JC3-2017/0153.

PROPOSING A CENTRAL AEC ONTOLOGY THAT ALLOWS FOR DOMAIN SPECIFIC EXTENSIONS

Mads Holten Rasmussen[1], Pieter Pauwels[2], Christian Anker Hviid[3] and Jan Karlshøj[4]

Abstract: In the last years, several ontologies focused on structuring domain specific information within the scope of Architecture, Engineering and Construction (AEC) have emerged. Several of these individual ontologies redefine core concepts of a building already specified in the publicly available ontology version of the ISO standardised Industry Foundation Classes (IFC) schema, thereby violating the W3C best practice rule of minimum redundancy. The voluminous IFC schema with origins in a closed world assumption is likewise violating this rule by redefining concepts about time, location, units etc. already available from other sources, and it is furthermore violating the rule of keeping ontologies simple for easy maintenance. Based on all the available ontologies, we propose a simple Building Topology Ontology (BOT) only covering the core concepts of a building, and three methods for extending this with domain specific ontologies. This approach makes it (1) possible to work with a limited set of core building classes, and (2) extend those as needed towards specific domain ontologies that are in hands of business professionals or domain-specific standardisation bodies, such as the European Telecommunications Standards Institute (ETSI), buildingSMART, the Open Geospatial Consortium (OGC), and so forth.

Keywords: Linked Data, Building Information Modelling, Web of Data, Building Topology Ontology.

1 INTRODUCTION

Design and planning of buildings is a complex and iterative process involving interaction between several stakeholders. A substantial part of the complexity originates from the amount of information to be handled, the number of parties possessing and consuming the information, and the lack of tools that can combine the distributed information, structure the data and present it in a way that is beneficial to the individual building planner (Kiviniemi, 2005). Building Information Modelling (BIM) has been introduced in the industry to overcome this problem. With this introduction, the entire industry is now shifting from an initial BIM stage with single-disciplinary silo models (maturity level 1), to a data-centric BIM stage (maturity level 3) with a purposefully interlinked network of data (Succar, 2009).

[1] PhD student, Department of Civil Engineering, Technical University of Denmark (DTU), Lyngby, DK, mhoras@byg.dtu.dk
[2] Assistant Professor, Department of Architecture and Urban Planning, Ghent University, BE pipauwel.Pauwels@UGent.be
[3] Assistant Professor, Department of Civil Engineering, Technical University of Denmark (DTU), Lyngby, DK cah@byg.dtu.dk
[4] Associate Professor, Head of Section, Department of Civil Engineering, Technical University of Denmark (DTU), Lyngby, DK jak@byg.dtu.dk

Linked Data technologies allow for this shareable data model to be not one model, but several smaller interlinked models together forming a complete data representation. The partitioning into sub-disciplinary data models solves the issue with several stakeholders needing to specify capabilities of a single building object. It makes it furthermore easier to specify and respect legal responsibilities and intellectual property rights (IPR). Another advantage of linked data is that a vast amount of data is already available online, making it possible to integrate the building models with linked open data such as weather information, physical capabilities of materials, product datasheets, sensor readings, unit conversion and Geographic Information System (GIS).

This paper is situated in this context of interlinking building data using web technologies. Several ontologies have now been specified and proposed, aiming to capture specific knowledge related to buildings (geographical location, sensor data, domotics, construction data, and so forth). The aim of this paper is to review these publicly available ontologies and to clarify how they overlap each other (section 2). In section 3 a simple BOT ontology is suggested, to represent the overlapping building data found in section 2, and in section 4, three different approaches to extending BOT by existing ontologies (GIS and appliances in this article), by linking datasets (each following an ontology) are examined, hence clearly showing how data and ontology reuse can be done for the AEC industry. The approaches are (1) through an upper ontology (2) by defining equivalence classes and (3) by using typed links, and the latter comes out as the preferred approach.

2 Review of Existing Building Ontologies

Recent research projects suggest using web of data technologies for managing building data, thereby making it possible to link interconnected information between disciplines in the AEC domain. Figure 1 shows examples of AEC-datasets that have inter-dependencies with other AEC-datasets. Common for them is that they all consume data about the single components of the building in scope. W3C Best Practices for data on the web (Lóscio *et al.*, 2016) states that in order to increase interoperability and reduce redundancies, reuse of vocabularies should be attempted, but in research projects the same concepts of a building are contradictory redefined repeatedly.

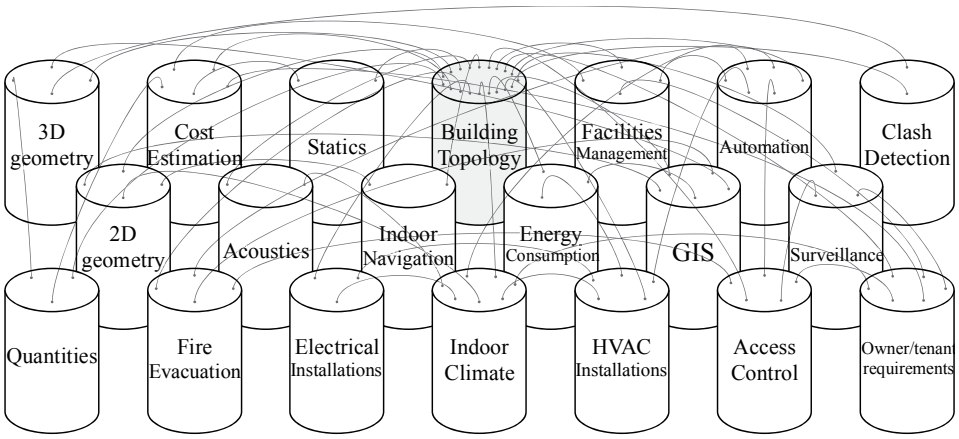


Figure 1: Example of interdependencies between datasets.

In this section a selection of ontologies are examined, specifically aiming to retrieve concepts and classes about buildings. We distinguish between generic ontologies (broad scope) and domain-specific ontologies (narrow scope). The considered ontologies are listed in Table 1 where it is also stated which domain they seek to cover, whether they are accessible online and the name of the building related classes if these are part of the ontology. The offline ontologies are mentioned in the literature, but as they are not publicly available, they are of very limited use in a linked data environment.

2.1 Generic ontologies

Generic ontologies encompass a wide span of concepts to represent knowledge about the world as we know it. One of the largest knowledge bases on the web of data is *DBpedia* which is a community effort to extract structured information from Wikipedia and make it available on the web (Auer *et al.*, 2007). The DBpedia ontology contains classes to describe a *place* and a *building*, but the main purpose of the schema is to define simple facts such as floor area, building type and number of floors, and hence no further details of the building components and their relationships and context to the building can be described. The same accounts for *schema.org*, a joint effort between Bing, Google and Yahoo (Ronallo, 2012), which can only hold information about where a building is located and mainly sees it as a location of a certain service of importance to people. Another widely used example is the *Suggested Upper Merged Ontology* (*SUMO*), a so-called upper ontology or superstructure for information that is true in lots of domains (Pease, no date). Concepts already defined in *SUMO* have been diligently used in some of the domain ontologies described in section 2.2.

The most general schema for describing buildings is IFC (Liebich and Wix, 1999). Several research projects have dealt with the conversion of this schema into an ontology. Pauwels and Terkaj (2016) suggested the *ifcOWL*, and T. M. De Farias, Roxin and Nicolle (2015) suggested the *ifcWOD*, which further simplifies querying. The *ifcOWL* ontology contains 1313 classes and 1580 object properties and besides from defining information about a building, it can also hold data related to time planning, costs, physical units, etc. An ongoing initiative aims at extending the schema to also hold information about roads (Lee and Kim, 2011) and bridges (Yabuki *et al.*, 2006). For the purpose of a simple building representation one may argue that the IFC schema is too extensive, and for the same reason (Pauwels and Roxin, 2016) suggested *SimpleBIM*, which cuts away elements like geometric data and intermediate relation instances between objects. *SimpleBIM* is not an individual ontology, but rather an approach to post-process an *ifcOWL*-compliant RDF graph with a simplified building representation.

A core ontology initiated by the Dutch civil engineering industry called *Cbim* consists of a minimal schema containing class definitions for objects, their properties and their relations (van Nederveen, Beheshti and Willems, 2010). The ontology functions as an upper ontology with the capability of checking that relationship constraints are not violated, and the minimal extent of it means that it should be extended depending on demand. Another general building ontology is the *BIM Shared Ontology* (*BIMSO*), which can likewise serve as a core for domain specific ontologies. It uses the UNIFORMAT II classification system for declaring the classes and is therefore not a minimal schema as *Cbim*, but rather an alternative to IFC (Karshenas and Niknam, 2013).

2.2 Domain-specific ontologies

Domain-specific ontologies limit to very specific domains, in contrast to the earlier mentioned generic ontologies. In a sense, building ontologies such as *ifcOWL*, *BIMSO*, and *Cbim* could be considered ontologies specific to the building domain as well, but they cover a wider domain as they point to ontologies outside the building domain as well (units, geometry, location, etc.). This also happens with, for example, *Geographical Information Systems* (*GIS*) and smart cities ontologies, which typically have a geographical basis that heavily points to buildings. *cityGML*, *DAREED* and *SEMANCO* are examples of ontologies in the smart city domain that include the concept of a building. *cityGML* and *SEMANCO* are the only ones that were available online at the time of writing. *cityGML* has its own definition of a building. *SEMANCO* includes *SUMO* for describing overall concepts like *Building* and *Floor* but defines its own classes for what is not contained in *SUMO* (Madrazo, Sicilia and Gamboa, 2012).

Table 1: Building topology information in domain specific ontologies.

Ontology	Domain	Online	Building	Storey	Room	Elements
DBpedia	World	X	Building	-	-	-
Schema.org	World	X	Civic-Structure	-	-	-
ifcOWL	AEC	X	IfcBuilding	IfcBuildingStorey	IfcSpace	IfcElement
ifcWOD	AEC	-	X	X	X	X
BIMSO	AEC	-	?	?	?	?
CBIM	AEC	X	-	-	-	Object
SEMANCO	Smart cities	X	SUMO: Building	SUMO: Floor	Space	subclasses
cityGML	Smart cities	X	Building	-	-	-
DAREED	Smart cities	-	?	?	?	?
SEAS	Systems	X	Building	Building-Storey	Room	Building-Space-Connection
SAREF	Smart homes	X	-	-	Building-Space	Building-Object
DogOnt	Smart homes	X	Building	Storey	Room	Building-Thing
ThinkHome	Smart homes	X	Building	Building-Storey	Space	Opening, Equipment

With the prevalence of Internet of Things (IoT) technologies, sensors, actuators, meters and home appliances need a common way of communicating with a building to create

so-called *Smart Homes. Smart Appliances REFerence* (*SAREF*) is an example of an ontology in this domain which includes the definition of a building (Daniele, 2015). *DogOnt* is also concerned with home automation, and it defines both controllable and architectural building elements as well as building spaces (Corno and Bonino, 2008). Another ontology, *ThinkHome*, which consists of several smaller domain-specific ontologies, that together shape the *Smart Home Ontology*, is a refinement of the *DogOnt* ontology, extending it with representations for energy information (Kofler, Reinisch and Kastner, 2012). Part of *ThinkHome* is the *Architecture and Building Physics Information* ontology, which represents building information for the scope of Smart Home Systems. It retrieves its classes from gbXML - an XML schema for representing buildings for indoor climate and energy simulations. *OntoFM* is an ontology for sensor-based building monitoring relying on the IFC specification for representing the building, but also inheriting key concepts from *SUMO* (Dibley et al., 2012).

Smart Energy Aware Systems (*SEAS*) consists of several smaller ontologies together forming a whole infrastructure for storing data about systems that consume energy. Part of this infrastructure is the Building ontology, which defines spatial elements of a building, but only the building elements which act as a connection between building spaces. The device ontology further defines controllable devices and sensors. The ontology uses terms defined in *ifcOWL, gbXML* and *SAREF*, but does not extend these ontologies as they are only referred to by *rdfs:seeAlso*.

3 Building Topology Ontology (BOT)

From Section 2, it can be concluded that several ontologies violate the principle of reuse, which might be due to the fact that several ontologies cover the whole AEC domain. Another W3C principle is to keep schemas light for easy reuse. This might be a second key reason why so little ontology reuse is present in the AEC domain. Although a domain-specific and standardised building ontology is available (*ifcOWL*), it is far from simple / reusable. Therefore, we suggest creating a simple *Building Topology Ontology* for easy reuse across the considered domain ontologies. This ontology is made available at (Rasmussen, 2016), and it is a simple ontology only defining the core topology of a building should include the physical and conceptual objects and their relationships. This can be limited to the following concepts:

- A *building* is subdivided into *storeys* and *spaces*
- A space can be bounded by *building elements*
- A space can contain *building elements*

According to these definitions, there is a direct relationship between a space and its adjacent or contained building elements, but relationships to the storey and the building can be inferred by an owl:propertyChainAxiom. The following First Order Logic (FOL) rules or subsets of these can infer *has element*-relationships:

$$\forall b,s,r,e : \text{hasStorey}(b,s) \wedge \text{hasSpace}(s,r) \wedge \text{adjacentElement}(r,e) \rightarrow \text{hasElement}(b,e)$$

where b=Building, s=Storey, r=Space, e=Element

Limiting the ontology to 4 key classes and 5 object properties, results in a very simple ontology that can easily be extended by anyone (see Figure 2). Even the ontologies listed in Table 1 are compatible with this ontology, so also they can reuse these key concepts. Such ontology reuse can appropriately stimulate a networked data exchange as explained in the introduction of this paper.

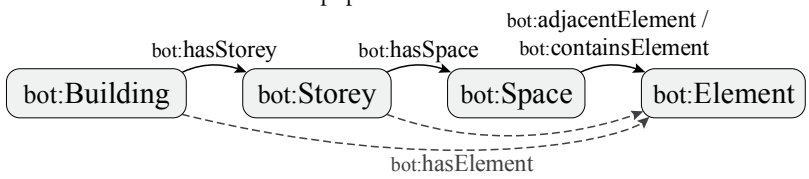


Figure 2: Simple Building Topology Ontology (BOT)

4 LINKING ONTOLOGIES

Several strategies exist for linking separate datasets that represent distinct ontologies. As we now have a situation with a central *BOT* ontology that is to be reused by several of the ontologies from section 2, we need to be specific in how this reuse should ideally take place. Overall, three approaches can be specified for linking data from different ontologies. These approaches are all visualised in the example shown in Figure 3, where a *BOT*-compliant dataset is extended with geographical and appliance data. The same approaches can be used for further extension with *SAREF*, *DogOnt* etc.

The ontology layer, also called the terminological component (TBox) is above the dashed line and the data layer also called the assertion component (ABox) is below.
To the left in Figure 3 the separate ontology abbreviated by a *h* for *heater ontology* has a class h:Heater, which is specified as a subclass of bot:Element. Thereby it inherits the properties of its superclass; in this case the OWL property chain rule defining that the heater is related to the building in which it belongs, given that it is contained in a space of that building (not shown in the figure). This automatically infers the bot:hasElement link from the building. When linking to an upper ontology this approach is used.

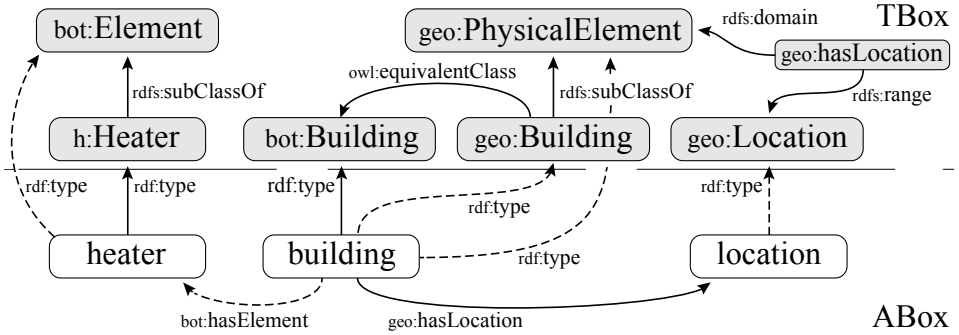


Figure 3: Extending a BOT-compliant dataset with geographical and appliance data.

owl:equivalentClass in the middle of Figure 3 specifies that a class defined in one ontology has an equivalent class in another ontology. The geo:Building class of the *geography ontology* is specified to be equal to the bot:Building class. This statement is

defined in the TBox, hence extending the *BOT* ontology, and it infers the fact that the building instance is of type geo:Building and its superclass geo:PhysicalElement.

Establishing typed links between individuals in the data layer is another approach to combining ontologies. If the GEO ontology specifies the domain and range of the geo:hasLocation property as illustrated, it is inferred that the building instance is a geo:PhysicalElement and the location instance is a geo:Location. The redundant geo:Building class is unnecessary with this approach. One advantage of the typed links approach is that the links are visible at the data layer, giving a better transparency to what is defined.

5 Conclusion

By examining domain specific ontologies in the scope of AEC it was found that they all redefine similar concepts of a building, hence creating overlaps. The suggested simple extendable BOT ontology will help overcome the observed redundancy issues currently violating W3C best practice rules, and it will allow for an easily accessible, extendable base to connect with existing and future domain ontologies, hopefully leading to a wider distribution and a more rapid development of the technology.

Distributed ontologies have several advantages over a one-size fits all, as it makes it possible for domain specialists to develop and maintain ontologies meeting the specific demands, and use of web technologies in addition makes it possible for experts in other industries to have a better interface to the AEC industry. In section 2 specific domain ontologies were investigated, but it would not be hard to imagine new ones in other domains emerging in near future.

Three different approaches to linking ontologies were explained. When extending an existing ontology with new classes, defining them as subclasses of a more general class is applicable as it allows for inheritance of properties. The suggested BOT ontology might benefit from having more specific classes for building related elements in the future as the bot:Element class is tending toward an upper ontology definition, which was not the intention. A starting point could be to define subclasses defined in IFC, and for further detailing more subclasses of bot:Element could be generated in domain ontologies. Establishing typed links was found to be the better linking approach as it provides transparency and omits the redundancy of ie. owl:equivalentClass. It also allows for a direct link between for example sensor data and a bot:Space instance, which can easily be defined by the end user.

6 Acknowledgments

Special thanks to the Alectia-Foundation and Innovation Fund Denmark for funding.

7 References

Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R. and Ives, Z. (2007) 'DBpedia: A Nucleus for a Web of Open Data', in *The semantic web*. Springer, pp. 722–735. Available at: http://www.cis.upenn.edu/~zives/research/dbpedia.pdf.

Corno, F. and Bonino, D. (2008) 'DogOnt - Ontology Modeling for Intelligent Domotic Environments', in ISWC '08 - Proceedings of the 7th International Conference on The Semantic Web, pp. 790–803. doi: 10.1007/978-3-540-88564-1_51.

Daniele, L. (2015) SAREF: the Smart Appliances REFerence ontology. Available at: http://ontology.tno.nl/saref/ (Accessed: 24 November 2016).

Dibley, M., Li, H., Rezgui, Y. and Miles, J. (2012) 'An ontology framework for intelligent sensor-based building monitoring', *Automation in Construction*. Elsevier B.V., 28, pp. 1–14. doi: 10.1016/j.autcon.2012.05.018.

Farias, T. M. De, Roxin, A.-M. and Nicolle, C. (2015) 'IfcWoD , Semantically Adapting IFC Model Relations into OWL Properties', in Proceedings of the 32nd CIB W78 Conference. Eindhoven, pp. 175–185.

Karshenas, S. and Niknam, M. (2013) 'Ontology-Based Building Information Modeling', *Computing in Civil Engineering*, pp. 476–483. doi: http://dx.doi.org/10.1061/9780784413029.060.

Kiviniemi, A. (2005) Requirements management interface to building product models, VTT Publications. Stanford University.

Kofler, M. J., Reinisch, C. and Kastner, W. (2012) 'A semantic representation of energy-related information in future smart homes', *Energy and Buildings*. Elsevier B.V., 47, pp. 169–179. doi: 10.1016/j.enbuild.2011.11.044.

Lee, S. H. and Kim, B. G. (2011) 'IFC extension for road structures and digital modeling', *Procedia Engineering*. Elsevier B.V., 14, pp. 1037–1042. doi: 10.1016/j.proeng.2011.07.130.

Liebich, T. and Wix, J. (1999) 'Highlights of the development process of industry foundation classes', in Proceedings of the 1999 CIB W78 Conference.

Lóscio, B. F., Burle, C., Calegari, N., Greiner, A., Isaac, A., Iglesias, C. and Laufer, C. (2016) Data on the Web Best Practices, W3C. Available at: http://www.w3.org/TR/dwbp/ (Accessed: 8 November 2016).

Madrazo, L., Sicilia, A. and Gamboa, G. (2012) 'SEMANCO: Semantic Tools for Carbon Reduction in Urban Planning', in Proceedings of the 9th European Conference on Product and Process Modelling.

Nederveen, S., Beheshti, R. and Willems, P. (2010) 'Building Information Modelling in the Netherlands : A Status Report', in W078-Special Track 18th CIB World Building Congress. Salford, United Kingdom, pp. 28–40.

Pauwels, P. and Roxin, A. (2016) 'SimpleBIM : From full ifcOWL graphs to simplified building graphs', in ECPPM2016.

Pauwels, P. and Terkaj, W. (2016) 'EXPRESS to OWL for construction industry: Towards a recommendable and usable ifcOWL ontology', *Automation in Construction*. Elsevier B.V., 63, pp. 100–133. doi: 10.1016/j.autcon.2015.12.003.

Pease, A. (no date) Suggested Upper Merged Ontology (SUMO). Available at: http://www.ontologyportal.org.

Rasmussen, M. H. (2016) BOT: Building Topology Ontology. Available at: www.student.dtu.dk/~mhoras/bot (Accessed: 6 December 2016).

Ronallo, J. (2012) 'HTML5 Microdata and Schema.org', Code4Lib Journal, 16. Available at: https://scholar.google.dk/scholar?hl=da&q=schema.org&btnG=.

Succar, B. (2009) 'Building information modelling framework: A research and delivery foundation for industry stakeholders', *Automation in Construction*. Elsevier B.V., 18(3), pp. 357–375. doi: 10.1016/j.autcon.2008.10.003.

Yabuki, N., Lebegue, E., Gual, J., Shitani, T. and Zhantao, L. (2006) 'International Collaboration for Developing the Bridge Product Model "IFC-Bridge"', in Proceedings of the 2006 Joint International Conference on Computing and Decision Making in Civil and Building Engineering. Montréal, Canada, pp. 1927–1936.

6.2 BOT2: Recent Changes in the Building Topology Ontology

After bringing BOT to the World Wide Web Consortium (W3C) Linked Building Data (LBD) Community Group (W3C LBD Community Group (W3C LBD CG)), some shortcomings of the ontology were identified. This paper describes these in detail and accommodate the issue by adding a set of new classes and properties.

Recent changes in the Building Topology Ontology

Mads Holten Rasmussen[1], Pieter Pauwels[2], Maxime Lefrançois[3], Georg Ferdinand Schneider[4], Christian Anker Hviid[1], and Jan Karlshøj[1]

[1] Technical University of Denmark, Copenhagen, Denmark
[2] Ghent University, Ghent, Belgium
[3] Univ Lyon, MINES Saint-Étienne, Laboratoire Hubert Curien UMR 5516, France
[4] Fraunhofer Institute for Building Physics IBP, Nuremberg, Germany

Abstract. The Building Topology Ontology (BOT) was in early 2017 suggested to the W3C community group for Linked Building Data as a simple ontology covering the core concepts of a building. Since it was first announced it has been extended to cover a building site, elements hosted by other elements, zones as a super-class of spaces, storeys, buildings and sites, interfaces between adjacent zones/elements, a transitive property to infer implicit relationships between building zone siblings among other refinements. In this paper, we describe in detail the changes and the reasons for implementing them.

1 Background

Several research projects have dealt with transforming building data to open web standards for integration with linked open data such as product catalogues, Geographic Information Systems (GIS), unit conversion, material properties etc. The most general schema for describing buildings, Industry Foundation Classes (IFC) [1], has over several attempts been translated to Web Ontology Language (OWL), latest by Pauwels and Terkaj (2016) with an ontology called *ifcOWL* [2]. However, since IFC was not initially designed for being used on the web, the structure, size and complexity of *ifcOWL* makes it hard to use and extend in practice. For that particular reason (Pauwels and Roxin, 2016) suggested a post-processing of ifcOWL called *SimpleBIM*, which omitted all geometry and intermediate relation instances between objects [3].

The Building Topology Ontology (BOT)[5] is a minimal ontology for defining relationships between the sub-components of a building [4]. It was suggested as an extensible baseline for use along with more domain specific ontologies following general W3C principles of encouraging reuse and keeping the schema no more complex than necessary [5]. Currently, the ontology is being developed by the *World Wide Web Consortium (W3C) Linked Building Data Community Group* (W3C LBD-CG), and this paper provides an overview of its current state and recent changes.

[5] https://w3id.org/bot#

2 Initial version

The first version of the ontology presented at LC3 in July 2017 included 4 key classes and 5 object properties.

In Fig. 1 above the horizontal dashed line, the classes and properties of the ontology are illustrated. A building basically consists of the building itself and a number of storeys, rooms and building elements potentially related to each other. Object properties between the classes all have domains and ranges specified, meaning that classes will be automatically inferred by a reasoning engine, given that typed links between the class instances are available. The dataset, illustrated in Fig. 1 using the horizontal dashed line, shows the inferred classes. It is for instance inferred that since `<buildingA>` has a `bot:hasStorey` link to `<storey01>`, then `<buildingA>` is an instance of `bot:Building` and `<storey01>` is an instance of `bot:Storey`. This particular example is inferred from the domain and range of `bot:hasStorey`.

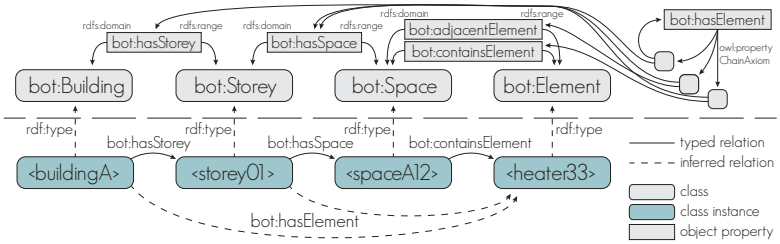


Fig. 1. BOT in the initial version. Typed links between class instances infer classes and the `bot:hasElement` property. Inferred relationships are illustrated with dashed arrows.

A `bot:hasElement` link can be inferred between some instance `?x` and an element `?e`: (a) whenever `?x` is linked by `bot:hasSpace` to some `?z`, itself linked to `e` by `bot:containsElement` or `bot:adjacentElement`, and (b) whenever `?x` is linked by `bot:hasStorey` to some `?z`, itself linked to `?e` by `bot:hasElement`. This inference capability is obtained using OWL property chain axiom.

3 Recent development

In the W3C LBD-CG, We gathered use cases and requirements for the BOT ontology and identified new competency questions that should be answered by a new version of the ontology. Fig. 2 illustrates the updated version of BOT. The following subsections list the new competency questions and how they have been taken into account in the new version of the BOT ontology.

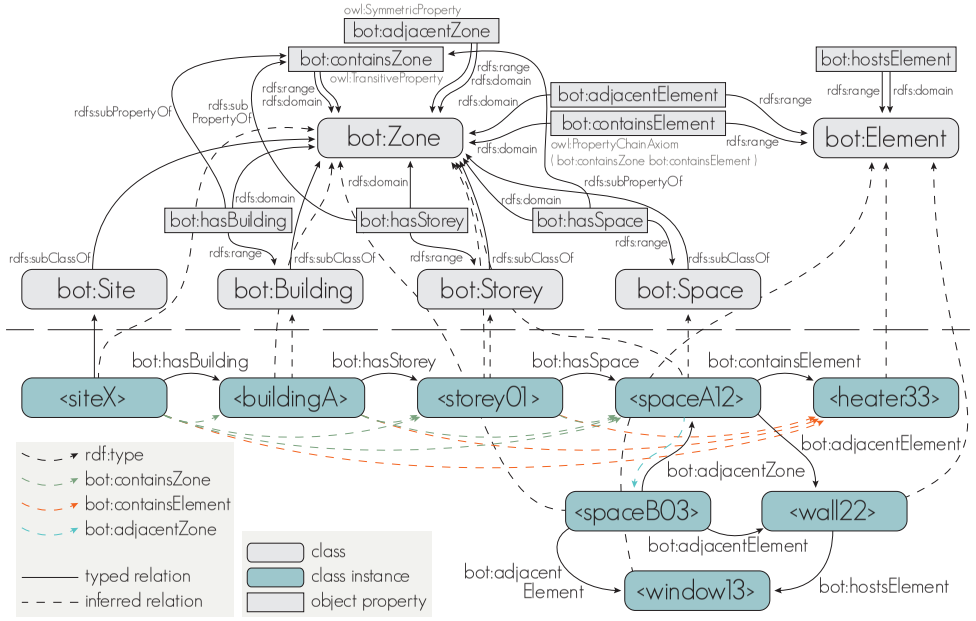


Fig. 2. BOT T- and A-Box.

3.1 Building site

New competency question: For Facilities Management (FM) purposes it is often the case that one property operator administers several buildings located at the same site. This is, for instance, the case for university campuses and hospitals. In such a case, a site and its relationship to the buildings it contains is needed.

Update on the ontology: Adding a new class `bot:Site` and an object property `bot:hasBuilding` with `rdfs:range` being a `bot:Building` to describe the relationship to the buildings contained in a site.

3.2 Domain definitions

New competency question: `bot:hasSpace`, `bot:adjacentElement` and `bot:containsElement` all had domains specified, meaning that something that has a space was necessarily inferred to be a storey. New use cases required that buildings also needed to contain spaces. Also, something that contained or had adjacency to elements was necessarily inferred to be a space. New use cases required that buildings and storeys also needed to contain or have adjacency to elements.

Update on the ontology: `bot:Site`, `bot:Building`, `bot:Storey` and `bot:Space` are all non-physical objects defining a spatial zone. A new class `bot:Zone` was added as a super-class of these. The domain of `bot:hasBuilding`, `bot:hasStorey` and `bot:hasSpace` was loosened to `bot:Zone`. A new common super-property of these object properties, called `bot:containsZone`, was added.

3.3 bot:hasElement

New competency question: The `bot:hasElement` property defined as an `owl:propertyChainAxiom`, stated that something that has a space which either contains an element or has an adjacency to one "has" the element. It further stated that if something has a storey that has such a space, then the storey also "has" the element. New use cases also required to loosen the semantics here, as in section 3.2.

Update on the ontology: The `bot:hasElement` property was changed to be valid in two situations: `bot:containsZone` followed by either `bot:adjacentElement` or `bot:containsElement`.

3.4 Hosted Elements

New competency question: The initial version of BOT did not allow for elements to be hosted by other elements. This relationship is necessary for describing situations where a window is for instance hosted by a wall, which is a fundamental part of a building's topology.

Update on the ontology: A new object property `bot:hostsElement` with domain and range being a `bot:Element` was added.

3.5 Zone connectivity

New competency question: When assessing architectural flow in a building, fire escape routes, etc., it is necessary to define a connection between zones.

Update on the ontology: `bot:adjacentZone` describes a relationship between two zones that share a common interface. With this super-property one can define more specific zone relationships stating wheter there is a direct (sharing a door), indirect (sharing a wall) or maybe an open connection between the zones. This property further enables the aggregation of zones; for instance to group architectural zones into a fire cell. In this regard `bot:containsZone` can further be used to subdivide an architectural zone into sub-zones. Fig. 3 illustrates these new concepts.

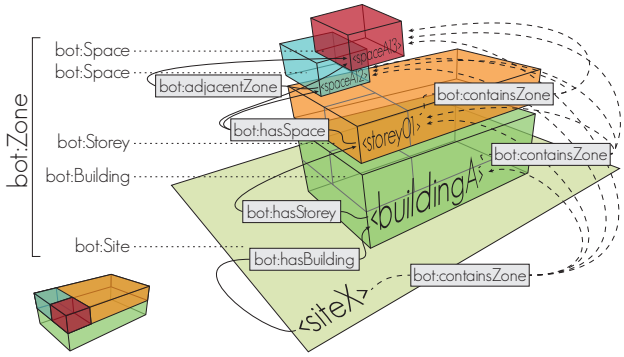


Fig. 3. BOT zone connectivity.

3.6 Interfaces

New competency question: For heat loss calculations, thermal simulations and other applications it is necessary to qualify the connection between elements or zones. A wall can cover several zones, but when defining the heat transfer area, only the shared surface between the zone and the element is of interest. BOT did not cover this representational need.

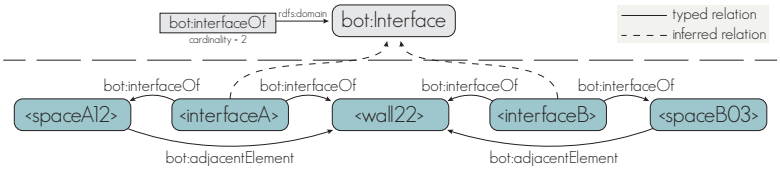


Fig. 4. BOT interfaces T- and A-Box.

Update on the ontology: A new class `bot:Interface` qualifies zone and element connectivity, i.e. the surface where two building elements, two zones or a building element and a zone meet. The interface is assigned to exactly two instances of either type `bot:Element` or `bot:Zone` by the object property `bot:interfaceOf`. Fig. 5 and 4 illustrate how to qualify the two separate adjacencies between <`spaceA12`> / <`wall22`> and between <`spaceB03`> / <`wall22`>. The same approach can for example be used to qualify a relationship between a pipe segment and the individual zones and wall elements it shares common interfaces with. These concepts are adaptations of the *Systems and Connections* pattern as defined in [6].

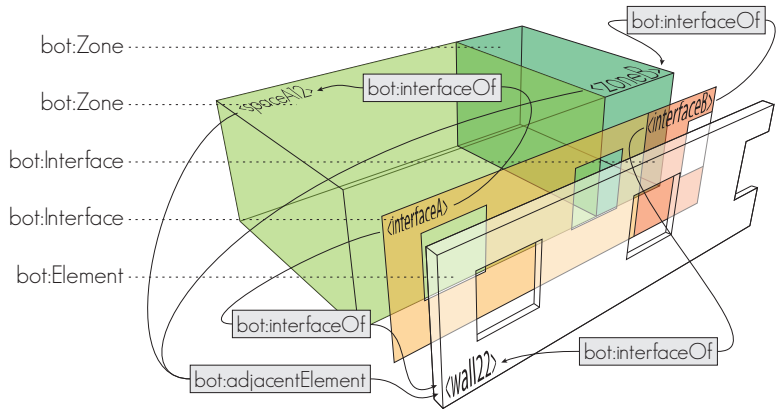


Fig. 5. BOT interfaces used to quantify each relationship between the zones and a wall which they have a shared adjacency to.

4 Conclusions and Future Work

This work provides an overview on the latest revisions and updates made to the initial version of the Building Topology Ontology (BOT) [4]. The supplementary classes and object properties enable the BOT ontology to answer six new competency questions: (1) how to define a building site, (2 & 3) how to enable transitivity when querying for either a zone of a zone or the elements that a zone "has", (4) how to have elements hosting other elements, (5) how to define adjacencies between zones (6) how to define interfaces between zone/zone, element/element or zone/element .

General development of use cases where BOT is used along with other domain ontologies is on the agenda for the W3C LBD-CG. Individual ontologies for geometry, products and properties are being developed in domain working groups, and these are all being aligned with BOT.

Implementations with existing BIM tools for extending with linked open data on the web is also on the agenda.

Acknowledgements

Special thanks to the NIRAS ALECTIA Foundation and Innovation Fund Denmark for funding.

References

1. Thomas Liebich and Jeffrey Wix. Highlights of the development process of industry foundation classes. In *Proceedings of the CIB W78 Conference*, volume 18, pages 1–18, Vancouver, Canada, May 1999.
2. Pieter Pauwels and Walter Terkaj. EXPRESS to OWL for construction industry: Towards a recommendable and usable ifcOWL ontology. *Automation in Construction*, 63:100–133, 2016.
3. Pieter Pauwels and Anna Roxin. SimpleBIM : From full ifcOWL graphs to simplified building graphs. In *Proceedings of the 11th European Conference on Product and Process Modelling (ECPPM)*, pages 11–18, Limassol, Cyprus, September 2016. CRC Press.
4. Mads Holten Rasmussen, Pieter Pauwels, Christian Anker Hviid, and Jan Karlshøj. Proposing a Central AEC Ontology That Allows for Domain Specific Extensions. In *Lean and Computing in Construction Congress (LC3) - Joint Conference on Computing in Construction (JC3)*, volume 1, pages 237–244, Heraklion, Greece, 2017.
5. Bernadette Farias Lóscio, Caroline Burle, Newton Calegari, Annette Greiner, Antoine Isaac, Carlos Iglesias, and Carlos Laufer. Data on the Web Best Practices. W3C Recommendation, W3C, January 31 2016.
6. Maxime Lefrançois. Planned ETSI SAREF Extensions based on the W3C&OGC SOSA/SSN-compatible SEAS Ontology Patterns. In *Proceedings of Workshop on Semantic Interoperability and Standardization in the IoT, SIS-IoT*, pages 1–8, Amsterdam, Netherlands, July 2017.

6.3 FORGE: Web-Based Topology Queries on a BIM Model

This paper presents the first software implementation of BOT. A custom exporter for the Building Information Modelling (BIM) authoring tool AutoDesk Revit was developed. The exporter assigns Uniform Resource Identifiers (URIs) to all elements in the model and exports BOT instances and relationships to a file. By loading the BIM model into a AutoDesk Forge based web viewer it is demonstrated how the 3D geometry can be queried and filtered based on the topological relationships.

Web-based topology queries on a BIM model

Mads Holten Rasmussen[1], Christian Anker Hviid, and Jan Karlshøj

Technical University of Denmark, Copenhagen, Denmark

Abstract. Building Information Modeling (BIM) is in the industry often confused with 3D-modeling regardless that the potential of modeling information goes way beyond performing clash detections on geometrical objects occupying the same physical space. Lately, several research projects have tried to change that by extending BIM with information using linked data technologies. However, when showing information alone the strong communication benefits of 3D are neglected, and a practical way of connecting the two worlds is currently missing.
In this paper, we present a prototype of a visual query interface running in a web browser, that enables the user to gain a deeper understanding of what can be extracted from a Building Topology Ontology (BOT) knowledge base. The implementation enables the user to query the graph, and provides visual 3D-feedback along with simple table results.
The main purpose of the paper is to establish a baseline for discussion of the general design choices that have been considered, and the developed application further serves as a proof of concept for combining BIM model data with a knowledge graph and potentially other sources of Linked Open Data, in a simple web interface.

1 Introduction

Today's BIM software allows the user to (1) create geometrical objects (2) establish geometrical constraints and relationships between them and (3) assign properties to them. Some software also offers calculation and simulation capabilities making them a full-suite solution. The complexity of the BIM tools entail that it requires special training to use them, and the fact that the most experienced engineer typically lacks IT-competencies induces that in the industry, they are often operated by technical designers. As the tools further lack interoperability with the software commonly used by engineers, they are obstructed from utilising the full BIM potential. The result is that essential information gets trapped in proprietary BIM files or "digital cemeteries" as one might be tempted to call them, while engineers waste time and introduce human errors by doing manual information takeoffs.

Point (1) and (2), respectively creating geometrical objects and establishing geometrical constraints and relationships between them, are the key features of the BIM tools of today and they conquer this task very well. When it comes to (3) assigning properties to BIM objects, however, it is not completely clear why this task is best accomplished in a large BIM suite. If other tools could instead

subscribe to the geometrical properties in the model and get notified when these change as the design progresses, a potential efficiency gain could be achieved in typical working tasks. By making data accessible in the cloud, larger engineering companies and software vendors would be able to develop task-specific design tools that are linked directly to the one true source of information, thereby reading, manipulating and writing to a linked framework of data. The prototype documented in this paper serves as a proof of concept of such a tool.

1.1 BIM and the semantic web

The Building Topology Ontology (BOT)[1] suggested by the *World Wide Web Consortium (W3C) Linked Building Data Community Group* (W3C LBD-CG) is a compact ontology designed for expressing the semantics of a building [1]. It was a result of a study in existing ontologies in the construction, automation and smart cities domains, where it was discovered that basic semantics of buildings were redundantly described in all ontologies. Currently, there are no software implementations of the ontology, which makes it hard to communicate its potential to people who do not have a background in ontology design. To illustrate the features of the ontology and for BIM practitioners and engineers to imagine potential use cases, a simple web application and an exporter for the commercial BIM tool, Autodesk Revit[2], was developed. The exporter was developed to establish a BOT-compliant knowledge representation of a building from the software tool most widely used in danish consulting engineering companies, but the knowledge representation could also have been generated through the API of another BIM-tool, or by parsing a file in the open ISO standardised Industry Foundation Classes (IFC)[2] format. One could also imagine using the same approach as presented in this paper to align with other OWL representations of building data such as ifcOWL [3].

In the following sections we seek to document the design considerations and the overall system architecture of the application and in closing we cover our reflections on future work.

2 Building Topology

In the prototype, a subset of the semantics defined in BOT is extracted in order to describe relationships in the BIM model. Some relationships are directly accessible in the Revit API and some are not. While anything can be derived by performing operations on the geometry, it can be a time-consuming task both to develop and run the code and for the purpose of this case study, only the directly available relationships have been extracted.

[1] https://w3id.org/bot#
[2] www.autodesk.com/Revit

2.1 Class definitions

In the current implementation only some of the main elements are exported. Walls, doors and windows are exported as `bot:Elements`, levels as `bot:Storeys` and spaces/rooms as `bot:Spaces`. Revit Spaces are the Mechanical engineers' representations of rooms, and the main reason for using them is that they have other predefined properties than for architectural rooms. Spaces can further be used to subdivide a room. From a BOT perspective, they are both spaces and are exported as such. This means that a model containing both rooms and spaces will have duplicate geometries at some locations, but when querying over the data this can be taken care of by including properties or filtering by the content of the URI. The design of the URIs is covered in section 3.

A level in Revit is not a physical zone with containment of and adjacencies to other zones and elements as in BOT, but even though it has no physical representation it still conceptually exists and is hence exported as such.

2.2 Relationships

A limited set of relationships have been extracted at the current stage. `bot:hasSpace` describes the spaces and rooms that are located on a level, `bot:adjacentElement` describes walls surrounding and sharing a common interface with rooms/spaces and `bot:hostsElement` describes the windows and doors that are hosted in a wall. In the further development of the exporter, also zone adjacencies/containments and relevant interfaces introduced in [4] should be exported.

2.3 Properties

Two general properties, `rdfs:label` from the Resource Description Framework (RDF) Schema[3] namespace and `rvt:guid` from a fictive Revit namespace are exported along with all the exported objects. `rvt:guid` holds the Globally Unique ID (GUID) of the Revit object and the `rdfs:label` is generated a little differently based on the type of object (see Table 1).

Table 1. Properties exported from Revit.

Revit element(s)	Revit property	OWL property	Datatype
Wall, Door, Window, Level, Space, Room	GUID	rvt:guid	xsd:string
Wall, Door, Window	Type	rdfs:label	xsd:string
Wall	Width	nir:width	cdt:length
Wall	Length	nir:length	cdt:length
Level	Name	rdfs:label	xsd:string
Room, Space	Name Number	rdfs:label	xsd:string
Room, Space	Area	nir:spaceArea	cdt:ucum
Room, Space	Volume	nir:spaceVolume	cdt:ucum

[3] https://www.w3.org/TR/rdf-schema/

As the W3C LBD-CG work on properties was at the time of writing not fully developed a set of properties in a fictive Niras[4] namespace were extracted. The prod vocabulary[5] generated from IFC4 was briefly reviewed, but none of the properties listed in Table 1 were available.

In the further development, property extraction could be handled through a mapping table, thereby leaving it up to the user to specify which vocabularies to export to.

3 System architecture

The overall system architecture enables the users to (1) generate BOT-compliant triples from a Revit model (2) Upload the BIM model and convert it for rendering in the browser (3) Upload BOT-triples to a triplestore and (4) perform SPARQL[6] queries on the triplestore and filter the 3D-view based on the result.

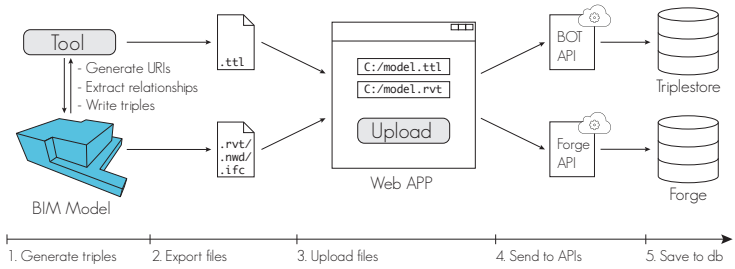


Fig. 1. The infrastructure from file export over the web application to pushing data to the two databases.

3.1 Revit

In Revit, a tool (1) generates custom property: `URI` for all objects (2) generates custom property: `host` for the project (3) generates URIs as a concatenation of `:host/:project_number/:object_type/:GUID` where the GUID is the one assigned by Revit (4) generates the BOT-relationships described in section 2 in the Resource Description Framework (RDF)[7] language and (5) exports the generated data to a Turtle[8]-file.

[4] The company in which the main author is employed.
[5] https://raw.githubusercontent.com/w3c-lbd-cg/props/master/IFC4-output.ttl
[6] https://www.w3.org/TR/sparql11-query
[7] https://www.w3.org/RDF/
[8] https://www.w3.org/TR/turtle/

Writing the URIs to a property on the Revit objects makes them available when exported to IFC or Navisworks. In the web app, these are used when filtering the 3D view. When visiting the URI in a browser the user should be presented with some useful information about the object and its relations, but at the current stage, the API is not capable of doing that.

Basing the URI on the Revit GUID establishes a mechanism for understanding the origin of an object as objects can also be created elsewhere. The URI design makes the `rvt:guid` property redundant, and it is a topic for discussion whether this property should exist.

Making the triples and the files accessible in the web app is currently handled by uploading the exported files from Revit, but it would be more user-friendly to push them directly to the web through the backend application. In Fig. 1 this would imply that step 2 and 3 could be skipped. To further improve the user experience the connection could be established through web sockets enabling a continuous communication between Revit and the server.

3.2 Web APP

Both the front- and backend of the application are built in Typescript[9] and compiled to JavaScript. JavaScript has a solid infrastructure through the Node Package Manager (npm)[10] and provides the convenience of running both on a server and in a browser. Typescript is a typed superset of JavaScript that enables better control as the application scales and further, it is the chosen language used in the Angular[11] framework which the frontend is built upon. The reason for choosing Angular is that it is developed and maintained by Google and is hence widely used and well documented. Angular further adds some guidelines for the application structure which makes it more modular and easier to scale. The BIM model is rendered in the threejs[12] based Forge Viewer[13] and Forge also handles the conversion and storage of the BIM models. Also for the triplestore a commercial product, Stardog[14], was used.

We would have preferred to base the prototype on non-commercial products like xeogl[15] on which BIMSurfer[16] is built or pure threejs for the viewer and Apache Jana Fuseki[17] for the triplestore, but the commercial solutions deliver a certainty of future maintenance and a flat learning curve since documentation is well developed.

---

[9] https://www.typescriptlang.org
[10] https://www.npmjs.com
[11] https://angular.io/
[12] https://threejs.org/
[13] https://developer.autodesk.com/en/docs/viewer/v2/overview
[14] http://www.stardog.com/
[15] https://github.com/xeolabs/xeogl
[16] http://bimsurfer.org/
[17] https://jena.apache.org/documentation/serving_data/

When adding a new project the backend creates a new database in the triplestore named P followed by the 6 digit project number (ex. `P100100`) and inserts some general information about the project in the default graph (see Lst. 1.1). Once a project is generated a so-called Bucket can be assigned to it. This can be done by clicking "add bucket" in the project overview, and doing so will generate a bucket using the Forge Data Management API and assign a new property `rvt:bucketKey` to the project in the triplestore. The bucket key must be globally unique, so in the current implementation it is niras_p followed by the project number (ex. `niras_p100100`). Buckets contain objects and objects are translated files such as BIM models. When uploading a model the file is uploaded to the backend that further sends it to the Forge API. The Forge API returns an object key and begins the translation of the file to SVF-format, which can be rendered by the viewer. The current implementation supports translations from IFC, Navisworks (NWC), Revit (RVT) and Sketchup (SKP) files but the Forge Derivative API supports +60 formats.

Listing 1.1. Project data

```
@prefix doap: <http://usefulinc.com/ns/doap#> .
@prefix prov: <http://www.w3.org/ns/prov#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
<generatedProjectURI> a foaf:Project , doap:Project ,
                                       prov:Entity ;
        rdfs:label "projectName" ;
        doap:name "projectName" ;
        doap:created "dateTime"^^xsd:datetime ;
        rdfs:comment "projectDescription" ;
        doap:description "projectDescription" .
```

When a test file is opened in the viewer there is a significant difference to how it is rendered. The RVT-file does not contain spaces and neither does the IFC. Several elements in the IFC has deformed elements. The NWC-file includes spaces, has no deformed geometry, includes spaces and textures and has a file size of less than 10 % of the IFC and nearly 1 % of the much larger RVT-file. However, the properties of the spaces don't seem to be available from the property tree and hence it is not possible to isolate spaces based on their URI.

As the BOT semantics are stored in a separate file, this turtle-file needs to be uploaded as well. The backend receives the file and puts the triples in a named graph named `tag:/:ttlFileName:` where :ttlFileName is extracted from the received file. When more models are available, one then has the option to query based on relationships in one graph or in all graphs. Further, the named graph makes it easy to delete all triples associated with a specific model.

The triplestore can be queried from the viewer and the results are returned in a table. All results consisting of URIs are extracted and in the viewer, elements carrying these URIs are isolated as shown in Fig. 2. The example shows what wall elements are adjacent to the space, and provides a visual feedback for quality assurance and understanding purposes. Communication with Forge

and Stardog is done through two separate Express[18] based REST APIs. The one communicating with Forge just adds a middle layer for security reasons and basically extends the routs already exposed by Forge. This will not be further described. The one communicating with the triplestore handles the management of projects, upload of triples and doing queries on the graph. The full set of routes are listed in Table 2.

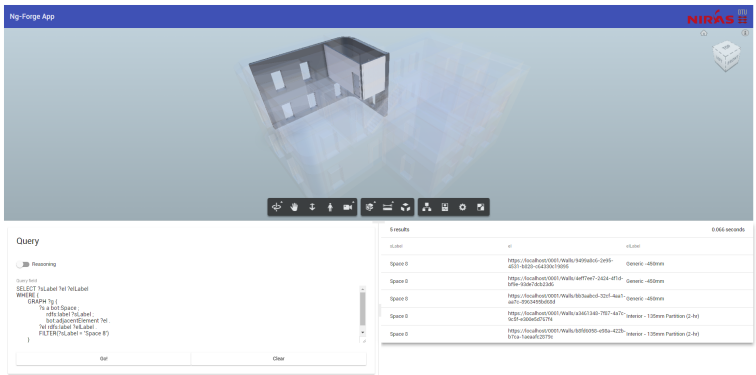


Fig. 2. Querying the model.

Get triples doesn't fully follow HTTP conventions as is it is handled by a `POST` request, but as the query is sent to the API through the request body, it is not possible to use a `GET` request.

Table 2. REST-API Routes for the backend handling communication with Stardog. :host is the address of the server that hosts the REST API (ex. https://niras.dk) and :name is the name of the database in which the requested data is stored.

Route	Method	Description
:host/projects	GET	Get all projects (databases)
:host/project	POST	Create project
:host/project/:name	DELETE	Delete project
:host/projects/:name	GET	Get project details
:host/:name/admin/postTriples	POST	Insert triples
:host/:name/admin/getTriples	POST	Get triples
:host/:name/bot	POST	Insert BOT-compliant file

[18] https://expressjs.com

4 Future work

The prototype is just a proof of concept and the list of future work is exhaustive. Some considerations were already considered in section 3, but in general we would like to see more projects and software implementations dealing with export of BIM data to BOT knowledge bases.

It would be interesting to see projects dealing with the generation of data outside the BIM tool and doing reasoning on a combination of the two. A simple example could be grouping of spaces into different zones like fire cells. It would also be interesting to see a project dealing with management of a continuous communication between the BIM tools and the triplestore. This implies dealing with properties that change over time, calculations based on properties that might change over time and the problems this might entail.

Some effort by the W3C LBD-CG is being put into developing product and property ontologies and it would be an obvious improvement of the Revit exporter to align with these. Product classes would allow for more specific queries than what is possible with `bot:Element` and there are several use cases that could benefit from having updated geometrical data from the BIM tool available at hand.

5 Conclusion

The main contribution of the study is the illustration of a simple architecture for combining 3D model data with data from a triplestore. It was succeeded to develop a working prototype of a tool to perform queries in the browser with visual 3D representations of the results, and it is the authors' belief that the visual feedback will enhance the communication of what can be expressed with BOT. At the current stage of development only some BOT data is exported from the BIM tool and this is a problem since the users might misinterpret the capabilities of the ontology.

References

1. Mads Holten Rasmussen, Pieter Pauwels, Christian Anker Hviid, and Jan Karlshøj. Proposing a central aec ontology that allows for domain specific extensions. In *Joint Conference on Computing in Construction*, volume 1, pages 237–244, 2017.
2. Thomas Liebich and Jeffrey Wix. Highlights of the development process of industry foundation classes. In *Proceedings of the 1999 CIB W78 Conference*, 1999.
3. Pieter Pauwels and Walter Terkaj. EXPRESS to OWL for construction industry: Towards a recommendable and usable ifcOWL ontology. *Automation in Construction*, 63:100–133, 2016.
4. Mads Holten Rasmussen, Pieter Pauwels, Maxime Lefrançois, Georg Ferdinand Schneider, Christian Anker Hviid, and Jan Karlshøj. Recent changes in the building topology ontology. In *Linked Data in Architecture and Engineering*, 2017.

6.4 OPM1: OPM: An Ontology for Describing Properties That Evolve over Time

This paper presents an ontology for property management (Ontology for Property Management (OPM)). This includes terminology to describe multiple property states for a single property, thereby allowing it to change over time while keeping track of its full historical evolution.

OPM: An ontology for describing properties that evolve over time

Mads Holten Rasmussen[1],
Maxime Lefrançois[2], Mathias Bonduel[3], Christian Anker Hviid[1], and Jan Karlshøj[1]

[1] Technical University of Denmark, Kgs. Lyngby, Denmark
mhoras@byg.dtu.dk
[2] Univ Lyon, MINES Saint-Étienne, Laboratoire Hubert Curien UMR 5516, France
[3] KU Leuven, Dept. of Civil Engineering, Technology Cluster Construction, Ghent, Belgium

Abstract. The W3C Linked Building Data on the Web community group discusses different potential patterns to associate values to properties of building elements. In this paper, we are interested in enabling a different value association method for these and other properties, to account for changes in time, or to annotate a value association with metadata such as provenance, reliability and origin data. Existing ontologies in the Architecture, Engineering and Construction (AEC) industry are reviewed first and we motivate the use of the Smart Energy-Aware Systems (SEAS) ontology as a starting point. Next, we list new competency questions to represent the aforementioned metadata and develop an extension of SEAS named the Ontology for Property Management (OPM). We illustrate the use of OPM with different scenarios where a value association needs to be annotated or updated in a dataset using SPARQL UPDATE queries.

1 Introduction

The W3C Linked Building Data on the Web Community Group (W3C LBD CG)[1] brings together experts in the area of Building Information Modeling (BIM) and Web of Data technologies to define existing and future use cases and requirements for Linked Data based applications across the life cycle of buildings. Of particular interest to this group (and possibly other domains) is the assignment of properties to any feature of interest (FoI) - in this particular case, building-related elements.

Design is an iterative process, and this is, in particular, the case when designing a building. The iterative nature entails that information which is valid at one point in time might no longer be valid in the future, and keeping an overview of information validity hence becomes a cumbersome task. When change management is furthermore handled in a predominantly manual manner by tracking changes in meeting minutes, mail correspondences or as a worst case, in the heads of the individual project participants it constitutes a serious threat to the project execution [5].

Modeling design changes that occur over time is complex as one must define when some FoI is the same as it was before, only with a changed property, and when it is a completely new FoI. Is a particular door, for instance, the same after the width of it has changed? Linked data provides us with the means to allow a concept defined by one party to be extended by other parties, and this is a useful feature in construction projects where most items have interfaces to several different parties from different domains. The door might have a requirement for thermal capacity defined by one party, whereas another party has

defined the fire rating. In those cases, it will cause complications if a FoI is substituted with a new one, and in this regard, it is preferred that the individual property is changed instead. However, changing a property can also cause problems as there are many interdependencies between properties of FoIs in a building. Changing a door width influences the heat loss of the room if the thermal resistance of the door is different from its hosting wall. To a certain degree the consequences might not be significant enough to revise the heating system, but as changes add up it might be necessary. Tracking of property evolution history allows designers to relate any derived property to the particular state of the property on which it was derived. Hence, at any time it is possible to evaluate the significance of the design changes and even assess consequences of a design change. In this work, we suggest a modeling approach which allows properties of any FoI to change over time while still keeping track on the history. The scope of the work is the core functionality of OPM, so dealing with derived properties and classification of a property's reliability is not included, although it is covered by the current version of the OPM ontology[2].

2 Ontologies and Patterns to Model Properties

The following ontologies can be used to describe properties, value assignment for properties, and provenance information. The Smart Energy-Aware Systems (SEAS) ontology [7,8] consists of a set of modules together providing terminology to describe physical systems and their interrelations. The core modules related to property management are the seas:FeatureOfInterestOntology and the seas:EvaluationOntology. Together they describe that some FoI can have a property assigned using the seas:hasProperty predicate, and that different evaluations of a same property can be described using the seas:Evaluation class.

The Provenance Ontology [6] provides classes and properties to describe provenance information such as when a prov:Entity was generated, by what prov:Activity if was generated, and who was the prov:Agent that was associated with that activity.

The schema.org ontology is developed as a collaborative, community activity, initiated by the major search engines [2]. It contains an updated version of the GoodRelations ontology [4], one of the main ontologies regarding e-commerce, which is now deprecated. schema.org allows to define quantitative property values by using the schema:value, schema:minValue and schema:maxValue predicates.

From within the W3C LBD CG, a need for a standardized approach towards building-related properties emerged [1]. Future developments aim at proposing both standardized modeling patterns (e.g. by using one or more levels of complexity as demonstrated in Section 2) and predefined, but expendable, lists of building-related properties.

The CDT Datatypes in [9] leverage the Unified Code of Units of Measures UCUM to define a series of RDF Datatypes to encode quantity values. The value and the unit are defined in the same literal with a custom RDF datatype, e.g. `"115 km.h-1"^^cdt:ucum`, or `"0.27 W/(m2.K)"^^cdt:ucum`.

Let `<wall_A>` be a FoI in a building model. At the moment of writing, three potential Linked Data patterns were proposed to the W3D LBD CG [1], each having a different degree of complexity: Level 1 (L1), Level 2 (L2) and Level 3 (L3). Each level number refers to the number of steps/relations between the FoI and the actual object (literal or individual) that encodes the value of its property. The following paragraphs illustrate how these different levels can be used to model the thermal transmittance of wall

element <wall_A>, and its main material. Throughout the paper, we use namespace prefixes as provided by the http://prefix.cc/ service.

Level 1: As illustrated in Listing 1, the FoI is directly linked to the UCUM literal that encodes the quantity value of the thermal transmittance of the wall, using a OWL Datatype property. It is also directly linked to the individual that represents material *concrete*, using an OWL Object property.

Listing 1: Level 1 using a cdt:ucum literal.

```
# ontology
ex:thermalTransmittance a owl:DatatypeProperty . ex:mainMaterial a owl:ObjectProperty .
# data
<wall_A> ex:thermalTransmittance "0.27 W/(m2.K)"^^cdt:ucum ; ex:mainMaterial ex:concrete .
```

Level 2: This level explicitly identifies the thermal transmittance property of <wall_A> with an intermediate instance of class seas:Property, following the approach defined in the W3C and OGC Semantic Sensor Networks (SSN) ontology [3]. Using SSN, this property instance may be the object of some observation or actuation activity. The SEAS ontology reuses this pattern, but defines OWL Datatype property seas:simpleValue and OWL Object property seas:value to directly link an instance of seas:Property to a literal that encodes its value, or to an individual that encodes its value, respectively [7].

Listing 2: Level 2 using a cdt:ucum literal.

```
# ontology
seas:thermalTransmittance a owl:ObjectProperty ; rdfs:subPropertyOf seas:hasProperty .
ex:mainMaterial a owl:ObjectProperty ; rdfs:subPropertyOf seas:hasProperty .
# data
<wall_A> seas:thermalTransmittance <wall_A#prop> ; ex:mainMaterial <wall_A#mat> .
<wall_A#prop> seas:simpleValue "0.27 W/(m2.K)"^^cdt:ucum .
<wall_A#mat> seas:value ex:concrete .
```

Level 3: SEAS defines an additional level where the link between a property instance and its value can be qualified. This is done using an intermediary object of class seas:Evaluation. The instance of seas:Evaluation can be used to specify the validity context for the value association (e.g. valid during a certain temporal interval), or the type of evaluation (e.g. the maximal operating value). OWL Datatype property seas:evaluatedSimpleValue and OWL Object property seas:evaluatedValue are then used to link an instance of seas:Evaluation to a literal that encodes the evaluated value for the property, or to an individual that encodes this value, respectively [7].

Listing 3: Level 3 using a cdt:ucum literal.

```
# ontology
seas:thermalTransmittance a owl:ObjectProperty ; rdfs:subPropertyOf seas:hasProperty .
ex:mainMaterial a owl:ObjectProperty ; rdfs:subPropertyOf seas:hasProperty .
# data
<wall_A> seas:thermalTransmittance <wall_A#prop> .
<wall_A#prop> seas:evaluation <wall_A#prop-eval1> .
<wall_A#prop-eval1> seas:evaluatedSimpleValue "0.27 W/(m2.K)"^^cdt:ucum.
<wall_A> ex:mainMaterial <wall_A#mat> .
<wall_A#mat> seas:evaluation <wall_A#mat-eval1> .
<wall_A#mat-eval1> seas:evaluatedValue ex:concrete .
```

3 The proposed OPM ontology

This ontology answers a set of competency questions that were identified during interviews with AEC experts. Section 4 lists and answers these competency questions, but for lack of space this section first describes the main terms of the ontology.

Property states. The value of a property can undergo changes over time, e.g. during the building design process or when managing an existing building. The Ontology for Property Management (OPM) enables to describe these changes using L3-modeling of properties of SEAS; reusing concepts from schema.org and PROV-O; and introducing a few classes specific to property management. These classes are all subclasses of opm:PropertyState, which itself is a subclass of seas:Evaluation and defined to differentiate with other types of evaluations as follows. A opm:PropertyState is an evaluation holding the value and metadata about a property that was true for the given time. Metadata must as a minimum be the time of generation stated by prov:generatedAtTime, but preferably also a prov:wasAttributedTo reference to the agent who created the state. Assigning a state to a property is achieved with the OWL Object Property opm:hasPropertyState (sub property of seas:evaluation) which will by its rdfs:range infer that the state is an instance of opm:PropertyState.

Current state and deleted states. So as to ensure efficient management of properties using SPARQL engines, a subclasses of opm:PropertyState is defined to deal with finding the most recent property states: opm:CurrentPropertyState.

PROV-O includes prov:generatedAtTime to indicate the generation time of some resource. Achieving the most recent state can therefore be accomplished by performing a sub-query to first achieve the most recent timestamp and then find the particular opm:PropertyState instance that was generated at this time. However, this query is (a) complex to write and (b) performs poorly. Therefore the opm:CurrentPropertyState class was introduced to explicitly state that a property state is the most recent one. The performance was evaluated by loading 50000 FoIs each having 5 properties with 5 states (5,250,000 triples total) into a triplestore. Two queries (1) by prov:generatedAtTime and (2) by opm:CurrentPropertyState were performed in order to retrieve the latest state of 100 properties. From a cold start (1) returned a result in 6900 ms and (2) in 640 ms, meaning a time reduction of a factor 10. A cold start was also evaluated by redoing each query 10 times and registering the minimum query time. The cold start results were (1) 4780 ms and (2) 630 ms respectively. The tests were performed on local triplestore served on a Lenovo P50 laptop with Intel Core i7-6820HQ 2.70 GHz CPU and 32 GB 2133 MHz DDR ram.

In order to maintain the history of the project and to be able to revert to an earlier state, data should never be removed from the knowledge graph. Using a opm:Deleted marker class enables omission of deleted properties when querying the data store, while they can still be stored in the same database. A deletion is reverted by introducing a new state that inherits the properties of the most recent state.

Property values. OPM does not provide a specific predicate for value assignment, but instead encourages the use of schema:value for single values and, schema:minValue/ schema:maxValue for ranges.

4 Demonstration of property management using OPM

In this section we show how to use OPM for managing properties in combination with SEAS, schema.org, PROV-O and a certain schema defining domain-specific properties, for example the emerging PROPS ontology for the AEC industry. For each of the competency questions below, that have been identified during interviews with AEC experts, a small dataset and example queries were developed and implemented in an online demo.[3]

Competency question 1: How to semantically describe a property such that its value is changeable while its historical record is maintained? Figure 1 illustrates how to assign a property with OPM. When modeling an OPM-compliant L3 property, the property instance must have at least one opm:hasPropertyState relation to a state (entails that the state is an opm:PropertyState class) and the opm:CurrentPropertyState class must be assigned to the most recent state. A state can host any metadata about the property, but should as a minimum have a value and preferably a generation time assigned. In the example (Fig. 1), schema.org is used for the relation between the state and the actual value of the property and PROV-O is used for assigning a generation time and the rdfs:domain of prov:generatedAtTime entails that `<state>` becomes an instance of prov:Entity. Listing 4 shows a complete query to assign an OPM compliant property state to some FoI.

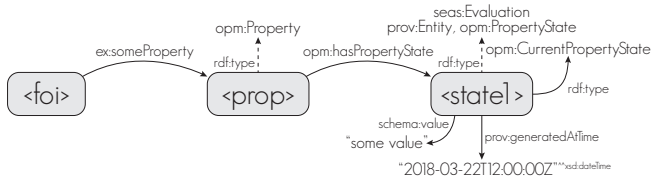


Fig. 1: Modeling a property using states

Listing 4: Insert a new property and an initial property value.

```
INSERT {
  ?foiURI ?prop ?propURI .
  ?propURI opm:hasPropertyState ?stateURI .
  ?stateURI a opm:CurrentPropertyState ;
    prov:generatedAtTime ?now ;
    schema:value ?val .
} WHERE {
  BIND(<wall_A> as ?foiURI)                 # define URI of FoI
  BIND(<wall_A#prop> as ?propURI)           # define URI of Property
  BIND(ex:thermalTransmittance as ?prop)    # define property
  BIND("0.27 W/(m2.K)"^^cdt:ucum as ?val)   # define value
  BIND(<wall_A#state> as ?stateURI)         # define URI of State
  BIND(NOW() as ?now)                        # get current time
  # Do not create a new property instance if the FoI already has it
  MINUS { ?foiURI ?prop ?propURI }
}
```

Competency question 2: How to revise a property value? Making property revisions is done by assigning a new opm:PropertyState to the property instance. The new property must be an instance of opm:CurrentPropertyState and as there cannot be two current states of a property, the class specifying that the previous property state was the current state must be removed (Fig. 2). Listing 5 shows an update query that will handle this.

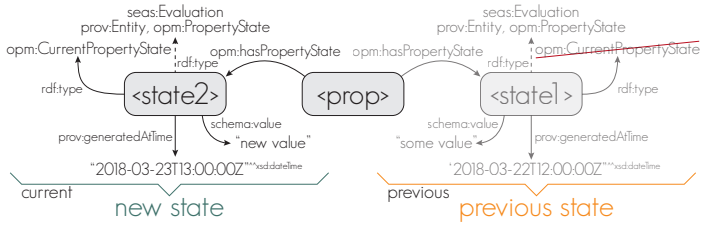


Fig. 2: Revising a property value. Revised state to the left and old property state to the right.

Listing 5: Update a property value.

```
DELETE { ?previousState a opm:CurrentPropertyState }
INSERT {
  ?propURI opm:hasPropertyState ?stateURI .
  ?stateURI a opm:CurrentPropertyState ;
    prov:generatedAtTime ?now ;
    schema:value ?val .
} WHERE {
  BIND(<wall_A#prop> as ?propURI)       # define URI of Property
  BIND("0.25 W/(m2.K)"^^cdt:ucum as ?val) # define new value
  BIND(<wall_A#state2> as ?stateURI)    # define URI for State
  BIND(NOW() as ?now)                   # get current time stamp
  ?propURI opm:hasPropertyState ?previousState .
  ?previousState a opm:CurrentPropertyState ;
    schema:value ?currentVal .          # get value of current state
  FILTER(?val != ?currentVal) # don't update if equal to latest state
}
```

Competency question 3: How to delete a property while still being able to retrieve the history of it and not break all the links to derived properties that depend on it? Deleting a property is done by assigning a new opm:PropertyState to the property instance. The new property state is both an instance of opm:CurrentPropertyState and opm:Deleted, and the opm:CurrentPropertyState class of the previous current state is removed (Fig. 3). Thereby the history is maintained and metadata such as when, why and by whom the property was deleted can be added to the opm:Deleted instance. Listing 6 shows a query for deleting a property in an OPM-compliant way.

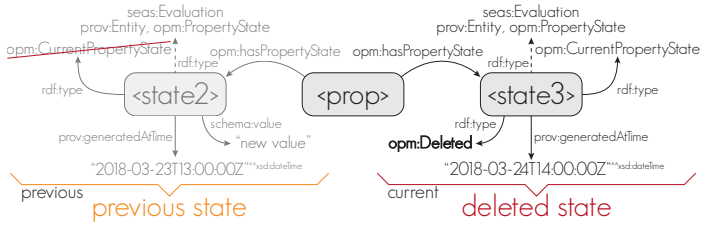


Fig. 3: Deleting a property.

Listing 6: Delete property.

```
DELETE { ?previousState a opm:CurrentPropertyState }
INSERT {
  ?propURI opm:hasPropertyState ?stateURI .
  ?stateURI a opm:CurrentPropertyState , opm:Deleted ;
            prov:generatedAtTime ?now .
} WHERE {
  BIND(<wall_A#prop> as ?propURI)    # define URI of Property
  BIND(<wall_A#state3> as ?stateURI) # define URI of deleted State
  BIND(NOW() as ?now)                # get current time stamp
  ?propURI opm:hasPropertyState ?previousState .
  ?previousState a opm:CurrentPropertyState . # get current state
  # do not delete if the current state is already a opm:Deleted
  MINUS { ?previousState a opm:Deleted }
}
```

Competency question 4: How to restore a deleted property? Restoring a deleted property is done by retrieving the metadata of the most recent property state that is not an instance of opm:Deleted and copy this to a new state (Fig. 4). It requires a sub-query to retrieve the time stamp of such property state (Lst. 7) and as the test in Section 3 revealed, this process is quite resource intensive. However, as it is not an everyday operation it is still acceptable. The reason for creating a new state rather than just deleting the opm:Deleted instance along with its data is to maintain the complete history (incl. deleted states) and record who restored the property, why and when.

Listing 7: Restore property.

```
DELETE { ?previousState a opm:CurrentPropertyState }
INSERT {
  ?propURI opm:hasPropertyState ?stateURI .
  ?stateURI a opm:CurrentPropertyState ;
    prov:generatedAtTime ?now ;
    ?key ?val .
} WHERE {
  BIND(<wall_A#prop> as ?propURI)    # define URI of Property
  BIND(<wall_A#state4> as ?stateURI) # define URI of new State
  BIND(NOW() as ?now)                # get current time stamp
  # get time stamp of most recent property state that was not deleted
  { SELECT ?propURI (MAX(?time) AS ?t)
    WHERE {
          ?propURI opm:hasPropertyState ?s .
            ?s schema:value ?lastVal ;
        prov:generatedAtTime ?time .
      MINUS { ?s a opm:Deleted }
    } GROUP BY ?propURI }
  # get key-value pairs of latest state that is not deleted
  ?propURI opm:hasPropertyState [
    prov:generatedAtTime ?t ;
    ?key ?val ]
  FILTER(?key != prov:generatedAtTime) # filter out time stamps
  # get previous state
  ?propURI opm:hasPropertyState ?previousState .
  ?previousState a opm:CurrentPropertyState .
}
```

Competency question 5: How to retrieve the full history of how the value of a property has evolved over time? The full history is simply retrieved by querying for all seas:PropertyStates of the property. By making it optional for a state to have a schema:value assigned, deleted states are also returned (Lst. 8).

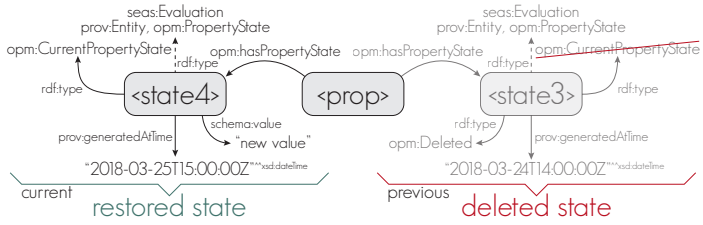


Fig. 4: Restoring a property.

Listing 8: Get property history.

```
SELECT ?dateTime ?value WHERE {
  <wall_A#prop> opm:hasPropertyState ?state .
  ?state prov:generatedAtTime ?dateTime .
  OPTIONAL{ ?state schema:value ?value }
} ORDER BY ?dateTime

### RESULTS
# March 22, 2018 12:00 PM    0.27 W/(m2.K)
# March 23, 2018 1:00 PM     0.25 W/(m2.K)
# March 24, 2018 2:00 PM     -
# March 25, 2018 3:00 PM     0.25 W/(m2.K)
```

Competency question 6: How to retrieve only the latest value of a property? The latest value is retrieved by querying for the opm:PropertyState which is an instance of opm:CurrentPropertyState. The result of the query in Listing 9 is simply `"0.25 W/(m2.K)"^^cdt:ucum`.

Listing 9: Get property value.

```
SELECT ?value
WHERE { <wall_A#prop> opm:hasPropertyState [
   a opm:CurrentPropertyState ; schema:value ?value ] }
```

Competency question 7: How to simplify a complex OPM property (using states) for easier and faster querying? Simplification from L3 to L2 or even L1 can be handled, but will consequently entail some information loss. For both L2 and L1 the property history is lost since only the most recent property state is inferred.

When simplifying to L2 any key-value pair of the most recent state is inferred directly to the property instance node (Fig. 5, yellow). This approach has the advantage that all metadata of the current state of the property such as property unit, provenance data etc. is maintained. It will also still allow for the property value to be specified as a range using schema:minValue and schema:maxValue. The disadvantage is that the property value is still two steps/relations away from the FoI.

When simplifying to L1 the value of the most recent state is inferred directly to the FoI as a datatype property (Fig. 5, red). The advantage is that it becomes very easy and fast to query for the properties of a FoI. Units can still be assigned using custom datatypes but simplifying to L1 comes with some disadvantages. First of all, no metadata can be assigned and hence provenance data is lost and value ranges

are not supported. Further, it will be incorrect to use an owl:ObjectProperty as a owl:DatatypeProperty, and therefore one of the following approaches must be considered: (1) the original property must be described as an rdfs:Property meaning that the dataset becomes less descriptive (RDFS level instead of OWL-DL level) or (2) when simplifying to L1 another predicate (a owl:DatatypeProperty) must be inferred instead. The latter could be handled by adding a suffix to the property URI as illustrated in Fig. 5 and have both an owl:ObjectProperty and owl:DatatypeProperty described in the ontology.

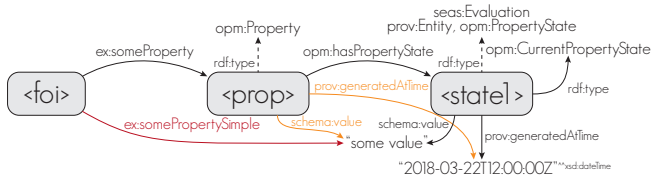


Fig. 5: Simplifying a L3 property to L2 and L1.

Inferring the simplified properties along with the more complex property states makes it easier to query the dataset, and there is no problem in having the data in the same data store. Listing 10 shows an update query that will automatically update all L1 simplifications and a similar approach can be used for L2 simplifications. These queries could be run as a routine job (backward chaining). As an alternative, the same dependency could simply be defined in SWRL rules (forward chaining). The latter has the advantage that there will never be a situation where an outdated property is returned, but it has the cost of a reduced query performance.

Listing 10: Simplify from OPM to simple datatype property.

```
DELETE { ?foi ?p ?simpleValOld }
INSERT { ?foi ?p ?simpleValNew }
WHERE {
  ?foi ?p ?prop .
  ?prop opm:hasPropertyState ?state .
  ?state a opm:CurrentPropertyState ;
    schema:value ?simpleValNew .
  # Get old simplified value (if any)
  OPTIONAL {
    ?foi ?p ?simpleValOld .
    FILTER(?simpleValOld != ?prop) # don't delete L2 property
    FILTER(?simpleValNew != ?simpleValOld) # don't update if unchanged
  }
}
```

5 Conclusions and Future Work

With this work, we propose an extension of the SEAS evaluation ontology with terms specific to tracking properties that evolve over time. We use existing ontologies to describe how to manage property changes of a building element; a wall instance, but OPM is also relevant in any other domain that deal with properties that change over time. The construction industry is rather fragmented, and in a construction project, there are many interdependencies between properties. OPM could be a good foundation for working with derived properties as it allows a derived property to be linked directly to the specific state of its arguments. Further investigation of the potential of OPM in relation to property interdependencies is therefore a future research topic of interest.

OPM can also be used to keep track of changes in BIM models received from other project participants. Communicating with an OPM-compliant SPARQL endpoint directly from a BIM authoring tool to store only state changes of properties could save space and allow insights that comprises an interesting research subject. For legal applications it would be interesting to investigate the use of blockchain technologies to document traceable state changes using OPM. It would also be worth investigating the possibility of having complex and simplified representations of properties co-existing, and using any for answering queries.

Notes

[1]W3C LBD CG - https://www.w3.org/community/lbd
[2]OPM - https://w3id.org/opm
[3]http://www.student.dtu.dk/~mhoras/ldac2018/

References

1. Bonduel, M.: Towards a PROPS ontology (2018), https://github.com/w3c-lbd-cg/lbd/blob/gh-pages/presentations/props/presentation_LBDcall_20180312_final.pdf
2. Guha, R.V., Brickley, D., Macbeth, S.: Schema. org: evolution of structured data on the web. Communications of the ACM **59**(2), 44–51 (2016)
3. Haller, A., Janowicz, K., Cox, S.J.D., Le Phuoc, D., Taylor, K., Lefrançois, M.: Semantic Sensor Network Ontology. W3C Recommendation, W3C (Oct 19 2017), https://www.w3.org/TR/vocab-ssn/
4. Hepp, M.: Goodrelations: An ontology for describing products and services offers on the web. In: International Conference on Knowledge Engineering and Knowledge Management. pp. 329–346. Springer (2008)
5. Kiviniemi, A., Fischer, M.: Requirements management interface to building product models (2004)
6. Lebo, T., Sahoo, S., McGuiness, D.: PROV-O: The PROV Ontology. W3C Recommendation, W3C (Apr 13 2013), https://www.w3.org/TR/prov-o/
7. Lefrançois, M.: Planned ETSI SAREF Extensions based on the W3C&OGC SOSA/SSN-compatible SEAS Ontology Patterns. In: Proceedings of Workshop on Semantic Interoperability and Standardization in the IoT, SIS-IoT, (July 2017)
8. Lefrançois, M., Kalaoja, J., Ghariani, T., Zimmermann, A.: SEAS Knowledge Model. Deliverable 2.2, ITEA2 12004 Smart Energy Aware Systems (2016), 76 p.
9. Lefrançois, M., Zimmermann, A.: The unified code for units of measure in RDF: cdt:ucum and other UCUM datatypes. In: Extended Semantic Web Conference (2018), demonstration paper

6.5 REQ: Managing Space Requirements of New Buildings Using Linked Building Data Technologies

This paper suggests how OPM can be used to describe a requirement for a future property. Specifically, it demonstrates how to describe different space requirements and comparing those to the actual design parameters. Space requirements are described at type level using BOT in combination with Web Ontology Language (OWL) property restrictions.

Managing Space Requirements of New Buildings Using Linked Building Data Technologies

M.H. Rasmussen, C.A. Hviid & J. Karlshøj
Department of Civil Engineering, Technical University of Denmark, Kgs. Lyngby, Denmark

M. Bonduel
Department of Civil Engineering, Technology Cluster Construction, KU Leuven, Ghent, Belgium

ABSTRACT: Any stakeholder operating in the AEC industry knows that designing a building is a complex and highly iterative task. The project evolves over time and changes happen rapidly, meaning that design requirements, as well as solutions (often as a consequence), must undergo revision. Since building requirements are, however, documented and handled in a predominantly manual manner, the work processes are not aligned with the dynamic nature of the projects. Tracking and acting upon changes is a manual, and therefore an error-prone and labour intensive task. In this article, we suggest a generic method for working with the concept of spaces at different abstraction levels in order to compare requirements with actual properties in a non-static manner using semantic web technologies, primarily developed by the W3C Linked Building Data (LBD) Community Group. The generic modelling approach has the potential of also being applied to other concepts than building spaces.

1 INTRODUCTION

When buying a product you can rightly expect it to correspond to the technical specifications on which the purchase was originally based. When buying a building, however, the reality is unfortunately not always so (Kiviniemi 2005). Bertelsen (2003) describes construction as a complex system because of three main characteristics: (1) autonomous agents (2) undefined values and (3) non-linearity. Delivering a complete product specification in the form of a building program at day 1 is nearly impossible as everyone gains knowledge and insights as the design evolves, and as a result, the building program itself cannot be static during the design. The documentation and handling of it, therefore, needs to be dynamic, which is unfortunately typically not the case (Kiviniemi 2005). The majority of building design processes are today characterized by manual information extraction from static documents, and as the design progresses it becomes a cumbersome task for the project participants to keep track of, and meet the evolving client requirements. Because of the predominantly manual information handling, the quality of information exchange between project stakeholders is furthermore highly determined by the social capabilities and communicative skills of the individual practitioners (Bendixen 2007). This is a challenge that the methodology of Building Information Modelling (BIM) will hopefully remedy over time. However, unfortunately the BIM authoring tools of today are not delivering satisfactory interoperability, and data is therefore often trapped in data silos (Terkaj 2017).

In this article, we first provide a brief overview of existing software and data modelling approaches that focus on building requirements specification. We then argue why we believe semantic web technologies can possibly provide the means to overcome current challenges when dealing with the dynamic behaviour of building requirements. Based on knowledge manually deduced from existing document-based building programs and discussions with practitioners in the consulting engineering company, Niras, we have defined a set of competency questions. These were used as constraints for what the data model should be capable of. The model was developed accordingly, chiefly by using terminology defined in already existing and widely adopted ontologies. Lastly, we developed a set of tests to evaluate the modelling approach on the Common BIM Model "Duplex Apartment"[1]. The dataset was established partly by manually defining requirements as

[1] https://www.nibs.org/?page=bsa_commonbimfiles#project1

an RDF-graph (Resource Description Framework) following the suggested modelling approach, and partly by using a custom developed exporter for the BIM authoring tool, Revit[2]. The latter establishes an RDF-graph using ontologies provided by the World Wide Web Consortium Linked Building Data Community Group (W3C LBD-CG).

1.1 *Open standards*

The effort of storing knowledge in a construction project, including the information exchange between its stakeholders, has been addressed by the buildingSMART organisation. With standards such as Industry Foundation Classes (IFC) (Liebich and Wix 1999), Information Delivery Manuals (IDM) and Model View Definitions (MVD) they deliver a solid framework for information exchange and storage.

The W3C also has made efforts to standardize information exchange using semantic web technologies such as the Web Ontology Language (OWL) to construct formal vocabularies to describe a certain domain of interest. The scope of these technologies is not limited to the AEC industry alone, and therefore researchers and practitioners from a wide variety of domains are contributing to their continuous development.

One main difference between the above two methodologies is that OWL relies on an Open World Assumption (OWA), meaning that the schema can evolve over time to include concepts not initially thought of. This is quite different from typical database systems that depend on a Closed World Assumption (CWA) for defining schemas, such as IFC. Another benefit is that the full dataset does not need to be available at one location but can be combined with other datasets as needed, being both Linked Open Datasets (LOD) available online (material data, weather data, geographical data etc.) and private datasets, possibly hosted by other project stakeholders. Owners of such private datasets can restrict the access to specific partners.

The W3C Resource Description Framework (RDF) standard is used to describe Linked Data in a directed graph consisting of a collection of triples. A triple has three parts: a node (the subject), an edge (the predicate) and another node (the object) connected to the first node through the predicate-edge. All sub-elements of a triple are made globally unique by denoting them with a Uniform Resource Identifier (URI) except for objects that are literal values such as strings, integers, Booleans etc. The datatype of such literals are also described with a URI, and is often defined in an ontology version of the Extensible Markup Language (XML) Schema Definition (XSD). Both the terminology layer (TBox) - including semantics for classes and properties, and the data layer (ABox), covering individual instances and their interrelations, are described using RDF. The W3C encourages developers to make their ontologies publicly available so that useful ontology-related information can be retrieved from the URI. To continue, the W3C recommends that terms from widely adopted ontologies are used to explicitly describe the data layer.

An RDF graph is traversed using the SPARQL Protocol and RDF Query Language (SPARQL) and if it is described using widely adopted ontologies it is possible to structure generic, globally applicable queries to deduce knowledge. The semantics described in the TBox also allow reasoning engines to deduce implicit knowledge from what is explicitly defined in the ABox. A simple example: If `chair` is a sub-class of `furniture` (TBox), then all instances of `chair` are also instances of `furniture` (ABox).

1.2 *Cloud-based BIM solutions*

Although building programs are typically defined in static documents (Word, PDF) there are a few cloud-based BIM applications for building requirements management on the market. They typically consist of a user interface (UI) that enables the user to do create, read, update and delete (CRUD) operations on requirements stored in a central database along with a communication link to native BIM authoring tools. Since each internal database has a closed proprietary schema rather than a schema defined according to the previously described open standards, interlinking the requirements to information that exists outside the application is not easily accomplished. Additionally, migrating from one tool to another is seen as a cumbersome task. Some applications do offer a REST (representational state transfer) API (application programming interface) providing a machine-accessible interface to the internal data model. However, the design of this interface is also following a proprietary schema and therefore a deep understanding of this schema is a prerequisite for interpreting and using the data in other applications.

Onuma and dRofus are examples of BIM applications for requirements management that offer a

[2] https://github.com/MadsHolten/revit-bot-exporter

REST API to interact with the data model[3,4], and they use XML and JavaScript Object Notation (JSON) respectively as data format. Both APIs offer only limited interaction with the data model and although accessible from outside, they are tightly coupled to their native data models.

The SPARQL Protocol (Feigenbaum et al. 2013) and SPARQL Graph Store HTTP protocol (Chimezie Ogbuji 2013) are W3C recommendations specifying how to make an RDF-graph available through a REST architecture. Accessing the graph is achieved by sending a SPARQL query to a URI hosting a SPARQL endpoint, and this provides an interface for clients to do CRUD operations on the dataset. A cloud-based BIM tool using the W3C open standards to describe the schema could host a SPARQL endpoint in order to allow clients to access the data model using standardised SPARQL queries, but to our knowledge, no such tool currently exists.

1.3 *Linked Building Data*

Research has provided us with several examples of how semantic web technologies can be used to enhance data handling in the AEC industry. The typical research contribution is an ontology which describes a subset of the construction domain with a distinct scope such as smart homes and sensor data or even the construction domain as a whole. Pauwels & Terkaj (2016) proposed ifcOWL as the OWL-based counterpart for the IFC schema and probably the most widely adopted ontology in the AEC domain.

It has later been argued that this quite literal conversion of the IFC schema is not appropriate as it (1) contains artefacts from the EXPRESS schema from which it originates making queries less logic and (2) describes too wide a scope, thereby violating the W3C best practice of omitting redundancy and making it hard to get familiarized with (Pauwels & Roxin 2016; Rasmussen et al. 2017a).

Another, more modular approach for building-related ontologies is suggested by the W3C LBD-CG. A minimal ontology, the Building Topology Ontology (BOT) (Rasmussen et al. 2017a) describes the main concepts of a building and thereby serves as an extensible core for describing any concept in its context of a building. Another ontology, PROPS, describes building-related properties and is at the time of writing a conversion of the properties contained in the IFC4 schema[5]. The conversion approach is also used in the PRODUCT ontology which describes building-related products. Finally, the Ontology for Property Management (OPM) extends concepts from the Smart Energy-Aware Systems (SEAS) ontology to provide the means to describe property reliability as well as property changes over time using property states.

Both the IFCtoLBD-converter[6] (Bonduel et al. 2018) and an exporter for Revit[7] (Rasmussen et al. 2017b) generate LBD compliant RDF triples from conventional BIM models.

In this study we have used and extended a set of widely adopted web ontologies for property handling (schema.org/goodrelations), provenance data (PROV-O), literal units (Unified Code for Units of Measure (UCUM) (Lefrançois 2018)) along with the earlier mentioned LBD ontologies. Using these ontologies in combination with OWL description logics, we illustrate an approach for specifying project specific space classes that explicitly state the client's requirements. We further show how the architectural spaces can automatically inherit requirements based on the class they are assigned to using standard OWL reasoning engines. Queries to compare and evaluate requirements to actual properties of the space instances are further illustrated and a simple use case, is presented to simulate both requirement and property changes and the handling of these.

2 REQUIREMENTS MODELLING

In this section we illustrate how concepts defined in the BOT, OPM and schema.org ontologies can be used to model space requirements. Initially, various client requirements specifications for construction projects in which Danish consulting company Niras has been involved, were reviewed. In these specifications, it is common practice to specify space requirements at *type* level rather than at *instance* level.

IFC and various BIM authoring tools use the concept of *types* and include a mechanism for inheriting properties of a *type* to *instances* belonging to that *type*. *Instances* can further extend the set of properties at an individual level and properties can even be overridden (Borgo et al. 2014). It is clear that the instances belong at ABox level, but the concepts of

[3] http://www.onuma-bim.com/platform/api
[4] https://wiki.drofus.com/display/DV/REST+API
[5] https://github.com/w3c-lbd-cg/props/blob/master/IFC4-output.ttl
[6] https://github.com/jyrkioraskari/IFCtoLBD
[7] https://github.com/MadsHolten/revit-bot-exporter

space and object *types* are less obvious. In BIM tools, space and object *type instances* are defined at the data layer rather than the schema layer, but from an ontology engineering perspective, it would arguably be more correct to consider the *type instances* themselves at schema level.

In the following section, we will investigate a TBox modelling approach of *space types* that must be capable of providing answers to the following competency questions:

- *CQ1:* How to model a *space type*?
- *CQ2:* How to assign a quantitative requirement to a *space type*?
- *CQ3:* How to state that a *designed space* instance matches a *space type* of the client's requirements specification?
- *CQ4:* How to check if a property that also exists as a requirement is fulfilled by the architectural design?
- *CQ5:* How to check an adjacency or quantity requirement?
- *CQ6:* How to update a *space type* and its assigned requirements?

2.1 *CQ1: Modelling a space type*

Modelling a *space type* is achieved by defining a project-specific extension of BOT, in this case in the namespace of the building client. In Figure 1 the class `client:spacetype_bathroom1` is defined as a sub-class of `bot:Space` meaning that any instance of the class will be classified as a `bot:Space`. The `rdfs:label` and `rdfs:comment` are widely adopted predicates from the RDF Schema (RDFS) that provide a human-readable specification of the class. In this example, in Danish and English language.

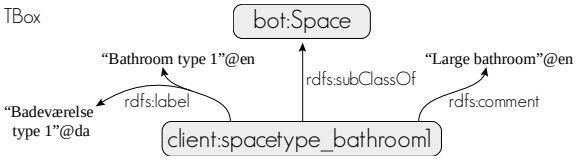


Figure 1. Modelling a *space type* with BOT.

2.2 *CQ2: Assigning a quantitative requirement*

In order to meet the demands for modelling a *space requirement*, it should be possible to capture the following information:

- Range, (minimum and maximum) or specific value to be matched
- Quantitative unit of the value
- Property changes over time (deleted, modified)

OWL includes logics to describe property restrictions for classes. For example, it is possible to describe that `:BlueCars` is a sub-class of all cars that have a blue color, which entails that every instance of the `:BlueCars` class will consequently be blue. Figure 2 illustrates how an `owl:Restriction` can be used to describe that all instances of `client:spacetype_bathroom1 have a props:area` with the value `client:property_001`. This objectified property belongs to the ABox of the client's dataset, which allows it to evolve over time.

Rasmussen et al. (2018) describe three levels of complexity for assigning properties to some feature of interest (FoI). Level 3, the most expressive form, satisfies the demand of allowing property changes over time and is therefore used to model *space requirements*. Figure 2 illustrates how the property has a property state (`client:state_p001_001`) assigned. This state is currently classified as the `opm:CurrentPropertyState`, which indicates that it is the most recent state of the property but this might change over time as the client requirements are revised. A new class `opm:Required` which we suggest to implement as an extension of OPM is used to specify that the state is a requirement rather than a designed property. A value range is specified using properties defined in schema.org and the generation time is captured using PROV-O. The unit is given as part of the value string using a custom datatype based on UCUM. Further metadata such as who created the property state for which reason can also be attached.

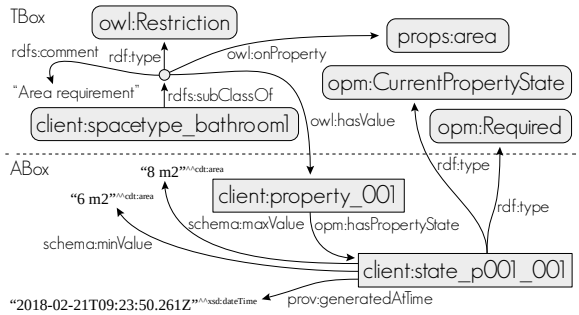


Figure 2. Assigning a requirement using (Rasmussen et al. 2018) Level 3.

2.3 *CQ3: Mapping designed space instances to spaces requested by the client*

At one point, as the architectural design progresses, the architect's dataset will hold a number of *designed spaces* that should match the *space types* required by the client. At this point, the architectural spaces are geometrically defined, and therefore they have an actual area.

Mapping a *designed space* to a client *space type* is handled by stating that the *designed space* is an instance of the specific *space type* class. Figure 3 illustrates how properties of the client *space type* (`client:spacetype_bathroom1`) are inherited to all instances of this class. In this example, spaces `inst:room123` and `inst:room213` both inherit `client:property_001` (and its property state) as the value for property `props:area`.

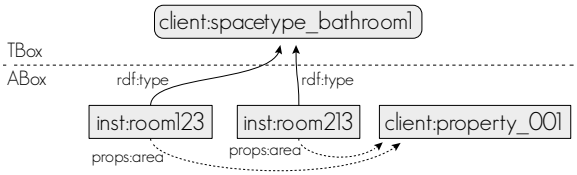


Figure 3. Two designed spaces are classified as client: spacetype_bathroom1. Therefore the properties (requirements) of the client *space type* are inherited by the *designed spaces*.

2.4 *CQ4: Checking that a requirement is fulfilled*

When the same space property exists both as a requirement and a designed property it is possible to do a comparison in order to check if the requirement is met. Figure 4 illustrates `inst:room123` which has the property `props:area` assigned twice. Explicitly as a result of its geometry and implicitly as a requirement inherited by the mechanism described in Figure 3. Performing the comparison is possible by traversing the graph using a SPARQL query.

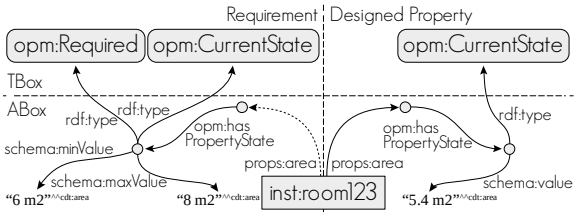


Figure 4. Requirement vs. property.

Listing 1 shows a SPARQL query to retrieve all violations of the `props:area` requirement in the model, when both the requirements and designed properties are all in one database. The query is structured as a graph traversal which operates by matching the defined patterns. The first triple pattern maps anything that is an instance of `bot:Space` to the variable `?space`. The next pattern is a sub-query which is used to get data from the state of `props:area` that is classified as `opm:Required`. The variable `?space` is used to match the same space, and the URI of the property object is mapped to variable `?reqURI`. All states of the property are assigned to variable `?reqState` but the next two triples limit the result to only include the one state which is both classified as `opm:Required` and `opm:CurrentPropertyState`. Since a requirement can be specified either as an exact match or as a range, each of the `schema:value` patterns are optional.

A similar pattern is used to get the actual property and by using a filter it is ensured that the requirement is not assigned to variable `?propURI` (since both match the pattern). The value of `?propURI`'s latest state is assigned to variable `?val` and compared to the required range to check if it is violated. A result is returned only if the requirement is violated.

Replacing `?space` with `inst:room123` or the URI of any other space will return violated requirements for this particular space and this approach can be used to switch any variable with a constant.

Listing 1. SPARQL query to retrieve violated requirements

```
SELECT *
WHERE {
  # Must be a space
  ?space rdf:type bot:Space .

  # Sub-query to get requirement
  {
    SELECT ?space ?reqURI ?reqVal ?reqMax ?reqMin
    WHERE {
      ?space    props:area           ?reqURI .
      ?reqURI   opm:hasPropertyState ?reqState .
      ?reqState rdf:type opm:Required .
      ?reqState rdf:type opm:CurrentPropertyState .
      OPTIONAL {?reqState schema:value ?reqVal}
      OPTIONAL {?reqState schema:minValue ?reqMin}
      OPTIONAL {?reqState schema:maxValue ?reqMax}
    }
  }

  # Get property
  ?space props:area ?propURI .
  FILTER(?propURI != ?reqURI) # Disjoint from req
  ?propURI opm:hasPropertyState ?propState .
  ?propState rdf:type opm:CurrentPropertyState .
  ?propState schema:value ?val

  # Compare requirements to actual value
  BIND( ?value !=  ?reqVal AS ?matchViolated   )
  BIND( ?value <?reqMin    AS ?minViolated     )
  BIND( ?value >?reqMax    AS ?maxViolated     )

  # Show only results where a requirement is
  # violated
  FILTER( ?matchViolated || ?minViolated ||
                                  ?maxViolated )
```

2.5 *CQ5: Adjacency and quantity requirements*

Specifying adjacency or quantity requirements is not different from any other requirement. However, special queries must be used to check whether these are violated. The same is the case for other requirements such as zone or element containment.

Checking if the required quantity of spaces of a certain *space type* is met, is accomplished by the query shown in Listing 2. Accessing the requirement can

be done in the main query since it is not necessary to distinguish between two properties of the same kind, but in order to count the number of *space type* occurences, a sub-query is necessary. Listing 2 shows the optional sub-query to count the number of *designed space* instances per client *space type*. Each *space type* is assigned a unique `?reqURI` for the `props:quantity` property requirement, so this can be used for the grouping. This query is executed before continuing to the next step where requirement `props:quantity` is compared to `?value`.

Listing 2. Sub-query to count number of *designed space* instances that have a specific quantity requirement assigned.

```
{
  SELECT ?reqURI (COUNT(?reqURI) AS ?qty)
  WHERE {
    ?space props:quantity ?reqURI .
  } GROUP BY ?reqURI
}
```

Finding violated adjacency requirements is likewise handled by first getting the requirement (like illustrated in Listing 1). Also in this case it can be done in the main query, and this time it is only necessary to get the `schema:value` and bind it to `?reqVal`. By using the MINUS clause a result is only returned if the space does not have an adjacency to a *designed space* defined as an instance of the required client *space type*.

Listing 3. SPARQL query to retrieve violated adjacency requirements.

```
# Return result if the space does not have an
# adjacent space of the required type
MINUS {
  ?space    bot:adjacentZone ?adjSpace .
  ?adjSpace rdf:type         ?reqVal .
}
```

2.6 *CQ6: Performing updates*

Changes to property requirements are according to Rasmussen et al. (2018) handled by creating a new current state and removing the `opm:Current-PropertyState` from the evaluation that was previously defined as the current state. This can be achieved with an update query which can be generated using the OPM query generator JavaScript library[8]. Since all the queries explicitly look for the current state of both properties and requirements, the evaluations will automatically reflect the changes.

3 USE CASE

To illustrate a possible workflow for modelling, mapping and evaluating requirements, a simple use case was set up. The Common BIM Model "Duplex Apartment" was used as a reference, and the Revit BOT exporter[9] plugin (Rasmussen et al. 2017b) was extended to include the concept of *space types* and OPM *property states*. The exporter was used to export the architectural model in LBD format. The steps to establish the dataset were the following:

1) Define client requirements in RDF. (This step should preferably be accomplished through a UI)
2) Run BOT exporter in Revit to:
   - Create and assign a Revit URI parameter to spaces and elements
   - Create Revit SpaceTypeURI parameter
   - Export BOT relationships and properties to RDF
3) Specify *space type* URI corresponding to the URI used for the client *space type* and re-export triples (Figure 5)
4) Use Dynamo script to export zone adjacencies to RDF. This functionality will be implemented in the exporter plugin in the future

Identity Data	
Number	B203
Name	Bedroom 2
OmniClass Table 13 Category	13-51 21 11: Bedroom
URI	https://architect.com/projA/room_0b74b3fa-1a...
SpaceTypeURI	https://client.com/projX/spacetype_bedroom

Figure 5. Revit shared parameters for URI and SpaceTypeURI.

Once the dataset was available it was loaded into a triplestore in order to do the checks described in the previous section. The checking is implemented in a JavaScript based testing tool that is available online[10], while the results are presented here.

3.1 *Testing property requirements*

All *space types* in the test have an area requirement specified. In general, the areas are fulfilled by the designed spaces, except for `inst:spacetype_bedroom` and `inst:spacetype_bathroom1`. Once the dataset is loaded into the triplestore, the tool performs the query from Listing 1 to find area requirement violations. Listing 4 shows the results.

Listing 4. Test tool output for violated property requirements. Numbers in parenthesis are (actual/range).

```
- 'Bathroom 2 B204' violates req. (5.44/(6-))
- 'Bathroom 2 A204' violates req. (5.42/(6-))
```

[8] https://www.npmjs.com/package/opm-qg

[9] https://github.com/MadsHolten/revit-bot-exporter

[10] www.student.dtu.dk/~mhoras/ecppm2018/test.zip

```
- 'Bedroom 1 B202'  violates req. (26.12/(20-25))
- 'Bedroom 2 B203'  violates req. (26.18/(20-25))
- 'Bedroom 2 A203'  violates req. (26.18/(20-25))
- 'Bedroom 1 A202'  violates req. (26.12/(20-25))
```

3.2 *Checking quantity of spaces*

Some *space types* have a requirement for quantity of *designed space instances*, and for `inst:spacetype_living_room` a requirement of seven occurrences is specified, which is not fulfilled in the case of the Duplex house model. A query to group the rooms into apartments based on the room numbers (which are suffixed with either A or B) was implemented in the test tool. The query from Listing 2 was modified slightly in order to accommodate this before counting the number of *designed space* occurrences for each *space type*. The result of this query was, correctly, that the requirement was not met as there is only one living room per apartment in the Duplex model.

Listing 5. Test tool output for violated quantity requirements. Numbers in parentheses are (actual/range).

```
- 'Living Room A102' (1/7)
- 'Living Room B102' (1/7)
```

3.3 *Testing adjacency requirements*

Two adjacency requirements were given as a client requirement:

- `spacetype_living_room`/`spacetype_kitchen`
- `spacetype_bedroom`/`spacetype_bathroom1`

The query from Listing 3 revealed that requirement 2 is only fulfilled by one of the bedrooms in each appartment, which is correct.

3.4 *Changing requirements*

By performing four SPARQL update queries, three client requirements were revised and a new one was added:

- Area requirement for `spacetype_bathroom1` relaxed from 6 m^2 to 5 m^2.
- `props:quantity` for `spacetype_living_room` relaxed from 7 to 1.
- New *space type* `spacetype_bedroom2` with `props:quantity` requirement of 1 and area requirement of minimum 9 m^2 added.
- Adjacency requirement between `spacetype_bedroom` and `spacetype_bathroom1` deleted by appending new `opm:PropertyState` of class `opm:Deleted`.

Re-running the tests from section 3.1 and 3.2 now concludes that the area requirements of `'Bathroom 2 A204'` and `'Bathroom 2 B204'`, the `props:quantity` requirement for `spacetype_living_room` and the adjacency requirements for `'Bedroom 1 A202'` and `'Bedroom 1 B202'` are no longer violated.

The requirements for the new bedroom type cannot be evaluated with the queries presented in Section 2 since the class is not assigned to any spaces. In order to check for required spaces which have not been instantiated, one must do a query starting from the client *space type* itself, and even though this is less intuitive, it is possible. Listing 6 shows a query pattern to retrieve a *space type* which has a quantity requirement assigned, but is not instantiated.

Listing 6. Find *space types* with a quantity requirement but no instances.

```
# GET QUANTITY REQUIREMENT
?spaceType rdfs:subClassOf [
  rdf:type owl:Restriction ;
  owl:onProperty props:quantity ;
  owl:hasValue ?reqURI
] .
MINUS { ?space a ?spaceType }
```

Since the initial requirements are all available in the model, the architect is able to track the changes and relate a property compliance check to a certain state of a requirement.

4 CONCLUSIONS AND FUTURE WORK

The main outcome of this work is the illustration of how to use semantic web technologies and existing ontologies, BOT in particular, to establish a knowledge model of requirements for spaces of a new building. The model illustrates an approach to describe space requirements at *type level* in a way that utilizes OWL reasoning capabilities thereby providing best practice examples of how to extend BOT at project level.

In the use case presented in this work, *designed architectural spaces* inherit properties of the *space types* described by the client. The same approach could be used for (1) other features of interest such as building elements or the building as a whole or (2) other generalisations such as an automation control strategy. In the use case, the requirements were modelled manually, but it is obviously not practical for practitioners to do this, so some CRUD application with a user-friendly UI should be developed.

Another interesting use case to investigate is *derived requirements*. Specific requirements such as minimum and maximum temperature, fresh air supply etc. are a result of the more general requirement; the desired indoor climate class (according to EN15251) and can be deduced by taking into account properties of the users of the space (ie. activity level, clothing). The specific indoor climate requirements set the constraints for the technical systems to be designed by the HVAC engineer, and modelling these interdependencies could potentially provide a valuable tool for *design change consequence analysis*.

In the use case, all data was stored in the same triplestore, but in a real world implementation the client would probably make the project specific classes and associated requirements available to project participants as a SPARQL-endpoint hosted on a separate server or as part of a Common Data Environment (CDE). Further research in how such an implementation could be configured is a separate research topic.

The use of OPM enables documentation of design and requirement changes over time, and in the use case it was used to revise requirements. Inferring into the graph that a requirements check was made based on a specific state of a requirement could be used for documentation purposes, but this was out of the scope for this work. The legal aspects of being able to document design changes, potentially in combination with block chain technology could entail great benefits and composes a separate research topic.

In summary, this work illustrates a data modelling approach that provides all the means to overcome current challenges when dealing with evolving design data and requirements in the complex construction industry. It is our belief that future BIM tools can benefit from adopting these technologies and methodologies.

5 ACKNOWLEDGEMENTS

Special thanks to the NIRAS ALECTIA Foundation and Innovation Fund Denmark for funding.

6 REFERENCES

Bendixen, M. 2007 The challenges of consulting engineers. *PhD Thesis.* Kgs. Lyngby: Technical University of Denmark.

Bertelsen, S. 2003 Construction as a Complex System. *Proceedings of IGLC* 11(February):143–68.

Bonduel, M., Oraskari, J. & Pauwels, P. 2018 The IFC to Linked Building Data Converter - Current Status. *6th Linked Data in Architecture and Construction Workshop (LDAC).*

Borgo, S. et al. 2014 Towards an Ontological Grounding of IFC *6th Workshop Formal Ontologies Meet Industry, Joint Ontology Workshops, CEUR.*

Chimezie O. 2013 SPARQL 1.1 Graph Store HTTP Protocol. Retrieved May 15, 2018 (https://www.w3.org/TR/sparql11-http-rdf-update/).

Feigenbaum, L. et al. 2013 SPARQL 1.1 Protocol. Retrieved May 15, 2018 (https://www.w3.org/TR/sparql11-protocol/).

Kiviniemi, A. 2005 Requirements Management Interface to Building Product Models *PhD Thesis.* Stanford University.

Lefrançois, M. & Zimmermann, A. 2018 The unified code for units of measure in RDF: cdt:ucum and other UCUM datatypes *Proceedings of the International Semantic Web Conference (ISWC)*, demonstration paper, submitted.

Liebich, T. & Wix, J. 1999 Highlights of the Development Process of Industry Foundation Classes. *Proceedings of the 1999 CIB W78 Conference*.

Pauwels, P. & Roxin, A. 2016 SimpleBIM : From Full IfcOWL Graphs to Simplified Building Graphs. *European Conference on Product and Process Modelling (ECPPM).*

Pauwels, P. & Terkaj, W. 2016. EXPRESS to OWL for Construction Industry: Towards a Recommendable and Usable IfcOWL Ontology. *Automation in Construction* 63:100–133. doi: 10.1016/j.autcon.2015.12.003.

Rasmussen, M. H. et al. 2017a Proposing a Central AEC Ontology That Allows for Domain Specific Extensions. *Lean and Computing in Construction Congress - Volume 1: Proceedings of the Joint Conference on Computing in Construction* 237–44. doi: 10.24928/JC3-2017/0153.

Rasmussen, M. H., et al. 2017b Web - Based Topology Queries on a BIM Model. *5th Linked Data in Architecture and Construction (LDAC2017) Workshop*. doi: 10.13140/RG.2.2.22298.95685.

Rasmussen, M. H. et al. 2018 OPM: An Ontology for Describing Properties That Evolve over Time. *6th Linked Data in Architecture and Construction Workshop (LDAC), CEUR.*

Terkaj, W. et al. 2017 Reusing Domain Ontologies in Linked Building Data : The Case of Building Automation and Control. *8th Workshop Formal Ontologies Meet Industry, Joint Ontology Workshops, CEUR*

6.6 SSN: Demo: Integrating Building Information Modeling and Sensor Observations Using Semantic Web

This paper demonstrates an integration between an architectural BIM model and sensors installed in the actual building. The integration uses mapping between the LBD-dataset and the sensor observations described with the Semantic Sensor Networks Ontology (SSN)/Sensor, Observation, Sample, and Actuator Ontology (SOSA) ontology. The paper, however, describes a future vision of having the low voltage engineer describing these relationships as part of the project design. Thereby, the implementation outlines a design approach for describing sensors and actuators of a future building.

Demo: Integrating Building Information Modeling and Sensor Observations using Semantic Web

Mads Holten Rasmussen[1], Christian Aaskov Frausing[1], Christian Anker Hviid[1], and Jan Karlshøj[1]

Technical University of Denmark, Kgs. Lyngby, Denmark
mhoras@byg.dtu.dk

Abstract. The W3C Linked Building Data on the Web community group is studying modeling approaches for the built environment using semantic web technologies. One outcome of this effort is a set of proposed ontologies together providing necessary terminology for the Architecture, Engineering, Construction and Operation (AECO) domains. In this paper, we demonstrate an integration between different datasets described using these ontologies in combination with the standard ontology for representing Sensors, Observations, Sampling, Actuation, and Sensor Networks (SSN/SOSA). In combination, the datasets cover the building's overall topology, 2D plan geometry, sensor and actuator locations and a log of their observations. We further suggest an integrated design approach that enables the designers to explicitly express the semantics of the sensors and actuators from the early stages of the project such that they can be carried on to construction and operation.

1 Introduction

The AECO industry involves numerous stakeholders. Each stakeholder generates, consumes and manipulates a shared, distributed project material on which they are all dependent. This dataset continuously evolves, and as the project undergoes different phases (programming, design, construction, operation), it is often handed over to new project participants. Handling a large distributed dataset in a fragmented, temporary organization is a challenge, and as the dataset usually consists of proprietary files, printed documents and the like, the complexity grows. It is a well-established fact that every time the project material is handed over at stage changes data is lost [2].

Building Information Modeling (BIM) is a methodology aimed at minimizing information loss by using technologies to model project data in a structured way. The buildingSMART organization is engaged in the development of industry standards to provide consensus in BIM implementations, and with the Industry Foundation Classes (IFC) schema [5] most terminology for describing a building is provided. However, where IFC is mainly aimed at file-based information exchanges, numerous research projects are focusing on how web technologies can support the dynamic nature of the projects by providing a data-based information exchange [10]. The World Wide Web Consortium Linked Building Data Community Group (W3C LBD CG) engages domain experts in the development of ontologies and modeling approaches, thereby hopefully paving the way for a near-future semantic web-based BIM.

In this work, we present an implementation between three datasets: (1) the architectural model described using the Building Topology Ontology (BOT) including simple

plan geometry described using Open Geospatial Consortium (OGC) Well Known Text (WKT) formatted literals (2) containment-relationships between building spaces and sensors/actuators and (3) actual observations from a building in operation. Dataset (2) was established in post-processing by mapping datasets (1) and (3) programmatically, but the ambition is that a semantic web-based BIM can enable the designers to describe the sensor and actuator semantics as part of the design material. Section 4 illustrates an integrated design workflow that supports this goal. Lastly we discuss the potential of a semantic web-based BIM for future smart buildings.

2 Proposed LBD standards

There exists numerous ontologies aimed at the AECO industry and ifcOWL[1] by Pauwels & Terkaj, 2016 [7] is probably the widest adopted. As the name indicates, it is a Web Ontology Language (OWL) version of the IFC schema, and as pointed out by [6,9] it (1) carries on relics from the EXPRESS schema on which IFC is based and (2) covers too broad a scope of which some is already described by widely adopted ontologies (provenance data, units of measure etc.). The Building Topology Ontology (BOT[2]), on the other hand, is a simple ontology aimed solely on describing tangible and spatial elements of a building in their topological context to each other. It is included in the work by the W3C LBD CG among other initiatives such as the PRODUCT[3] ontology for describing building related products and the PROPS[4] ontology describing properties.

BOT was proposed as a central AEC ontology that provides generic terms for specifying any feature of interest in the context of its location in a building [9]. It includes the predicate bot:containsElement which has an owl:propertyChainAxiom stating the element inheritance from sub- to super zones. This property entails that a building inherits all elements contained in spaces of the building, and thereby provides a practical mechanism for establishing an overview of the subcomponents of the building. This is advantageous e.g. for cost scheduling or grouping of Heating, Ventilation and Air Conditioning (HVAC) zones.

In the context of sensors and actuators bot:containsElement is a useful term to describe the location in relation to the building in which they operate. The sosa:hosts-relationship between a sosa:Platform and a sosa:Platform, sosa:Sensor, sosa:Actuator or sosa:Sampler can be used for describing a similar relationship [3]. The space that hosts a sosa:Sensor would in this case be classified as a sosa:Platform. However, this domain specific term is hard to interpret for practitioners of other domains, and therefore the general building specific bot:containsElement can be used in addition to provide more knowledge.

3 The datasets

The case model, Navitas, is an educational facility in Aarhus, Denmark. It was completed in 2014, has a footprint of approximately 38,000 m^2 above ground and the BIM model

[1] http://www.buildingsmart-tech.org/ifcOWL/IFC4#
[2] https://w3id.org/bot#
[3] https://github.com/w3c-lbd-cg/product
[4] https://github.com/w3c-lbd-cg/props

has a total of 1392 spaces. A data dump from the Building Management System (BMS) provides a dataset consisting of observations from sensors and actuators for 301 of the building's spaces. The number of observations from the different spaces varies from 7294 to 13855 and are from the period April 18, 2017 - March 4, 2018. Table 1 is illustrating an example measurement.

Table 1: An example of available observations for each space.

Item	Example	Unit
Time	2017-09-16 16:21:54	
Room status	STANDBY	STANDBY/COMFORT
Regulator status	COOLING	COOLING/HEATING
Holding time	1800	s
Air quality	-	ppm
Actual temp.	22.8	degC
Setpoint temp. (calculated)	21	degC
Setpoint temp. (comfort)	21.5	degC
Setpoint temp. (standby)	21.5	degC
Hysteresis temp. (heat)	0.3	degC
Hysteresis temp. (ventilation)	0.3	degC
User temp. (maximum)	23	degC
User temp. (minimum)	19	degC
Radiator opening	0	%
Ventilation flow	100	%
Ventilation unit	VE10	
Minimum ventilation (comfort)	10	%
Minimum ventilation (standby)	10	%
Minimum ventilation (night)	0	%
Boot ventilation	0	%
Actual LUX	450	lux
Desired LUX	300	lux
Light 1	0	%
Light 2	0	%

Besides from the BMS data, the architectural model in the proprietary format of the Revit BIM authoring tool was available. The space numbers used in the Revit model and the BMS system were assumed to match.

The data was parsed to RDF to create a knowledge graph described with terminology from BOT, PROPS, CDT, SOSA and GEO (Fig. 1).

4 An integrated workflow

During the design of a building, the low voltage engineer must develop specifications for the BMS. The system must comply with the client's monitoring demands, the capabilities of the HVAC system and the control strategy defined by the indoor climate engineer. Further, it must be aligned with the architectural design. During the design stages these boundary conditions change occasionally, and having a clear up-to-date overview of the design is therefore crucial.

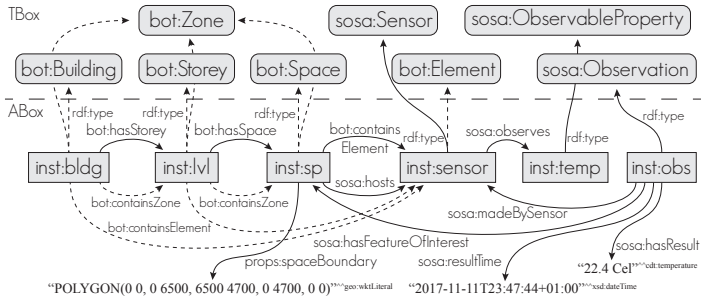


Fig. 1: The overall data structure. Dashed arrows indicate inferred knowledge.

Establishing a Linked Building Data (LBD) compliant architectural model from the proprietary BIM format was achieved by using the Revit-BOT-exporter[5] described in [8]. 2D space boundaries were exported by implementing a WKT polygon parser implemented in the visual programming environment, Dynamo for Revit. WKT is compliant with geoSPARQL [1] - a SPARQL Protocol and RDF Query Language (SPARQL) for geographic data. This allows for including region connection calculus in queries such as finding anything located within the boundaries of a polygon.

Units are described using the CDT Datatypes that leverage the Unified Code of Units of Measures UCUM [4].

Listing 1: Subset of Architect's model

```
# BUILDING TOPOLOGY (MODELLED BY ARCHITECT)
inst:level_57d0ded0-4341-4dba-8f32-8dbdcaa9877c-0004879d a bot:Storey ;
   bot:hasSpace inst:room_4b80808e-2f04-46a0-b84d-0ad6ee9d6b1b-0012a494 .
inst:room_4b80808e-2f04-46a0-b84d-0ad6ee9d6b1b-0012a494 a bot:Space ;
   props:identityDataNumber "04.196" ;
   props:dimensionsArea "13.78 m2"^^cdt:area ;
   props:identityDataName "Gr. rum 04.196" ;
   props:spaceBoundary "POLYGON((-3319 14852, -8040 16954, -8226 17037, -8077 13710,
      -4529 12131, -3319 14852))"^^geo:wktLiteral .
```

Since the sensor data was already available (Sec. 3), establishing a SSN/SOSA compliant dataset with mappings to the architectural spaces was just a matter of writing a parser. The mapping table between Uniform Resource Identifiers (URI) of architectural spaces and their room number was created from a simple SPARQL query returning all bot:Space instances and their props:identityData-Number. Listing 2 shows an example of the output. In the example, the dog:TemperatureSensor is used to specify that it is a temperature sensor. An alternative solution to determining the kind of sensor could be to use a generic property `inst:Temperature` instead of the location-specific `inst:room_04.196-Temp` like illustrated in Fig. 1.

[5] https://github.com/MadsHolten/revit-bot-exporter

Listing 2: Sensor and property data

```
# SENSOR AND PROPERTY (MODELLED BY ENGINEER)
inst:room_4b80808e-2f04-46a0-b84d-0ad6ee9d6b1b-0012a494
   bot:containsElement inst:room_04.196-Temp-Sensor .
inst:room_04.196-Temp-Sensor a sosa:Sensor , dog:TemperatureSensor ;
   sosa:observes inst:room_04.196-Temp .
inst:room_04.196-Temp a sosa:ObservableProperty .
# OBSERVATION (OUTPUT FROM BMS)
inst:room_04.196-Temp-obs0 a sosa:Observation ;
   sosa:hasFeatureOfInterest inst:room_4b80808e-2f04-46a0-b84d-0ad6ee9d6b1b-0012a494 ;
   sosa:hasResult "22.8 Cel"^^cdt:temperature ;
   sosa:madeBySensor inst:room_04.196-Temp-Sensor ;
   sosa:observedProperty inst:room_04.196-Temp ;
   sosa:resultTime "2017-09-16T16:21:54+01:00"^^xsd:dateTime .
```

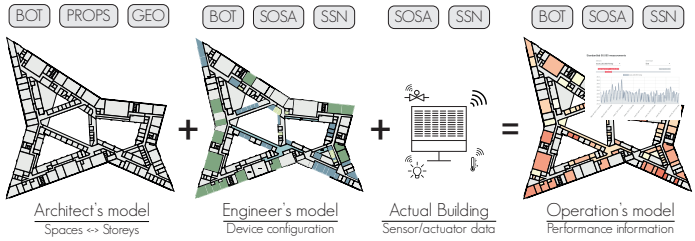


Fig. 2: Integration of the different datasets.

In an LBD mediated integrated workflow, the mapping between the architectural spaces and the sensors and their observed properties could be part of the project delivery of the low voltage engineer. Such a workflow could look like the one illustrated in Fig. 2. Based on the architectural model, the engineer defines templates for how the different space types should be equipped with sensors and actuators and potentially what control strategy to use. In a web application (Fig. 3) the engineer defines and assigns these equipment templates to each space, and the graph is extended with sensor and/or actuator instances (Lst. 2).

When following this workflow, sensor URIs exist in the building model prior to the installation phase. With correct mappings between the actual sensors and their digital twins, the semantics are already established when observation logs become available. Observations can therefore be interpreted instantly - even for third-party applications.

Part of the work presented in this paper is the development of a simple application that integrates the three datasets illustrated in Fig. 2. The application first queries all the instances of bot:Storey that bot:hasSpaces which have a props:spaceBoundary assigned. These populate a drop-down list from where the user can select a specific level. When choosing a level, the WKT polygons are retrieved, parsed to geoJSON (OGC) and rendered as a 2D Scalable Vector Graphics (SVG) plan. In parallel, a query for bot:containsElement relationships to sosa:Sensor instances and their sosa:Observations grouped by bot:Space instances is executed to get the maximum temperature in each space. The results are translated to a color grade, which is appended to the 2D plan (Fig. 4).

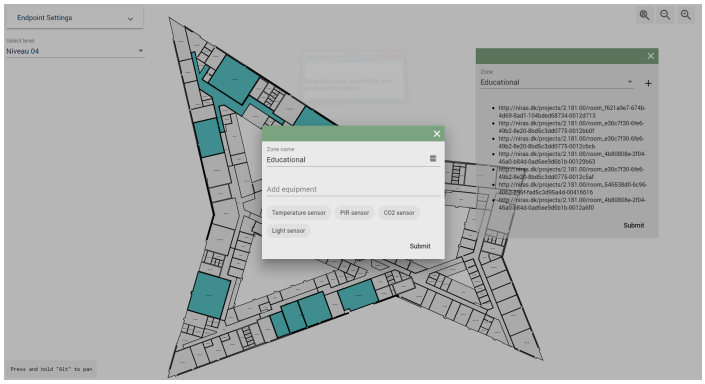


Fig. 3: Application for assigning equipment templates to architectural spaces.

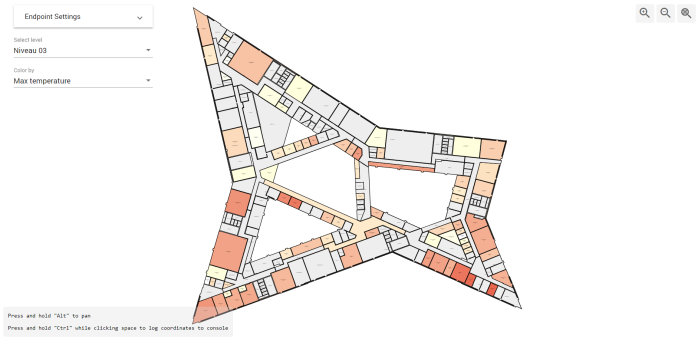


Fig. 4: Plan drawing shown in the web-app. Colors indicate max temperature.

When clicking a space, a line chart view of the sensor data is presented (Fig. 5). A drop-down list is populated with the dcterms:identifier of each sensor and when selecting from this list all the available observations are retrieved and visualized. A slider allows the user to restrict the time range of the observations.

5 Discussion

The illustrated workflow shows how a BIM model can be enriched with sensors and actuators described with SSN/SOSA. In this work, the sensors and actuators were related to the building in which they operate using BOT semantics, but they could additionally be described in the context of the systems on which they operate. These opportunities bring a new incentive for the engineer to engage in BIM, which is often mistakenly comprehended as only 3D models. Establishing a semantic model of a BMS

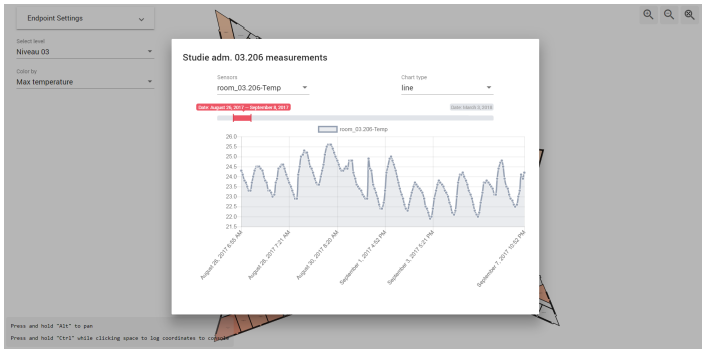


Fig. 5: Illustrate measurements for a given time range.

in the design stages and relating it to the features of interest on which they operate will further provide documentation which is crucial for the overall design overview.

Having the semantics of the BMS available in an open format when the building is put into operation allows for interpreting the observations of the sensors out of the box without the need for an integrated BMS solution. This interpretation separates the devices from the software applications and marks the first step in democratizing the market for BMS. It enables building owners to freely choose devices without being tied to one particular manufacturer for the full life cycle of the building and further makes it possible for a new industry to arise as universal, versatile software solutions can be developed.

Designing systems for building automation typically undergoes several stages. Initially, an Indoor Climate and Energy (ICE) engineer simulates the spaces - often only the critical ones regarding internal and solar heat gains, but in some cases also the whole building. When doing such simulations, a control strategy for heating, cooling, and ventilation is applied, and this should be reflected in the actual systems of the building. The capacity of the systems used in the simulation should match the ones described by the HVAC engineer, and the control strategy should be reflected in the description of the low voltage engineer. Installed systems in the building must further be programmed in order to comply with these specifications. The physical design of the spaces often change during the design stages, and this might influence the technical systems. It is therefore crucial that changes are carried on all the way from the ICE engineer to the contractor. Being able to specify the control strategy in an explicit format could significantly reduce the risks in this supply chain.

The implementation consisted of a 20M triples graph of which the observations were the primary component. Some of the more resource intensive queries like getting the maximum temperature of all spaces at a storey took up to 3.5 seconds, thereby devoting the user experience slightly (query performed on local Stardog triplestore served on a Lenovo P50 laptop with Intel Core i7-6820HQ 2.70 GHz CPU and 32 GB 2133 MHz DDR ram). This could be solved by doing some pre-processing on the server to infer hourly, daily, weekly, monthly and annual maximum temperatures explicitly. Most queries, however, like getting all observations (5000) from a server ordered by time can be accomplished in less than 500 ms.

6 Conclusion

With this work, we present an integration between a building dataset described using proposed Linked Building Data (LBD) ontologies and an SSN/SOSA compliant dataset with sensor and actuator observations. Sensors and actuators are typically not part of the BIM model as it provides only little profit for the overall project. With the showcased integration between the BIM model and the observations, however, there is an incentive for the engineer to model sensors and actuators conceptually. Dedicated tools for assisting in modeling the sensors and actuators in their context of the building, the control strategies, thermal simulations etc. is a future research topic of interest.

The simple demo application serves as a proof of concept for integrating data from different sources in a web of data based viewer application and although the functionality is limited it showcases the potential.

Acknowledgements

Special thanks to the NIRAS ALECTIA Foundation and Innovation Fund Denmark for funding and to Michael K. Therkildsen, Aarhus University, for providing the Navitas dataset.

References

1. Robert Battle and Dave Kolas. Enabling the geospatial semantic web with parliament and geosparql. *Semantic Web*, 3(4):355–370, 2012.
2. Charles M Eastman, Chuck Eastman, Paul Teicholz, and Rafael Sacks. *BIM handbook: A guide to building information modeling for owners, managers, designers, engineers and contractors*. John Wiley & Sons, 2011.
3. Armin Haller, Krzysztof Janowicz, Simon J D Cox, Danh Le Phuoc, Kerry Taylor, and Maxime Lefrançois. Semantic Sensor Network Ontology. W3C and OGC Recommendation, W3C & OGC, October 19 2017.
4. Maxime Lefrançois and Antoine Zimmermann. The unified code for units of measure in RDF: cdt:ucum and other UCUM datatypes. In *Extended Semantic Web Conference*, 2018. Demonstration paper.
5. Thomas Liebich and Jeffrey Wix. Highlights of the development process of industry foundation classes. In *Proceedings of the 1999 CIB W78 Conference*, 1999.
6. Pieter Pauwels and Anna Roxin. SimpleBIM : From full ifcOWL graphs to simplified building graphs. In *ECPPM2016*, number October, 2016.
7. Pieter Pauwels and Walter Terkaj. EXPRESS to OWL for construction industry: Towards a recommendable and usable ifcOWL ontology. *Automation in Construction*, 63:100–133, 2016.
8. Mads Holten Rasmussen, Christian Anker Hviid, and Jan Karlshø. Web-based topology queries on a bim model. In *LDAC2017–5th Linked Data in Architecture and Construction Workshop*, 2017.
9. Mads Holten Rasmussen, Pieter Pauwels, Christian Anker Hviid, and Jan Karlshoj. Proposing a central aec ontology that allows for domain specific extensions. In *Joint Conference on Computing in Construction*, volume 1, pages 237–244, 2017.
10. Alan Redmond, Alan Hore, Mustafa Alshawi, and Roger West. Exploring how information exchanges can be enhanced through cloud bim. *Automation in construction*, 24:175–183, 2012.

6.7 BOT3: The BOT Ontology: Standards Within a Decentralised Web-based AEC Industry

Under review. Submitted to Automation in Construction on April 4th 2018.

This paper describes a vision of having various interconnected Knowledge Graphs (KGs) described with BOT and other LBD ontologies. It uses BOT as a reference ontology and documents its latest developments as well as linking principles and methods for generating BOT-compliant data (LBD datasets). Through two use-cases it is demonstrated why a (semantic) web-based BIM can help bringing down the silos in Today's BIM implementations by allowing the dataset to be distributed and extended. Three BIM models are converted to LBD datasets and reasoning performance as well as file sizes is evaluated for these models. These are further used in a proof of concept where the decentralisation benefits are demonstrated by showing how the architect's dataset can be extended by Indoor Climate and Energy (ICE) and Heating, Ventilation and Air Conditioning (HVAC) engineers using simple queries to deduce new data.

The BOT Ontology: Towards a decentralised web-based AEC industry based on Linked Building Data

Mads Holten Rasmussen[a,*], Pieter Pauwels[b], Christian Anker Hviid[a], Jan Karlshøj[a]

[a] *Technical University of Denmark, Department of Civil Engineering, Denmark*
[b] *Ghent University, Department of Architecture and Urban Planning, Belgium*

Abstract

Research in the AEC industry has provided us with a good deal of ontologies by now. Although domain ontologies are available with a very broad scope, such as the Industry Foundation Classes (IFC), practice often sees much smaller and detailed ontologies. The large and overarching ontologies then become distant from those practice-oriented ontologies, resulting in calls for simple, modular, and extensible ontologies instead. In this article, we look into earlier work that provides such ontologies for the domain of Architecture, Engineering, and Construction (AEC), often starting from IFC. We hereby limit to ontologies that are defined using semantic web technologies. From this brief literature review, we define and outline the BOT ontology in this paper, including its recent changes, and we pinpoint how this ontology can respond to that call for simplicity, modularity, and extensibility, in combination with many of those newly emerging ontologies. We briefly document how existing data can be made available using this combination. We take geometry into account, but only to a limited extent, and thus focus more on the proposal of combining multiple modular ontologies in general. Furthermore, this article investigates two practical use cases and a proof of concept (PoC) implementation for capturing building data. Performance in a real-world setting is investigated, thereby looking at model size, web server implementation, and query performance. This performance investigation shows the feasibility of using this approach as an alternative means for collaborating with building data over the web, even though more detailed performance analyses should still be made, comparing to existing benchmark studies. Finally, this paper concludes to what extent a fully web-based approach for building information handling (BIM Level 3) can be built based on the proposed network of simple, extensible, and modular domain ontologies.

Keywords: Linked Data, Building Information Modelling, Ontologies, Building Topology Ontology

1. Introduction

Operating in the fragmented Architecture, Engineering and Construction (AEC) industry is a complex task. Temporary project organisations are established around each project, and changing stakeholders from project to project makes it hard to carry through a global optimisation of the project planning and execution. A vast amount of data is generated and processed, and pieces of information need to be carried on to other project participants in other parts of the organisation, potentially originating from another company (Bertelsen, 2003). The information supply chain needs to be reestablished from near scratch with each new project organisation, and, as the information exchanges are handled in a predominantly manual manner (file-based common data environment (CDE) at best), this task is often inefficient and error-prone.

In order to enhance interdisciplinary communication, a common language and understanding of the information exchanged is crucial. Building Information Modelling (BIM) has been introduced, in combination with the Industry Foundation Classes[1] (IFC), to overcome this task. BIM is a set of methodologies and technologies that should ultimately mediate a better information flow between

*Corresponding author
Email address: mhoras@byg.dtu.dk (Mads Holten Rasmussen)

[1] http://www.buildingsmart-tech.org/specifications

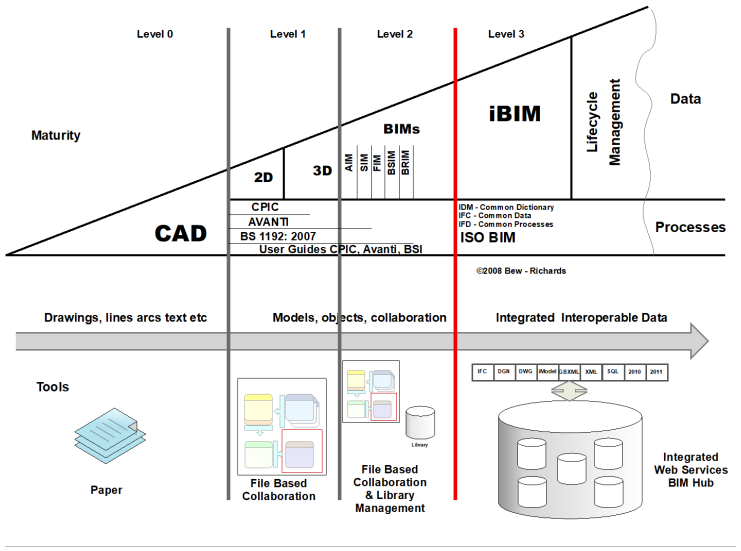


Figure 1: BIM Levels of Maturity, with the web-based BIM Level 3 on the far right (copyrighted image: Bew and Richards (2008).

the project participants and hence provide access to relevant information for solving any given task. Most of the interoperability, however, comes from the use of a standard common language, which is exactly what IFC aims to provide. Hence, currently, building information can be modelled in BIM authoring tools, and a common exchange format is available that allows to exchange the data and improve interoperability. In practice, IFC often serves as a file exchange mechanism that takes part in the overall BIM exchange process of a design and construction project.

A number of maturity levels have been defined to indicate to what extent the above is successfully implemented in companies. These BIM maturity levels are based on the 'wedge' diagram given in Figure 1, which forms an important part of the PAS 1192 Specification of BSI Standards Limited (2013). The majority of the AEC market, if not the entire AEC market, is situated in Level 0, 1, or 2 of this diagram, which are all document-based ways of working (exchanging whole documents and files), as opposed to the far right web-based BIM Level 3. BIM Level 3 implies 'integrated interoperable data' and an 'integrated web services BIM hub' based on standards (ISO BIM). Other than that, BIM Level 3 is relatively undefined, yet it does imply the use of web technologies, standards, and one integrated hub.

To enable this web-based way of working (BIM Level 3), the BIM market needs to follow the same evolution that is happening in the world wide web (www) in moving from a web of documents to a web of data (Bizer et al., 2009). In other words, software and standards in the AEC industry need to be made web-compliant. Industry practitioners actively work towards that direction, and, as a result, different open source community-based software projects have evolved in the past years (eg. Flux.io[2], vA3C[3] and speckle.works[4]). These are aiming at enabling direct information exchanges, mainly concerning geometry, between native Computer Aided Design (CAD) and BIM software using web Application Programming Interfaces (APIs).

In terms of making standards for BIM information exchanges web-compliant, earlier work was done to transform the IFC EXPRESS schema into an OWL (Web Ontology Language[5]) ontology (ifcOWL (Pauwels and Terkaj, 2016)), as this would

[2]Discontinued, no longer online
[3]https://va3c.github.io/
[4]https://speckle.works/
[5]https://www.w3.org/TR/owl2-overview/

allow building a web of building data using (semantic) web technologies. However, the resulting ifcOWL ontology is directly derived from EXPRESS, making it different from many of the available ontologies and hence not so easily accessible or usable using Semantic Web technologies. Most of the drawbacks of ifcOWL are directly related to this dependency on IFC EXPRESS, making the result large in size, complex as a data model, non-modular, and not easily extensible (Pauwels et al., 2018; Schneider et al., 2018).

Other initiatives have similarly aimed at combining IFC with web technologies (Beetz et al., 2014; Liu et al., 2017; Gao et al., 2017, 2015). In many such cases, however, the IFC file-based exchange mechanism still prevails, and the challenge of obtaining a simple, modular, and extensible data model remains standing. Alternatively, a number of efforts looked directly in providing a simplified view on top of the ifcOWL ontology, thereby staying completely in a web environment of linked data (Mendes de Farias et al., 2015; Pauwels and Roxin, 2016). These approaches have been proven successful, yet they do not have any ontology defined and instead perform graph simplification on demand.

The challenge of having a simplified, modular, extensible approach that effectively includes an ontology was more directly addressed with the minimal and extensible Building Topology Ontology (BOT). BOT was initially presented in (Rasmussen et al., 2017b) and is recommended by the World Wide Web Consortium (W3C) Linked Building Data Community Group (W3C LBD CG)[6] who have continuously extended and refined the concepts and interlinked them with other efforts of the group.. Later additions were documented in (Rasmussen et al., 2017c), but, until now, an in-depth walk-through of its concepts, extendability and capabilities as part of a full LBD approach (BIM Level 3) has been missing, which is the main target of the current article.

In this article, we will first give a brief overview of the state of the art in moving the AEC industry in the direction of the web of data (Sec. 2). Then, we will go through the design of BOT, thereby giving brief examples and indications of how the ontology can be combined with other ontologies in support of specific use cases (Sec. 3). By using an exporter for the BIM authoring tool Revit, three LBD datasets of native BIM models are generated and the exporter's performance along with the content of the exported data is evaluated. Exports from one of the models (the Common BIM file Duplex Apartment[7]) have been made available in an open Github repository[8]. Section 4 indicates BOT linking methods to illustrate two use cases, which are documented in small datasets that can be studied and engaged with in an interactive web application[9]. We encourage the reader to test the examples while reading this. After showcasing the concepts in the two use cases of Section 4, the second case is applied in a proof of concept (PoC) alternative BIM object modelling approach where new objects are inferred by performing queries on the objects and properties that are already available (Sec. 5). Besides showing an approach to operate a decentralised data model, this experiment is a test case for validating query performance on large datasets. The workflow can be reviewed in a second interactive web application[10]. Finally, we conclude in Section 6 on how this alternative way of modelling and publishing data in the AEC sector can shift the industry away from document-based file exchanges towards web-based collaboration through interlinked data models.

2. State of the Art

A full overview of semantic web technologies and ontologies[11]. in the AEC domain is available in Pauwels et al. (2017b). Hence, we will not go in full detail in this section, but instead limit to a state of the art on extending IFC using semantic web technologies and on key reference ontologies.

2.1. Extending IFC into the web

The standard schema for the exchange of BIM data is IFC (ISO 16739:2013), which is a data model described in EXPRESS and which has a strong focus on the representation of 3D geometry (Pauwels et al., 2018). Several research projects have dealt with extending IFC using semantic web technologies. For example, Beetz et al. (2014) describe a

[6] `https://www.w3.org/community/lbd/`

[7] `https://www.nibs.org/page/bsa_commonbimfiles#project1`

[8] `https://github.com/MadsHolten/BOT-Duplex-house/tree/v1.0`

[9] `http://www.student.dtu.dk/~mhoras/ac/use-cases/`

[10] `http://www.student.dtu.dk/~mhoras/ac/proof-of-concept/`

[11] An ontology is defined as *"a formal, explicit specification of a shared conceptualization"* (Studer et al., 1998)

method for extending an IFC model towards external RDF graphs. This allows a transitional approach in which the core of IFC, and in particular the geometry, can still be used, while also allowing to link to external vocabularies at no additional cost. Indeed, a full shift to RDF demands a considerable shift in technologies. This approach thus addresses the extensibility issues of IFC.

Furthermore, (Liu et al., 2017; Gao et al., 2015, 2017) aim to use the IFC data model in combination with web technologies for the purpose of improved information retrieval. In this case, a domain ontology of IFC for information retrieval (IFC-IR) is suggested (Gao et al., 2015), which is then used to annotate online resources for improved information retrieval (Liu et al., 2017; Gao et al., 2017). As a result, IFC data can more easily be retrieved using semantic queries.

2.2. Ontologies in the AEC domain

In addition to the extension mechanisms mentioned above, a number of distinct ontologies have been proposed before, aiming explicitly to bring building data to the web.

ifcOWL. The ifcOWL ontology[12] by Pauwels and Terkaj (2016) was first proposed by Beetz et al. (2008). The ontology is a conversion of the IFC schema to a version described in the Web Ontology Language (OWL). Since IFC includes not only the terminology required to describe the building itself, but also its systems and data related to describing time scheduling, cost estimation and quantitative units, the latest version (IFC4_ADD2) consists of an overwhelming number of classes and properties (1331 classes and 1599 properties)[13]. Ongoing work aims at extending IFC towards roads (Lee and Kim, 2011) and bridges (Yabuki et al., 2006), which makes the result even bigger and more complex, and hence even more difficult to use.

The data structure of ifcOWL is also influenced by its origin in the EXPRESS schema. Maintaining the structure of its origin makes it backwards compatible, but it adds an extra layer of complexity. This makes the ontology hard to manage, hard to understand, and hard to query. For the purpose of a simple building representation, one may argue that the IFC schema is too extensive, and for the same reason, it has been proposed to provide a simplified version of the ifcOWL ontology: IFCWoD (IFC Web of Data) (Mendes de Farias et al., 2015) and SimpleBIM (Pauwels and Roxin, 2016). Both approaches cut away elements like geometric data and intermediate EXPRESS-derived relation instances between objects. Both approaches lack an individual ontology, but rather provide an approach to post-process an ifcOWL-compliant Resource Description Framework (RDF[14]) dataset with a more simplified representation. In order to meet the best practice recommendations of modularisation, Terkaj and Pauwels (2017) have later suggested an approach to generate a modular version of ifcOWL, based on the modules that are present at the core of IFC. This approach thus addresses the modularity issues of IFC. Also in this case, however, the result stays relatively close to the EXPRESS version of IFC.

BIMSO/BIMDO. Another approach that has been suggested is the BIM Shared Ontology (BIMSO), which is a foundation ontology for the AEC and Facility Management (FM) industry with the purpose of being extended with various building domain ontologies (Niknam and Karshenas, 2017). At the overall level, the ontology has only a few classes and relationships scoped at describing a building's elements, levels, spaces and construction phases, but it relies on the full Uniformat II classification system for further organising the elements, meaning that it ends up being a rather large and overwhelming ontology as well. Furthermore, it lacks the necessary object properties (owl:ObjectProperty) to describe relationships between elements, subdivision of zones and to quantify these relationships. These are however partly covered by a separate ontology, the BIM Design Ontology (BIMDO), which is also covered in (Niknam and Karshenas, 2017). Both the BIMSO and BIMDO ontologies are not publicly available, which makes them essentially unusable.

W3C LBD CG. In most research projects covering subsets of the AEC domain, the same concepts of a building are unfortunately contradictorily redefined repeatedly (Rasmussen et al., 2017b). These include, for example, the BIMSO, the Smart Appliances REFerence (SAREF) ontology (Daniele et al., 2015), DogOnt (Bonino and Corno, 2008),

[12] http://www.buildingsmart-tech.org/ifcOWL/IFC4#

[13] http://www.visualdataweb.de/webvowl/#iri=http://ifcowl.openbimstandards.org/IFC4_ADD2.ttl

[14] https://www.w3.org/TR/rdf11-primer/

ThinkHome (Reinisch et al., 2011) and Brick (Balaji et al., 2016). As an example, SAREF4BLDG includes a subschema to capture the structure of a building. This subschema is then in this case extended with appliance and device data, similar to how BIMSO comes with uniformat entirely included. This makes it difficult to build a really modular set of ontologies that allows capturing building data in a stepwise extensible manner.

Part of the W3C LBD CG work has been to establish a minimal ontology that describes the concepts related to a building that are redundantly repeated in these ontologies, and the result is the Building Topology Ontology (BOT) described in this work. The scope of BOT is to explicitly define necessary relationships between the sub-components of a building. As such, it aims to provide the means for representing interlinked information in a future (semantic) web driven AEC industry, satisfying the recommendation of reusing terms already described in well-known vocabularies wherever possible (Bizer et al., 2009). Other work of the group includes the three following ontologies.

- a *product* ontology for describing building products:
  A PRODUCT ontology is available in a modular structure in itself[15]. This product ontology includes a Product class that can include itself (decomposition). Furthermore, a large number of subclasses are made available to capture classes such as a wall, beam, flow terminal, and so forth. These subclasses are inspired by the building element classes that are present in IFC.
- a *props* ontology for describing properties:
  A PROPS ontology is tentatively made available in the W3C LBD CG pages[16]. This ontology captures the diverse properties that products may have, with classes of the PRODUCT ontology as their associated domains. This ontology is in its current form inspired by the property sets made available as part of the IFC standard.
- an Ontology for Property Management (*OPM*):
  OPM is an ontology for describing property states, thereby allowing property values to evolve over time while keeping track of their history (Rasmussen et al., 2018b). This ontology uses terminology from the Smart Energy-Aware Systems ontology (SEAS) and the Provenance Ontology (PROV-O) (Lefranois et al., 2017; Lebo et al., 2013) and encourages the use of schema.org properties for describing quantitative values[17] (a quantified value with a unit).

3. The Building Topology Ontology (BOT)

In this section, we describe the BOT terminology in detail, after which we evaluate how it can link to other ontologies available in the domain. We then show how to attach 3D geometry to any BOT resource and describe in detail how to establish LBD-compliant datasets. [18]

3.1. BOT as a reference ontology

BOT is designed to provide the means to describe a building element (tangible object), or zone (spatial 3D-division) in its topological context of the building in which it exists. Its classes can be used not only for existing buildings but can also be used to create an abstract requirements model of a future building. An example approach for this is described by Rasmussen et al. (2018a), who defines sub-classes of bot:Space specifying the client's requirements for spaces of a future building. The ontology is designed as a minimal schema only defining the core terminology necessary to describe building topology. It was first presented in (Rasmussen et al., 2017b) and later extended in (Rasmussen et al., 2017c). Since then it has been further extended to accommodate modelling demands that could not be captured by the existing terminology. These demands include handling

[15] https://github.com/w3c-lbd-cg/product
[16] https://github.com/w3c-lbd-cg/props
[17] Full definition at https://schema.org/QuantitativeValue
[18] Throughout the remainder of the paper, we use namespace prefixes when referring to classes and properties that are defined in ontologies. bot:Element for example describes the Element class in the BOT namespace. The full Uniform Resource Identifier (URI) is embedded as a hyperlink for the namespaces that expose the ontology publicly online. Prefix inst: is used to describe instances that belong to the data layer.

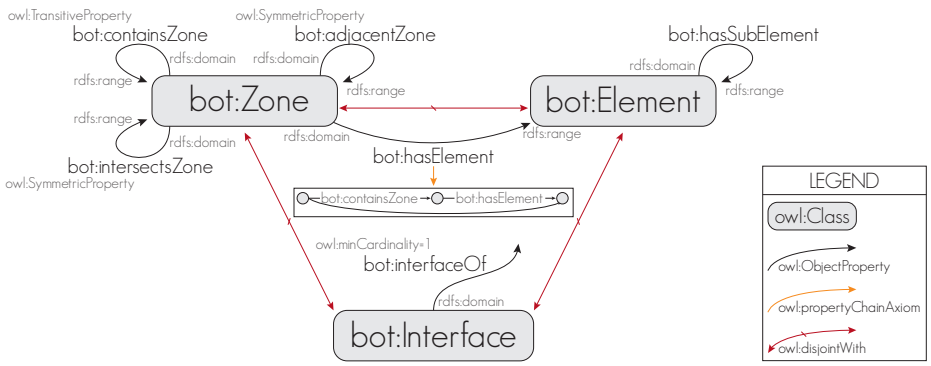


Figure 2: The Building Topology Ontology overall terminology

of zones intersecting with multiple storeys (staircases), inheritance of space bounding elements to the building in which the spaces are contained, and attachment of 3D geometry. In this section, we will give an in-depth overview of what information can be modelled using BOT terminology (Figure 3). Overall it can be summarized to:

- A building consists of zones and building elements.
- Subzones include, but are not limited to sites, buildings, storeys and spaces.
- A zone can "have" elements.
- Subproperties of this generic zone-element relationship include adjacent elements, contained elements and elements that intersect with the zone but are not fully contained.
- A zone can contain other zones, intersect with other zones and be adjacent to other zones.
- A building element can host sub-elements.
- Interfaces between zone/zone, zone/element or element/element are quantifiable.

Relationships. Relationships between zones and elements are all described using direct object properties, whereas ifcOWL and other ontologies have an intermediate object in-between. Pauwels and Roxin (2016) uses a similar approach in simplifying an ifcOWL-based dataset with the argument that intermediate `IfcRelAggregates` instances are adding unnecessary complexity making the dataset hard to query. Having the intermediate step allows for adding metadata to the relationship and for some cases this is indeed very practical. Rasmussen et al. (2018b) describe property assignment at three levels where the level describes the distance between the subject and the property value and each level allows for adding more metadata about the property. For BOT, simplicity is key and therefore the most direct approach is used. bot:Interfaces can be used for quantifying a relationship in cases where this is necessary.

Main classes. At an overall level, BOT has three main classes: bot:Element, bot:Zone and bot:Interface. A bot:Element is any tangible object (product, device, construction element etc.) that exists in the context of a building. A bot:Zone is spatial 3D division (ie. building, space, thermal zone, fire cell) or a subdivision or an aggregation of such divisions. A bot:Interface is used for quantification of the relationships between instances of the two other classes. The high level terminology of the ontology is illustrated in Figure 2, and the more detailed terminology is illustrated in Figure 3.

Zones and Elements. In addition to the three main classes, 4 predefined subclasses of bot:Zone are included: bot:Site, bot:Building, bot:Storey and bot:Space. A zone can contain other zones (Figure 4), defined by the bot:containsZone relationship, which is a transitive property meaning that:

$$\forall x, y, z : containsZone(x, y) \wedge containsZone(y, z) \rightarrow containsZone(x, z)$$

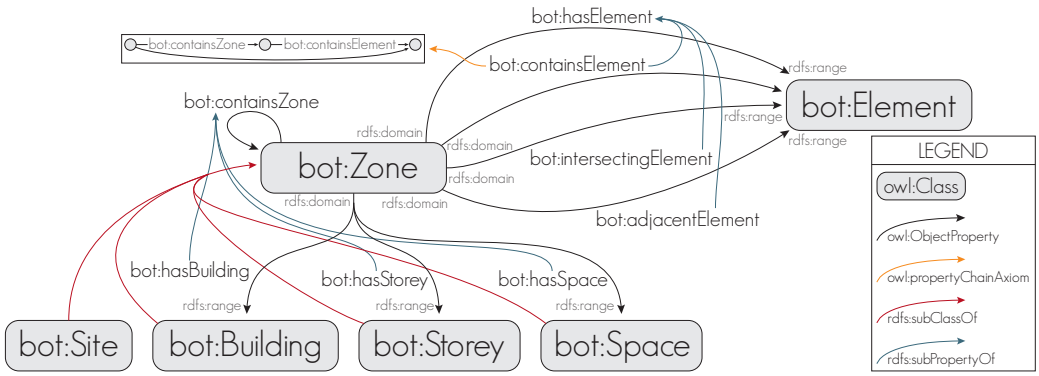


Figure 3: The Building Topology Ontology sub classes and sub-properties

More specific properties (bot:hasBuilding, bot:hasStorey and bot:hasSpace) are all defined as sub-properties of bot:containsZone meaning that:

$$\forall x, y : hasBuilding(x, y) \vee hasStorey(x, y) \vee hasSpace(x, y) \rightarrow containsZone(x, y)$$

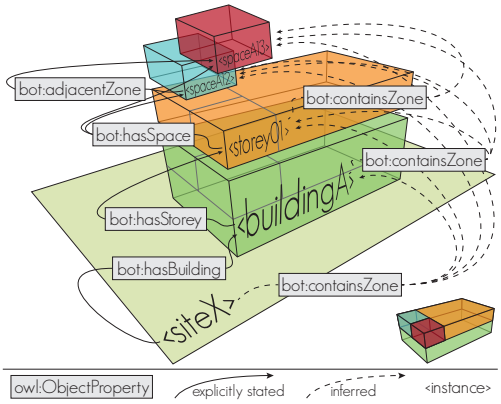


Figure 4: Zones in BOT follow a Matryoshka doll principle where one zone can be contained within another zone and so forth (Rasmussen et al., 2017c).

The ranges of these more specific properties are instances of bot:Building, bot:Storey and bot:Space respectively, meaning that anything having for example the bot:hasBuilding relationship assigned is itself an instance of bot:Building. Besides from containing other zones, adjacencies between zones can be specified using the bot:adjacentZone property and two zones that partly occupy the same space can be specified using the bot:intersectsZone property. These properties are symmetric properties meaning that:

$$\forall x, y : adjacentZone(x, y) \rightarrow adjacentZone(y, x)$$
$$\forall a, b : intersectsZone(a, b) \rightarrow intersectsZone(b, a)$$

Relationships between zones and elements can be described using the bot:hasElement relationship. The intended use of this term is however not to be specified explicitly, but to be inferred from its sub-properties in order to provide a mechanism for achieving all elements having some relationship to a zone. bot:hasElement has a property chain axiom providing a mechanism for also inheriting elements that have some relationship to zones contained inside the zone:

$$\forall x, y, z : containsZone(x, y) \wedge hasElement(y, z) \rightarrow hasElement(x, z)$$

bot:Element is in most cases too generic to describe a building element. With BOT alone it is not possible to distinguish between windows, walls, ducts or defibrillators in a building as they all belong to the bot:Element class. The main reason for not having these specific products in BOT is that the list would be large and forever growing as new products arise (cfr. simplicity, complexity, extensibility). The PRODUCT ontologies describe

specific product classes, and together with the PROPS ontology, relevant properties can also be assigned to each specific product. A product is not a bot:Element until it exists in a topological context of a building, and the rdfs:range of the bot:hasElement relationship takes care of inferring this knowledge. This mechanism is illustrated in Figure 5. At the top, a subset of the terminology layer (TBox) is illustrated, and at the bottom, some sample instances are represented in the assertion layer (ABox). The TBox must be seen in combination with Figure 2 and 3 to get the full set of axioms required to infer the implicit relationships.

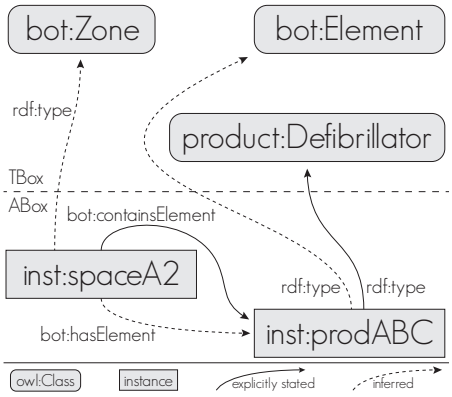


Figure 5: Using zone-element relationships to infer the bot:Element class.

More specific zone-element relationships (subproperties of bot:hasElement) include adjacency to surrounding elements, which is specified by the bot:adjacentElement relationship, intersecting elements specified by the bot:intersectingElement relationship and containment defined by bot:containsElement which is a property chain axiom of bot:containsZone and bot:containsElement meaning that:

$$\forall x, y, z : containsZone(x, y) \wedge containsElement(y, z) \rightarrow containsElement(x, z)$$

Elements can have sub-elements defined by a bot:hasSubElement relationship from the superelement. This can, for instance, be used to specify a relationship between a window and the wall in which it is hosted.

Interfaces. Interfaces are used to quantify the relation between zones and elements (Figure 6). Three practical use cases of interfaces could be for quantification of (1) the heat transmission area of the surface between a space and an adjacent wall, (2) the interface between a pipe and a wall or (3) the access between two zones. (1) can be used to determine the heat loss from a space through the particular wall, (2) to specify where to apply fire sealing and (3) to specify access restrictions for use in indoor navigation. An interface is assigned to the elements or zones using the bot:interfaceOf property, where the domain is always a bot:Interface and the range can be anything.

At least one bot:hasInterface relationship must be specified for each interface, and in most situations, an interface allows to add further detail to the relationship between two objects. There exists no restriction for the maximum number of bot:hasInterface relations, and for (1) it might be a good idea to let the interface define the relationship between three objects: The wall which it represents and each of the two thermal environments (the space and outside) it shares a surface with (Figure 6).

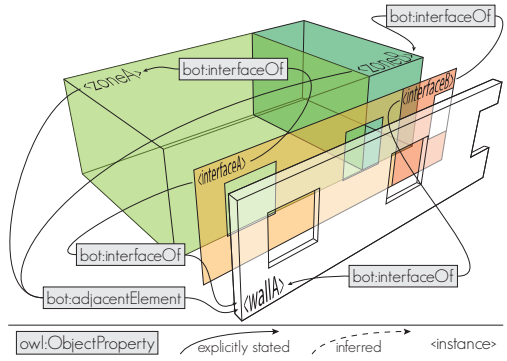


Figure 6: Interfaces can be used to quantify heat transmission areas (Rasmussen et al., 2017c).

3.2. Linking to BOT

There are several strategies for linking separate ontologies together, and they can be applied interchangeably depending on the desired outcome. BOT is designed to function as a central element

in the interdisciplinary communication of the AEC sector, and in this section, we will illustrate a set of approaches to use BOT as a link between domain knowledge rooted in domain ontologies. Schneider (2017); Schneider et al. (2018) uses some of these approaches to align BOT with a set of ontologies commonly used in the construction domain of which a subset was covered in Section 2. For illustration purposes, fictive namespaces of non-existing ontologies are used in the examples.

Declaration of sub-classes. For some purposes, the bot:Element, bot:Zone and bot:Interface classes and even the more specific subclasses such as bot:Space are too generic. Therefore, it is essential to be able to extend BOT with more specific classes. Figure 7 illustrates one approach to extending BOT by adding a `fso:Heater` from a fictive Flow Systems Ontology. rdfs:subClassOf states that:

$$\forall x, y, z : \ type(x, y) \ \wedge \ subClassOf(y, z) \rightarrow type(x, z)$$

Hence, it is automatically inferred by a reasoner that if `inst:heater33` is of type `fso:Heater`, then it is also of type bot:Element, thereby giving it a more generic abstraction understandable by other domains.

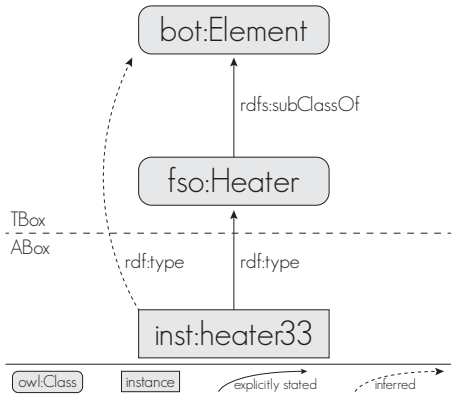


Figure 7: Linking by defining sub-classes.

Typed links. When using declaration of sub-classes as a linking approach, the dataset needs to specifically state that `inst:heater33` is of type `fso:Heater`, but this can alternatively be inferred automatically by a typed link between two ABox instances (data layer). This approach includes defining a sub-property of one of the more generic BOT properties. Figure 8 illustrates an approach where a new property `fso:heatedBy` is defined as a rdfs:subPropertyOf bot:containsElement, entailing that:

$$\forall x, y : \ heatedBy(x, y) \rightarrow containsElement(x, y)$$

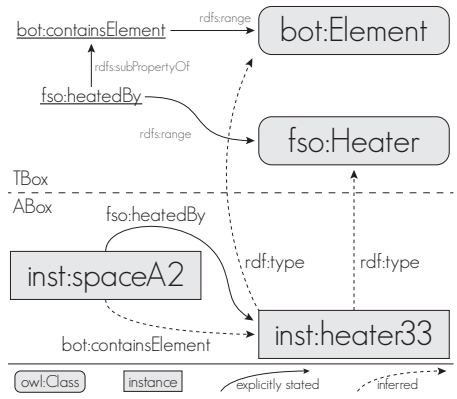


Figure 8: Linking by defining subClasses.

Since it is stated that the range of the `fso:heatedBy` property is a `fso:Heater`, it can automatically be deduced that `inst:heater33` is a `fso:Heater`. Because the inferred bot:containsElement property has range bot:Element, it can further be inferred that `inst:heater33` is also a bot:Element.

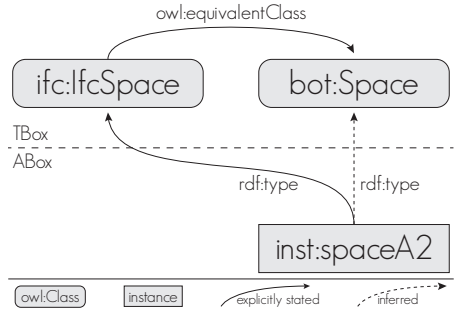


Figure 9: Linking by defining an equivalentClass.

Declaration of equivalent classes. In the last approach, direct equivalent class links are embedded in the ontologies. This approach must be used with caution, as such links state that two classes are completely equivalent. This means that any facts that *ontology A* has stated about the class will also be true for *ontology B*, which is not always desirable. In Figure 9, bot:Space is defined as an equivalent class to ifcOWL's ifcSpace. In an ifcOWL-compliant dataset, a reasoner would hence infer that all instances of `ifc:IfcSpace` are also instances of bot:Space. Note that this equivalency works in two directions. In other words, any bot:Space instance would load all definitions for an ifcOWL IfcSpace and append them to that bot:Space, leading to an undesired overhead of data.

3.3. BOT and geometry

Describing detailed geometry is not in the scope of BOT, but being able to relate some object to a 3D model does provide value from a topological point of view. Therefore, BOT includes properties to relate some object to a 3D model. How the model is encoded is not in the scope of BOT, but an indication of options based on the ifcOWL work can be found in Pauwels et al. (2017a). The simplest way to define a 3D model of some object is to describe it with the bot:hasSimple3DModel datatype property. The related literal can, for example, contain mesh geometry encoded as a Wavefront OBJ.

Alternatively, the bot:has3DModel property is an object property which points to a resource describing the geometry in more detail. This approach can be used if storing of metadata is necessary or if more 3D models (eg. different levels of detail) of the same object exist. The approach can also be used to give a more complex description of the geometry, including boundary representations and boolean operations. Bonsma et al. (2018) discusses different considerations for capturing such complex geometry, including references to the ontoBREP approach (Perzylo et al., 2015) and the ifcOWL approach. Since BIM model geometry is specified relative to the local coordinate system of the model, the site must have a zero point reference in order for the model to be used in a global (GIS) context. The zero point can be specified using the bot:hasZeroPoint property, which must point to a wgs84:Point.

As an example, simple mesh geometry originating from a BIM authoring tool can be attached to any bot:Element or bot:Zone instance. The meshes can provide valuable visual feedback for a user querying the data model (Figure 10). Further, it provides a snapshot of the data that can be used to compare model states using OPM, and more valuable semantics can be deduced from the meshes and potentially inferred in the data model.

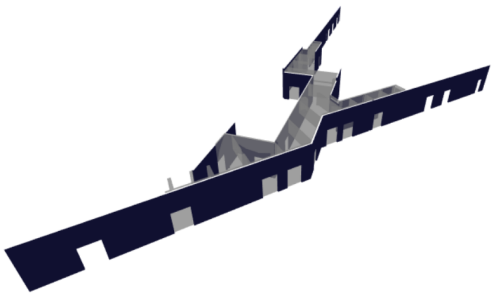


Figure 10: Example of a graphical feedback from a request for all wall elements adjacent to a particular space using BOT terminoligy. The 3D model consists of simple mesh geometry. Simple demo[19].

3.4. Generating BOT-compliant data

Data according to BOT and the other ontologies suggested by the W3C LBD CG (LBD datasets) can be generated in a number of ways (see also Bonduel et al. (2018a)). The simplest way is to simply load the ontology in memory (e.g. Jena API) and directly create instances of the classes and properties in BOT. This is of course also the most tedious way, as this would imply building an entirely new modelling application and using that one instead of the existing applications. Note, however, that such an approach is presented as the preferred option for modelling existing buildings by Bonduel et al. (2018b).

A second way is to convert existing IFC models into LBD datasets. This can be done using a Java-based IFCtoLBD converter[20] (Bonduel et al., 2018a). The tool uses concepts defined in the ontologies of the W3C LBD CG, namely BOT, PROPS, PRODUCT and OPM to describe the model. IFCtoLBD exports instances of bot:Site, bot:Building, bot:Storey, bot:Space and relationships between space instances and bot:adjacentElements and bot:containsElements.

[19] https://madsholten.github.io/ng-plan/index.html
[20] `https://github.com/jyrkioraskari/IFCtoLBD`

It further exports bot:hasSubElement-relationships between elements, but in its current version, bot:Interfaces and zone adjacency are not supported.

A third way is to convert native BIM model data directly into a LBD dataset, without passing through the IFC export mechanism. When aiming for a linked data based CDE (BIM Level 3), this approach is in fact desired, as it gives a view directly on the native data (single source). In this approach, a LBD dataset is generated using a plug-in[21] in the native environment of a BIM authoring tool (e.g. Revit) and uploaded to a web-based application that allows users to do SPARQL Protocol and RDF Query Language (SPARQL[22]) queries in order to traverse the graph based on the BOT relationships.

Rasmussen et al. (2017a) described a method for exporting building topology data to a triplestore while the building geometry was exported to a separate store with the commercial Forge API by Autodesk. In a simple web application, the authors showed a PoC of how topology queries could be used to filter the model view and provide table-based results of the queries. Later developments of the Revit exporter bring data exports complying with the full LBD toolset (PROPS, PRODUCT, OPM) along with the export of 3D models of spaces and elements as OBJ encoded mesh geometry. The mesh geometry, shown in Figure 10, is generated using the exporter plugin, and visualized in a web browser with an application built with the Web Graphics Library (WebGL).

Figure 12 shows the overall process of getting data from the BIM authoring tool to the triplestore from where the web application (Figure 10) reads the data. With this architecture, the web application is completely separated from the data, and it simply becomes a viewer that can combine data from various sources (i.e. a linked data based CDE). The view does not necessarily need to be read-only as it can do CRUD (Create, Read, Update and Delete) operations on the datasets using the Hypertext Transfer Protocol (HTTP) if the authentication system allows the user to do so. As such, the application serves as a PoC for a future decentralized CDE that organically grows a distributed dataset as the design progresses.

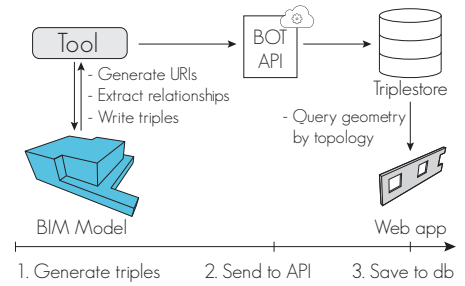


Figure 12: The infrastructure from triple extraction over the web API to pushing data to the triplestore.

A set of three native Revit BIM models (Figure 11) were converted to LBD datasets using the most recent version of the Revit BOT exporter. These models include the Common BIM file *Duplex Apartment* [490 m^2] and two models from finalized construction projects by the Danish consulting engineering company Niras[23]; *RTC*: a Technical College in Roskilde, Denmark [4,970 m^2], and *AU*: a university building (Navitas) at Aarhus Uni-

[21] https://github.com/MadsHolten/revit-bot-exporter
[22] https://www.w3.org/TR/sparql11-query/
[23] http://www.niras.com/

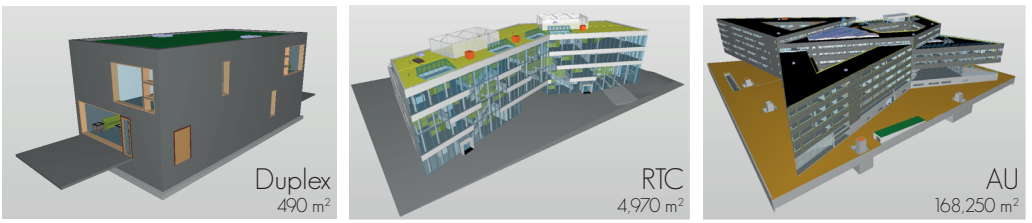


Figure 11: The three BIM models (Duplex Apartments [Duplex], Technical College in Roskilde [RTC], and the Navitas building at Aarhus University [AU]) viewed in Solibri Model Viewer.

versity, Denmark [168,250 m^2]. The exports were performed on a Lenovo P50 laptop with Intel Core i7-6820HQ 2.70 GHz CPU and 32 GB 2133 MHz DDR ram.

Table 1: File conversion

	Duplex	**RTC**	**AU**
Proprietary (Revit) file sizes [KB]			
Uncompressed	10,148	137,068	245,036
Compressed	9,381	104,043	223,861
IFC file sizes [KB]			
Uncompressed	2,363	36,847	183,014
Compressed	304	4,891	20,520
LBD file sizes [KB]			
Uncompressed	278	6,495	27,623
Compressed	27	523	2,448
Conversion time [hh:mm:ss]			
Min	00:00:03.5	00:00:31	00:15:51
Avg	00:00:04.3	00:00:33	00:16:14
LBD file size sub-components [%]			
BOT	17.7	10.7	15.7
PRODUCT	2.1	1.1	1.5
Classes	1.1	0.1	0.1
PROPS	10.3	4.2	7.8
Geometry	68.8	83.9	74.9
Number of triples [-/%]			
Total	1,715	20,219	125,973
BOT	53.2	55.0	57.1
PRODUCT	6.7	6.6	6.8
Classes	2.6	0.6	0.3
PROPS	20.3	15.8	16.7
Geometry	17.2	22.0	19.1

At its current development stage, the exporter does not support all BOT axioms (for example adjacent elements are limited to walls) and it only exports a limited set of properties and element classes (Revit types), but it has advanced significantly since it was initially presented in (Rasmussen et al., 2017a). Table 1 provides some insights into what is exported by the exporter and how the exporter performs. The conversion times are based on 5 consecutive exports and all properties are exported as simple datatype properties (owl:DatatypeProperty) with no units - this also applies to the 2D geometry boundaries that are exported as Well Known Text (WKT) formatted literals and the 3D geometry which is exported as OBJ formatted literals.

The file sizes are significantly smaller than the native Revit files and the IFC files. File sizes are further reduced when compressed, which is not the case for the Revit files. The information in the files is limited compared to their native source, especially since no geometry semantics are maintained (mesh geometry is included in OBJ encoding). Nevertheless, the necessary topology semantics and properties (ie. deduced from native geometry) are present. This graph (basic geometry, topology, and properties) forms an excellent baseline for deducing further information and for web-based collaboration. If fully semantic geometry would be desired, the ifcOWL or BREP ontologies can still be used.

Conversion times increase significantly with file sizes, but at this point, no special attention has been paid towards improving performance. A large fragment of the conversion time is constituted by the generation of geometry, and, when omitting this part, the conversion time of the AU model is halved (00:07:38 on average). Extraction of relationships that do not immediately exist in the native BIM model requires significant resources as they depend on operations such as ray tracing. This processing, however, provides some information that was not easily available anyway, and there are several cases where this approach could provide great value to the project participants. The synchronization of changes to the CDE could be performed on a server and do not necessarily need to happen very often, and in such a setup, conversion performance is not crucial.

Geometry constitutes more than half of the file sizes but when looking at the triple counts it only represents around 20 %. In this regard, the BOT semantics constitute the largest fragment of the export. The LBD export of the *Duplex apartment* is available in a Github repository[24] for further study. The repository also includes a demo application that performs topology queries on the model and returns the geometrical representations of the returned elements and zones in 2D and 3D.

3.5. Reasoning performance

To give an indication of how much time it takes an off the shelf reasoning engine to infer implicit knowledge of an LBD dataset on a regular workstation, a test program was developed. The program loads a dataset into a Stardog[25] triplestore (v5.2.2)

[24] https://github.com/MadsHolten/BOT-Duplex-house/tree/v1.0
[25] http://www.stardog.com/

and executes a set of queries 10 times to establish a mean value. The models and test system from the previous section was used for the performance test.

A set of queries asking for axioms that are not directly present in the dataset, but are given by BOT, were executed on the models in order to evaluate how resource-intensive the reasoning work is. To make sure that no results were stored in memory, a cold start on a new database was done for each query loop. The queries were sent to a local Stardog triplestore through its HTTP API. *SL* was used as reasoning type, thereby supporting both *RDFS* and *OWL 2 QL*, *RL*, and *EL* axioms. For the tested queries, there was no significant performance difference between profiles *SL*, *RL* and *EL*, but since BOT has an expressivity of $\mathcal{SRIN}(\mathcal{D})$, profiles *RDFS* and *OWL 2 QL* returned none or incorrect results for queries *Q3-5*. Stardog describes that profile *SL* takes into account all *OWL 2 DL* axioms in addition to Semantic Web Rule Language (*SWRL*[26]) rules, but as the results in Table 3 reveal, there are quite significant differences between *DL* and *SL*.

Q1. Return all instances of bot:Zone. All instances of bot:Site, bot:Building, bot:Storey and bot:Space will be returned since they are subclasses of bot:Zone.

Q2. Return zones contained in a bot:Storey using bot:containsZone. All instances of bot:Space related to a bot:Storey using the bot:hasSpace property will be returned since this is a sub-property of bot:containsZone.

Q3. Return zones contained in a bot:Site using bot:containsZone. The query is similar to query 2, but since bot:containsZone is a transitive property all zones contained in another zone and so on will be returned.

Q4. Return elements contained in a bot:Site using bot:containsElement. All elements contained in a zone which is contained in the bot:Site instance will be returned.

Q5. Return elements with a relation to a bot:Site using bot:hasElement. All elements contained, adjacent to or intersecting with a zone which is contained in the bot:Site-instance will be returned.

Q6. Return the props:width of each wall instance. When exporting from Revit, the width is given at type level and is inherited to all instances of this type. How this is achieved is described in the next section (Listing. 1).

Table 2: Reasoning performance

	Duplex		**RTC**		**AU**	
	Res. [-]	Time [sec]	Res. [-]	Time [sec]	Res. [-]	Time [sec]
Q1	27	0.04	169	0.17	1,406	0.94
Q2	21	0.01	146	0.02	1,392	0.11
Q3	26	0.01	153	0.01	1,405	0.06
Q4	61	0.02	1,468	0.01	7,460	0.35
Q5	102	0.03	1,858	0.19	11,183	0.87
Q6	57	0.01	976	0.08	6,181	1.24

The results in Table 2 show that query 1, 5 and (for the large model) query 6 are the more resource intensive ones. Query 4 and 5 were expected to be demanding as the reasoner first has to infer the super-property bot:containsZone and afterwards retrieve all the elements contained in, intersecting with and adjacent to these zones. Query 1 asking for instances of the bot:Zone class is resource intensive from a reasoning perspective as any dataset has a large amount of rdf:type properties specified, thereby leaving the reasoner a lot of data to process. Inheriting properties from type level is quite resource intensive, but this is not the case when using reasoning profile DL (Table 3). Also Q4-Q5 perform significantly better using this profile. The overall picture is that query times are optimal even with quite large datasets.

Table 3: Reasoning performance for different reasoning profiles. All results are from model *AU*. Gray indicates lower performance.

	EL	**RL**	**DL**	**SL**
	Query time [sec]			
Q1	0.97	0.99	1.17	0.94
Q2	0.09	0.10	1.09	0.11
Q3	0.07	0.06	0.04	0.06
Q4	0.35	0.36	0.18	0.35
Q5	0.92	0.91	0.26	0.87
Q6	1.26	1.25	0.14	1.24

[26]https://www.w3.org/Submission/SWRL/

4. Use cases

In this section, we describe two examples of how to use semantic web technologies with BOT as a mediator to establish an alternative to the "silo models" of today's BIM. Every project stakeholder sees the building from their own perspective, and these are examples of how the project knowledge base of each stakeholder, and hence the common knowledge base for the project as a whole, could organically grow with the project in a web-based manner (BIM Level 3).

For simplicity, we do not deal with units of quantitative properties when constructing the datasets and their associated queries[27]. There are several methods for doing this, where the simpler approach is to just specify a unit-value set as a simple string and assign a datatype to it that specifies how the string should be interpreted. This approach is used by Custom Datatypes (CDT) (Lefrançois and Zimmermann, 2016). Another approach is to assign the quantitative property as an object property. Then, the object of that property instance holds the value and the unit as two separate properties. This approach is, among others, used by QUDT (Quantities, Units, Dimensions, and Types)[28], OM (Ontology of units of Measure) and schema.org (Rijgersberg et al., 2013; Guha et al., 2016).

4.1. Case 1: One element with multiple disciplinary interfaces

The first case deals with an often occuring situation where multiple stakeholders each have specific requirements for a single building element. This task cannot be accomplished in the current BIM setup where each component resides in a model owned by one discipline. A simple example is a wall instance, which typically belongs to the architect, and therefore originates from the architectural model. When the energy engineer needs to specify thermal requirements of the wall and its subcomponents, or the structural engineer needs to specify its static capabilities, they have no way to do this in a BIM-integrated way. Therefore, in today's BIM processes, they will manually have to make their own representation of the wall instance that they will later need to maintain as the origin changes. Some BIM authoring tools like Revit do offer mechanisms to maintain a relationship between the original instance and a copy in another model, which is what Törmä (2013) refers to as the *linking approach*. In such case, meaningful relations (location, geometry etc.) are maintained between the objects describing same real-world entities. This, however, forces all designers to work in the same proprietary BIM environment. In reality, the different representations of the entities are either defined in a separate software package specifically aimed at energy simulation or finite element calculations or simply in a spreadsheet or a text document. As there exists no digital link between the different representations of the wall, a change in the source model will not be reflected throughout the other software applications used, and inconsistencies will eventually occur.

In this use case, we will illustrate how each project participant can (1) define and assign classes that specify a set of properties that are valid for all instances belonging to that class and (2) use BOT terminology to assign more knowledge to the architect's wall instance. (1) is achieved by defining OWL property restrictions as shown in Listing 1. In the example, the prod:Wall class is extended with a project-specific wall class. This class has a restriction stating that it must have a props:thickness of 360 mm. Defining the wall thickness at type level rather than instance makes sense since all instances must have the same width. The area, however, must be defined explicitly for all instances as it is illustrated in Figure 13. This figure illustrates a wall instance with URI `<http://some-arch.com/proj/100100/Walls_123>`. The hostname (some-arch.com) indicates that it was initially created by the architect. The wall is an instance of `arch:HeavyWall1` and hence also of prod:Wall. Since it is specified that it is bot:adjacentElement to a space, the wall instance also becomes an instance of bot:Element due to the mechanisms described in the previous section.

Figure 13 uses two predicates for the area property. This can be achieved by stating in the TBox that props:area is an owl:equivalentProperty to a fictive `bsdd:area` referring to the buildingSMART data dictionaries definition of an area. The wall instance also has the property props:uValue assigned, but this property has been specified by the Indoor Climate and Energy (ICE) engineer (as a type property valid for all `ice:WallTypeA` instances) and therefore resides

[27] http://www.student.dtu.dk/~mhoras/ac/use-cases/
[28] http://qudt.org

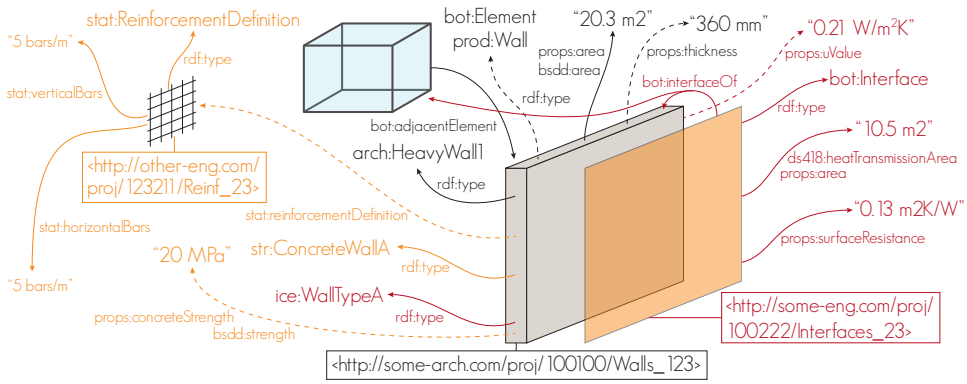


Figure 13: One wall, different interpretations. The illustrated wall has data described by three different stakeholders. Black text (middle) indicates data from the architect's dataset, red (right) from the ICE engineer's and orange (left) from the structural engineer's. Dashed lines indicate implicit knowledge which can be inferred.

in another dataset. This dataset also contains an instance of a bot:Interface representing a subset of the building envelope intersecting with the wall, which is attached to the architect's wall and space instance by the bot:interfaceOf relationship. The interface has a props:surfaceResistance and a `ds418:heatTransmissionArea` assigned. The latter is defined in a fictive Danish Standard 418 namespace (representing a national code), which is in the TBox specified as a sub-property of props:area. Similar to the property in the bsdd namespace, this is an illustration of how the more generic props:area will also be inferred in order to provide meaning for the parties who cannot interpret `ds418:heatTransmissionArea`.

Listing 1: Example of how to assign properties at type level that will be inherited by all instances.

```
@prefix arch: <http://some-arch.com/proj/100100/> .

arch:HeavyWall1 rdfs:subClassOf prod:Wall .
arch:HeavyWall1 rdfs:subClassOf [
  a owl:Restriction ;
  owl:onProperty props:thickness ;
  owl:hasValue "360"^^xsd:decimal
]
```

The structural engineer has a different interpretation of this type of wall, specified by the `str:ConcreteWallA` class. This class contains property restrictions for `stat:concreteStrength` (sub-property of `bsdd:strength`) and a `stat:re-inforcementDefinition` holding data about the number of bars in horizontal and vertical directions respectively. The `stat:` abbreviation is a prefix for another fictive namespace, and the properties attached to it are also just examples to illustrate the potentials of a distributed dataset.

Querying the data. Getting a specific property of the wall can be accomplished using a simple SPARQL SELECT query (Listing 2).

Listing 2: SPARQL query to get all properties of the wall in Figure 13.

```
SELECT ?property ?value
WHERE {
  arch:Walls_123 ?property ?value .
}
```

When the query (Listing 2) is performed only on the architect's dataset, it returns the results shown in the upper section of Table 4. Including the datasets of the two engineers further returns the results of the next two sections, and enabling reasoning will infer the data shown in the last section.

4.2. Case 2: Building Systems

The second case deals with a related BIM collaboration challenge; the fact that the different stakeholders (architect, ICE engineer, HVAC engineer, structural engineer, automation engineer etc.) of a construction project each have their own interpretations of and demands to the project material based on their individual perspectives. The challenge originates from the same problem that was dealt with in the previous case: the

Table 4: Query from Listing 2 on the dataset.

property	value
Architect's dataset	
rdf:type	arch:HeavyWallA
props:area	20.3 m^2
ICE engineer's dataset	
rdf:type	ice:WallTypeA
Structural engineer's dataset	
rdf:type	str:ConcreteWallA
Inferred knowledge	
rdf:type	bot:Element
rdf:type	prod:Wall
props:thickness	360 mm
bsdd:area	20.3 m^2
props:uValue	0.21 W/m^2K
props:concreteStrength	20 MPa
stat:reinforcementDefinition	str:Reinf_23
bsdd:strength	20 MPa

silo structure of today's BIM tools. Below is a brief overview of how building spaces are roughly interpreted from the point of view of different practitioners in the AEC domain.

From an architect's perspective a building space is something which serves certain functional capabilities in terms of aesthetic aspects, view to the outside, access to daylight, number of occupants to accommodate etc.

From an ICE engineer's perspective a building space is a thermal zone which has internal heat gains depending on the owner or tenant's intended use of the spaces and external solar heat gains and losses that are highly dependent on the space's geometry.

From an HVAC engineer's perspective a building space is not of particular interest, but the heating-, cooling- and ventilation demands of the space given by the ICE engineer are relevant as they define the boundary conditions of the flow terminals located in the space. Actually, the capacity and layout of the future systems can to a certain degree be determined solely by grouping spaces into zones.

From a Facility Manager's (FM) perspective a building space needs to fulfil the demand of the space's users. If equipment located in the space is malfunctioning it is convenient to file a report where it is described in which space the equipment is located.

From an automation engineer's perspective a building space is a control zone that has certain demands that are to be fulfilled by the automation systems operating on that zone. This involves light control, fire evacuation and control of the thermal and atmospheric indoor climate. A control zone hosts a set of devices, actuators and sensors that integrate with the electrical and mechanical systems of the building.

From a contractor's perspective all components must be priced, and so must the labour associated with the execution of the construction. When construction is ongoing, the contractor further needs to schedule what components go where when they are delivered on site and building spaces and zones

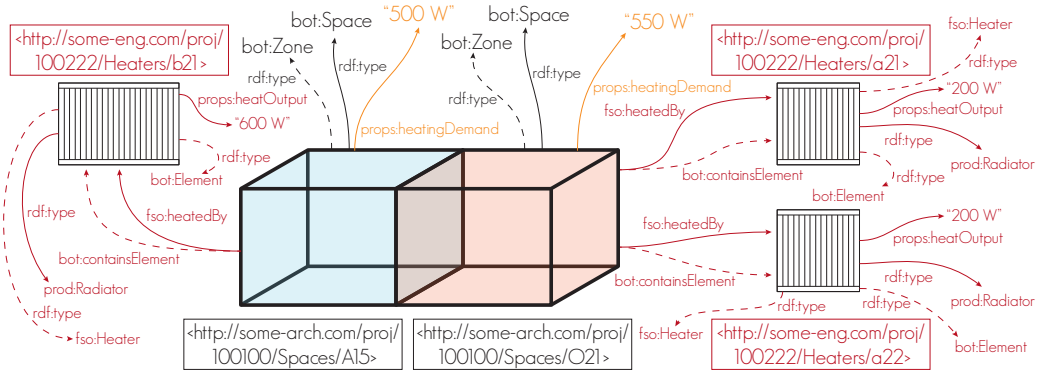


Figure 14: Space heating demand vs. actual heat supplied to the space. See Figure 13 for color and line type specifications.

then become containers for products.

The different interpretations and demands of the same physical spaces emphasize the need for a data model that can facilitate the management of all the interdependencies as the project evolves over time both during the design stages, but also during and after construction. *Use case 2* deals with a subset of the example and illustrates a concept of how BOT can be used in managing the interdependencies.

The dataset. Part of the dataset for use case 2 is illustrated in Figure 14. Two architectural spaces are enriched with props:heatingDemands specified by the ICE engineer, and the HVAC engineer further connects this to a `fso:Heater` instance using a `fso:heatedBy` object property. The TBox defines `fso:heatedBy` as a sub-property of bot:containsElement and its range is a `fso:Heater`. Each heater is assigned a props:heatOutput and it is explicitly classified as a prod:Radiator.

Heater capacity Quality Assessment (QA). The total capacity of the heaters in a space should match the heating demand of the space, and the dataset of the architect, ICE engineer and HVAC engineer together contain all the necessary data to do a QA of the heater capacities.

Listing 3 shows a full SPARQL SELECT query to achieve a list of all spaces that have a props:heatingDemand specified and is `fso:heatedBy` at least one heater. It sums up the props:heatOutput of the heaters of each space, compares the demand to the design and returns the results shown in Table 5.

Listing 3: SPARQL query to compare heating demands to heat output.

```
SELECT ?space ?heatingDemand ?heatOutput ?overCapacity
       ?heatRequirementFulfilled
WHERE {
  ?space a bot:Space .
  ?space props:heatingDemand ?heatingDemand .

  # TOTAL HEAT OUTPUT TO EACH SPACE (SUB-QUERY)
  { SELECT ?space (SUM(?ho) AS ?heatOutput)
    WHERE {
      ?space  fso:heatedBy ?heater .
      ?heater props:heatOutput ?ho .
    } GROUP BY ?space
  }

  # CALCULATE OVER CAPACITY
  BIND(?heatOutput/?heatingDemand AS ?overCap)

  # CHECK IF DEMAND IS MET
  BIND(IF(?overCapacity > 1, xsd:boolean(true),
    xsd:boolean(false)) AS ?demandMet)
}
```

Table 5: Result of query from Listing 3 where hd=heatingDemand, ho=heatOutput, oc=overCapacity, hf=heatingDemandFulfilled

space	hd	ho	oc	hf
arch:A15	550	600	1.09	true
arch:O21	500	400	0.8	false

Heater colours. The architect might wish to specify the colour of the radiators in a space, and, since FSO specifies the domain-specific `fso:heatedBy` property as a sub-property of bot:containsElement, it is possible for the architect to query for this generic property instead, in order to retrieve the radiators (Listing 4). As the number of domain-specific terms in the dataset increase, the generic BOT terms become increasingly important for other domains to decode their meaning. With a full list of the requested radiators (Table 6), the architect can construct an `INSERT` query to assign the desired colours.

Listing 4: SPARQL query to get all elements in a space.

```
SELECT ?space ?radiator
WHERE {
  ?space a bot:Space .
  ?space bot:containsElement ?radiator .
  ?radiator a prod:Radiator .
}
```

Table 6: Result of query from Listing 4

space	radiator
arch:A15	eng:b21
arch:O21	eng:a21
arch:O21	eng:a22

Delivery zones. Aggregation of zones is advantageous for various purposes. In this example, we extend the dataset from Figure 14 with similar spaces and heaters and define two delivery zones (bot:Zone instances) for specifying where at the construction site to deliver specific products. The aggregation is achieved by adding a bot:containsZone link to the space instances contained in each zone. The property chain axiom bot:containsZone/bot:containsElement ensures that the elements contained in the spaces will be assigned to the delivery zones as well. Hence, as elements are delivered on site, one can ask what delivery zone they belong to. Figure 15 shows the sub-

graph returned when asking for all elements that belong to a specific delivery zone (Listing 5).

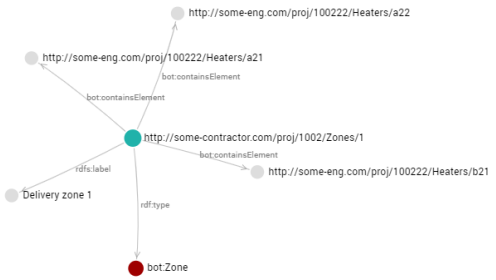


Figure 15: Elements contained in a delivery zone. Screenshot from interactive web application[29]

The more generic bot:hasElement could also have been used to inherit all elements intersecting the zone or with adjacency to the zone (such as windows). Caution must, however, be taken in this situation since one element will then be present in several delivery zones unless the geometry is divided.

Listing 5: SPARQL query to get all elements in "Delivery zone 1".

```
CONSTRUCT
WHERE {
  ?zone rdfs:label "Delivery zone 1" .
  ?zone bot:containsElement ?el .
}
```

5. Proof of Concept

In this section, the modelling approach from use case 2, Section 4, is tested on the models that were also tested in regard to query performance in Section 3. All queries are available in the earlier mentioned web application[30]. The purpose is to (1) examine performance on larger datasets and (2) illustrate an alternative workflow for modelling BIM objects even at early design stages where they only exist conceptually and not geometrically in a BIM authoring tool (see also Bonduel et al. (2018a)). This implies that each object can be referred to explicitly, and hence properties and requirements to the object along with its relationships to other design objects can be described.

[29] http://www.student.dtu.dk/m̃horas/ac/use-cases/

[30] `http://www.student.dtu.dk/~mhoras/ac/proof-of-concept/`

5.1. Simulating distributed datasets

There are a few different approaches when querying over distributed datasets. Federated SPARQL queries are possible using the `SERVICE` keyword which tells the query engine that part of the dataset resides on another server. This option is also possible with Linked Data Fragments (LDF) (Verborgh et al., 2014), which is a proposal that takes part of the workload of the queries away from the servers and instead handles it on the client. Another option is to simply keep copies of the other parties' datasets in different sub-graphs in the same database; so-called named graphs. Keeping the copies up to date can be handled automatically as a routine job, thereby removing reliance on 3rd party's servers. For this experiment, the latter approach is used. Each project participant hosts a database where only part of it is available to the other project participants. This is achieved by defining access restrictions for the different named graphs in the database. Some data is deduced from and therefore relies on data that was initially defined in one of the other project participants' datasets.

For the PoC, the three models from Section 3 were used, but, in order to illustrate a workflow which operates with the concept of property states, the property values were exported from Revit as OPM property states. Initially, the LBD models were loaded into a named graph in the namespace of the architect (`http://architect.com/projects/projectNo`), as the source files are architectural models. A set of queries were performed on the datasets to simulate a workflow where new data is deduced from existing data in an interconnected way that allows for tracking. The result is an alternative BIM information modelling approach which is documented step by step in the web app for model *Duplex*.

The modelling approach is based on a set of queries to access, process and extend other parties' datasets in an integrated linked data based manner. Figure 16 shows a sample of the dataset. The content of named graph 1 (NG1) comes from the architectural model, whereas the other 2 (NG2 and NG3) are generated with the queries described below. Future changes to the architect's model are handled by inferring new property states of the affected properties using OPM terminology. As the other parties relate the derived properties directly to the specific state of the property on which they

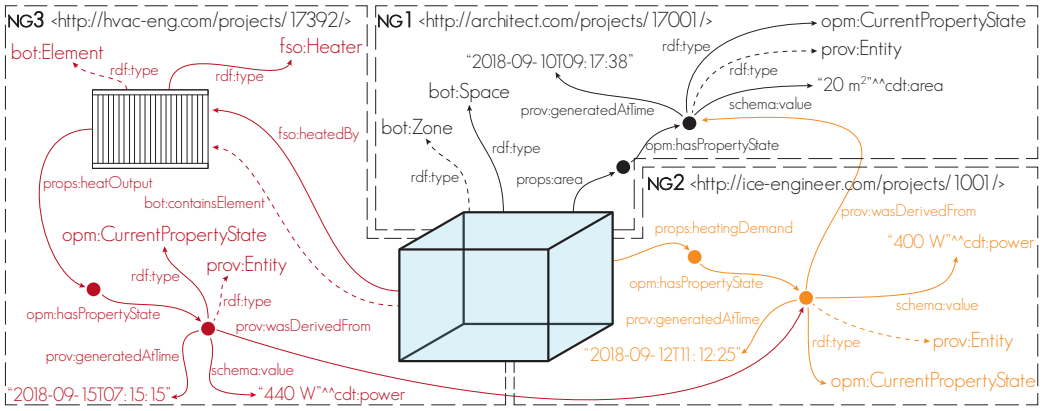


Figure 16: Simulating a distributed dataset. The content of the HVAC engineer's dataset in named graph 3 (NG3) is generated based on features in the ICE engineer's dataset in NG2 which is again based on the content of features in the architect's dataset in NG1. See Figure 13 for color and line type specifications. Graph nodes are either illustrated or shown as dots.

are derived, it is possible, at any future point in time, to check whether the property has become outdated due to changes in the architectural design. In Figure 16, these relations (described with prov:wasDerivedFrom) are the only ones that cross the borders of the named graphs..

Q1. The first task when the data is loaded into the store is to perform a query on the dataset to retrieve all space instances in the model along with the latest state of their names as specified by the architect. This is done by asking for anything that is an instance of bot:Space and the property props:identityDataName (Listing 6).

Listing 6: SPARQL query to get all spaces and the latest state of their names. Namespace prefixes as provided by the `http://prefix.cc/` service.

```
PREFIX arch: <http://architect.com/projects/>
SELECT ?space ?name
WHERE { GRAPH arch:17001 {
  ?space a bot:Space .
  ?space props:identityDataName/opm:hasPropertyState [
    a opm:CurrentPropertyState ;
    schema:value ?name ]
} }
```

Q2. The second query retrieves all spaces and their areas from the architect's named graph in a subquery similar to the one in Q1. To illustrate how new data can be generated based on existing data, dummy heating demands are generated as the space area multiplied by a factor of 20. The heating demands are stored in another named graph in the namespace of the ICE engineer, referring to the architectural spaces by their URIs. This is all handled in one `INSERT` query (Listing 7).

Had the data been available in the model, the heating demands could potentially be calculated as an infiltration heat loss (often given by a factor to multiply with the space area) and the sum of the transmission heat losses through the part of the building envelope that shares an interface with the space.

Modelling preliminary heating demands that are likely subject to future changes is desirable as it provides a source for other stakeholders to relate to. As derived properties are linked to the heating demands it will be possible to do quality checks similar to the *Heater capacity QA* example in use case 2 in order to evaluate if a revision is required.

Q3. The third query illustrates how data from different datasets (in this case different named graphs) can be retrieved. The query returns all spaces and their areas (from the architect's dataset) and the heating demands (from the ICE engineer's dataset). The query is similar to Listing 6, but additionally looks into graph `ice:1001` to retrieve the heating demands.

Q4. The fourth query is similar to the second (Listing 7) but, this time, dummy heaters are generated

based on the heating demands specified by the ICE engineer. Each heater is assigned an initial heat output of the heating demand + 10 %, and all the data is stored in a third named graph in the namespace of the HVAC engineer. As the project progresses, a dummy heater can be split into several heaters and the heat output of each heater can then be adjusted accordingly.

Listing 7: SPARQL query to infer dummy heating demands. Namespace prefixes as provided by the `http://prefix.cc/` service.

```
PREFIX ice: <http://ice-engineer.com/projects/>
INSERT{
  GRAPH ?gICE {
    ?space          props:heatingDemand ?newPropertyURI .
    ?newPropertyURI opm:hasPropertyState ?newStateURI .
    ?newStateURI    a opm:CurrentPropertyState .
    ?newStateURI    schema:value ?hd .
    ?newStateURI    prov:wasDerivedFrom ?stateURI .
    ?newStateURI    prov:generatedAtTime ?now .
  }
}
WHERE {
  BIND(props:dimensionsArea AS ?prop)
  BIND(arch:17001 AS ?gArch)
  BIND(ice:1001 AS ?gICE)

  # GET SPACE AREA FROM ARCHITECT'S GRAPH
  GRAPH ?gArch {
    ?space         a bot:Space .
    ?space         ?prop ?propURI .
    ?propURI       opm:hasPropertyState ?stateURI .
    ?stateURI      a opm:CurrentPropertyState .
        ?stateURI      schema:value ?area .
  }
  # GENERATE DUMMY HEATING DEMANDS
  BIND( CEIL(20 * ?area) AS ?hd )

  # ONLY CREATE IF PROPERTY NOT ALREADY DEFINED
  MINUS{GRAPH ?gICE {?space props:heatingDemand ?oldHd}}

    # GENERATE URIs AND GET CURRENT TIME
  BIND(URI(CONCAT(STR(?gICE),"#","property_",STRUUID()))
      AS ?newPropertyURI )
  BIND(URI(CONCAT(STR(?gICE),"#","state_",STRUUID()))
      AS ?newStateURI )
  BIND( NOW() AS ?now )
}
```

Q5. The fifth query is similar to the third, but this time the data is retrieved from 3 different graphs.

Q6. The sixth query is similar to the one shown in Listing 3 except that it takes the information from two different named graphs in order to make the comparison between the heat demanded by the ICE engineer and the heat that is actually supplied by heaters in each space. An over-capacity is calculated and a simple test is indicating whether it is fulfilled or not. This yields results similar to the ones shown in Table 5.

Q7. The seventh query is to illustrate the change tracking concept described in *Q2*. An update of a subset of the space heating demands is simulated by adding 20 % to the previous heating demand in the case that it is currently below 100 W. The structure of this query is similar to the one shown in Listing 7, but no data is to be retrieved from the architect's graph this time.

Q8. The eighth query is similar to the sixth, but, since a subset of the ICE engineer's heating demands were updated in *Q7*, some of the heaters are now undersized. Using a filter, only the spaces with insufficient heat supply are returned.

5.2. Performance

To give an indication of how responsive the data model will be when performing the above queries, the test program from Section 3 was used again. Since the triplestore indexes data based on incoming queries, query response times improve significantly as they are performed again. To show both first-time performance and the performance a user of the system will most often experience, both a cold and warm start on the `GET` queries was performed. For `INSERT` queries, it would be unfair to make the comparison as no new triples are generated the second time. The warm start is simulated by performing the queries 10 times without wiping the store for each loop and take the minimum query time.

Table 7: Query performance.

	Duplex		**RTC**		**AU**	
	cold [sec]	warm [sec]	cold [sec]	warm [sec]	cold [sec]	warm [sec]
Q1	0.01	0.01	0.02	0.01	0.05	0.04
Q2	0.08	-	0.17	-	0.30	-
Q3	1.02	0.01	0.64	0.01	0.75	0.05
Q4	0.06	-	0.08	-	0.21	-
Q5	1.17	0.16	1.02	0.02	1.12	0.09
Q6	0.05	0.01	0.48	0.02	0.56	0.11
Q7	0.06	-	0.07	-	0.14	-
Q8	0.06	0.01	0.46	0.01	0.49	0.05

The results in Table 7 show query execution times for both cold and warm starts and reveal quite impressive performance even with complex queries on large datasets. Being able to query such large datasets in a matter of milliseconds could significantly improve the productivity of industry practitioners, and the alternative approach to processing

BIM data illustrates a glimpse of how available data can be used to deduce new data in a structured way.

6. Conclusion

With the evolution towards the use of web technologies in the AEC industry, a common web-compliant language is needed to mediate the interdisciplinary communication in the AEC industry. This ideally happens in a standardised manner that is nevertheless still closely related to practice. IFC is the de facto standard in this regard, yet it has specific shortcomings in its implementation, thereby typically limiting its scope to the exchange of data files focused on geometry exchange between BIM authoring tools. IFC itself is furthermore not inherently web-compliant, considering its focus on EXPRESS and ISO/STEP as information modelling languages. Furthermore, the IFC schema is often perceived as non-modular, complex and non-extensible, leading to calls for simplicity, extensibility, and modularity from various corners. An important first step is made towards the web with the current OWL version of IFC (ifcOWL). However, notwithstanding its web-based character, a number of important drawbacks persist that limit its use for full semantic interoperability, namely (1) lack of modularity, (2) near to impossible extensibility, and (3) high complexity.

6.1. Contributing to a web-based AEC industry

This article looked into this situation and investigated alternatives that deploy state of the art semantic web standards (W3C), languages (OWL, RDF, RDFS) and best practices (modularity and simplicity). We presented an alternative LBD approach that starts from simplicity, modularity and extensibility as a baseline and relies on the BOT ontology as a cornerstone ontology. We illustrated linking approaches for extending the minimal ontology with domain-specific classes and properties, and by applying these in simple use cases, we proved that the Building Topology Ontology (BOT) can function as a foundation for the requested common language.

By applying the common language in a PoC case we showcased an approach to a data-driven BIM process dealing with interrelated properties purely from semantics, which can eventually take the industry to BIM maturity Level 3. The approach allows all project participants to access a network of updated project information that can be managed, queried, processed and extended for specific purposes. Furthermore, since it is defined in a semantic web language it can be further extended with Linked Open Data such as material properties and life cycle assessments, physical units, weather data, and so forth. This easily opens the door to a web-based integration on a data level with geospatial information systems (GIS) and facility management (FM) applications.

We have evaluated LBD datasets generated from three native Revit BIM models in terms of exporter performance and content of the datasets, and, although optimizations can clearly still be made, the exporter is currently capable of extracting a substantial amount of information from the native BIM environment. Based on the same models, we performed benchmarks on the PoC case and found that interacting with the datastore is responsive with typical query times of less than a second even on a large university building of more than 150,000 m^2. This proves that the technologies are mature and could already today vastly improve productivity in an AEC industry where information acquisition is handled in a predominantly manual, labour-intensive and error-prone manner.

6.2. Future outlook

A number of topics can be outlined for future work. A first topic that will need to be further investigated is scalability of the LBD approach into a fully functional practical environment, with running tools, running processes, and a building under construction. Furthermore, additional performance tests will be required in such future stress tests.

A second topic for further research is the extension towards custom domain ontologies, in particular regarding products and properties. At the moment, BOT only covers a building's core topology and the intention is that it should not change too much in the future. Of course, it can be extended by other ontologies that import the BOT ontology as a baseline. Since it does not describe specific sub-elements of a building it also will not be influenced by the evolution of BIM authoring tools and future releases of IFC. New products will continue to emerge and the PRODUCT and PROPS ontologies will need to deal with this reality, which should be possible by relying on the proposed modularity and extensibility as key futures for any future ontology. At the time of writing, these PRODUCT and PROPS ontologies resemble to open vocabularies.

Methods for extending them while maintaining a connection to the current and future IFC schemas should be investigated.

A third important element for investigation is the link with geometry. BOT does not describe geometry semantics but it provides two properties to attach complex or simple geometrical representations to any object in an LBD dataset using an appropriate object or datatype property depending on what suits the specific purpose. We demonstrated one approach where mesh representations of elements and spaces were exported as OBJ formatted literals. Visualising these meshes is possible using the widely adopted WebGL technology which runs in any modern web browser and this was demonstrated in a simple implementation. The read-only mesh geometry can be queried and visualized on demand, and thereby provide end users with a valuable visual feedback.

We envision that the presented web-based BIM (BIM Level 3) will primarily take the form of a linked data based CDE, thus preserving the existing BIM authoring tools mostly in their current shape. Also, existing workflows (agreements, contracts, exchanges, collaboration forms) are considered to remain in existence in their current form, yet, they may instead rely on pure data, rather than documents and files.

7. Acknowledgements

Special thanks to the NIRAS ALECTIA Foundation and Innovation Fund Denmark for funding. Also thanks to Niras for allowing open distribution of the developed artifacts. It is a fundamental necessity for the future growth of the proposed standards that they are adopted and further developed by the community.

We would also like to thank the reviewers for their valuable feedback towards improving this paper.

8. References

Balaji, B., Agarwal, Y., Berges, M., Culler, D., Gupta, R., Kjærgaard, M.B., Srivastava, M., Whitehouse, K., Bhattacharya, A., Fierro, G., Gao, J., Gluck, J., Hong, D., Johansen, A., Koh, J., Ploennigs, J., 2016. Brick, in: Bergès, M. (Ed.), Proceedings of the 3rd ACM International Conference on Systems for Energy-Efficient Built Environments - BuildSys '16, ACM Press, Palo Alto, CA, USA. doi:10.1145/2993422.2993577.

Beetz, J., van den Braak, W., Botter, R., Zlatanova, S., de Laat, R., 2014. Interoperable data models for infrastructural artefacts: a novel IFC extension method using RDF vocabularies exemplified with quay wall structures for harbors, in: eWork and eBusiness in Architecture, Engineering and Construction (ECPPM 2014). CRC Press, Vienna, Austria, pp. 135–140. doi:10.1201/b17396-26.

Beetz, J., van Leeuwen, J., de Vries, B., 2008. IfcOWL: A case of transforming EXPRESS schemas into ontologies. Artificial Intelligence for Engineering Design, Analysis and Manufacturing 23, 89. doi:10.1017/s0890060409000122.

Bertelsen, S., 2003. Construction as a complex system, in: Proceedings of the 11th Annual Conference of the International Group for Lean Construction, pp. 143–168. URL: http://www.iglc.net/papers/details/231.

Bew, M., Richards, M., 2008. Bew-Richards BIM maturity model, in: BuildingSMART Construct IT Autumn Members Meeting, Brighton, UK.

Bizer, C., Heath, T., Berners-Lee, T., 2009. Linked data - the story so far. International Journal on Semantic Web and Information Systems 5, 1–22. doi:10.4018/jswis.2009081901.

Bonduel, M., Oraskari, J., Pauwels, P., Vergauwen, M., Klein, R., 2018a. The IFC to linked building data converter - current status, in: Poveda-Villalón, M., Pauwels, P., Roxin, A. (Eds.), Proceedings of the 6th Linked Data in Architecture and Construction Workshop, CEUR-WS.org, UCL, London, UK. pp. 34–43. URL: http://ceur-ws.org/Vol-2159/04paper.pdf. accessed September 2018.

Bonduel, M., Rasmussen, M.H., Pauwels, P., Vergauwen, M., Klein, R., 2018b. A novel workflow to combine bim and linked data for existing buildings, in: eWork and eBusiness in Architecture, Engineering and Construction (ECPPM 2018), CRC Press, Copenhagen, Denmark. pp. 407–414.

Bonino, D., Corno, F., 2008. DogOnt - ontology modeling for intelligent domotic environments, in: Sheth, A., Staab, S., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (Eds.), Proceedings of the 7th International Semantic Web Conference, Springer Berlin Heidelberg, Karlsruhe, Germany. pp. 790–803. doi:10.1007/978-3-540-88564-1_51.

Bonsma, P., Bonsma, I., Ziri, A.E., Iadanza, E., Maietti, F., Medici, M., Ferrari, F., Sebastian, R., Bruinenberg, S., Lerones, P.M., 2018. Handling huge and complex 3d geometries with semantic web technology. IOP Conference Series: Materials Science and Engineering 364, 1–7. doi:10.1088/1757-899x/364/1/012041.

BSI Standards Limited, 2013. Specification for information management for the capital/delivery phase of construction projects using building information modelling. doi:10.3403/30259522u.

Daniele, L., den Hartog, F., Roes, J., 2015. Created in close interaction with the industry: The smart appliances REFerence (SAREF) ontology, in: Cuel, R., Young, R. (Eds.), 7th International Workshop Formal Ontologies Meet Industries, Springer International Publishing, Berlin, Germany. pp. 100–112. doi:10.1007/978-3-319-21545-7_9.

Mendes de Farias, T., Roxin, A., Nicolle, C., 2015. IfcWoD, semantically adapting IFC model relations into OWL properties, in: Proceedings of the 32nd CIB W78 Conference on Information Technology in Construction. Available from: https://arxiv.org/abs/1511.03897 [March

2018].
Gao, G., Liu, Y.S., Lin, P., Wang, M., Gu, M., Yong, J.H., 2017. BIMTag: Concept-based automatic semantic annotation of online BIM product resources. Advanced Engineering Informatics 31, 48–61. doi:10.1016/j.aei.2015.10.003.
Gao, G., Liu, Y.S., Wang, M., Gu, M., Yong, J.H., 2015. A query expansion method for retrieving online BIM resources based on industry foundation classes. Automation in Construction 56, 14–25. doi:10.1016/j.autcon.2015.04.006.
Guha, R.V., Brickley, D., Macbeth, S., 2016. Schema.org: evolution of structured data on the web. Communications of the ACM 59, 44–51. doi:10.1145/2844544.
Lebo, T., Sahoo, S., McGuinness, D., Belhajjame, K., Cheney, J., Corsar, D., Garijo, D., Soiland-Reyes, S., Zednik, S., Zhao, J., 2013. PROV-O: The PROV ontology. URL: https://www.w3.org/TR/prov-o/. accessed September 2018.
Lee, S.H., Kim, B.G., 2011. IFC extension for road structures and digital modeling. Procedia Engineering 14, 1037–1042. doi:10.1016/j.proeng.2011.07.130.
Lefrançois, M., Zimmermann, A., 2016. Supporting arbitrary custom datatypes in RDF and SPARQL, in: Sack, H., Blomqvist, E., d'Aquin, M., Ghidini, C., Paolo Ponzetto, S., Lange, C. (Eds.), The Semantic Web. Latest Advances and New Domains, Springer International Publishing, Heraklion, Crete, Greece. pp. 371–386. doi:10.1007/978-3-319-34129-3_23.
Lefranois, M., Zimmermann, A., Siira, E., Ghariani, T., Girod-Genet, M., Santos, G., Silva, F., Temal, L., Teixeira, B., Jrvinen, H., 2017. SEAS Ontology. URL: https://ci.mines-stetienne.fr/seas/index.html. accessed September 2018.
Liu, H., Liu, Y.S., Pauwels, P., Guo, H., Gu, M., 2017. Enhanced explicit semantic analysis for product model retrieval in construction industry. IEEE Transactions on Industrial Informatics 13, 3361–3369. doi:10.1109/TII.2017.2708727.
Niknam, M., Karshenas, S., 2017. A shared ontology approach to semantic representation of BIM data. Automation in Construction 80, 22–36. doi:10.1016/j.autcon.2017.03.013.
Pauwels, P., Krijnen, T., Terkaj, W., Beetz, J., 2017a. Enhancing the ifcOWL ontology with an alternative representation for geometric data. Automation in Construction 80, 77–94. doi:10.1016/j.autcon.2017.03.001.
Pauwels, P., Poveda-Villalón, M., Sicilia, Á., Euzenat, J., 2018. Semantic technologies and interoperability in the built environment. Semantic Web 9, 731–734. doi:10.3233/sw-180321.
Pauwels, P., Roxin, A., 2016. SimpleBIM : From full ifcOWL graphs to simplified building graphs, in: Christodoulou, S.E., Scherer, R. (Eds.), eWork and eBusiness in Architecture, Engineering and Construction (ECPPM 2016), CRC Press, Limassol, Cyprus. pp. 11–18. doi:10.1201/9781315386904.
Pauwels, P., Terkaj, W., 2016. EXPRESS to OWL for construction industry: Towards a recommendable and usable ifcOWL ontology. Automation in Construction 63, 100–133. doi:10.1016/j.autcon.2015.12.003.
Pauwels, P., Zhang, S., Lee, Y.C., 2017b. Semantic web technologies in aec industry: a literature review. Automation in Construction 73, 145–165. doi:10.1016/j.autcon.2016.10.003.
Perzylo, A., Somani, N., Rickert, M., Knoll, A., 2015. An ontology for CAD data and geometric constraints as a link between product models and semantic robot task descriptions, in: Burgard, W., Tapus, A., Pradalier, C., Torras, C., Kragic, D., Song, D., Yoshida, E., Arai, F., Oriolo, G., Castellanos, J.A., Dias, J., Peters, J., Merlet, J.P., Kim, K., Bennewitz, M., Gross, R., Harai, S., Srinivasa, S., Arai, T., Kroeger, T. (Eds.), 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, Hamburg, Germany. pp. 4197–4203. doi:10.1109/iros.2015.7353971.
Rasmussen, M.H., Bonduel, M., Hviid, C.A., Karlshøj, J., 2018a. Managing space requirements of new buildings using linked building data technologies, in: eWork and eBusiness in Architecture, Engineering and Construction (ECPPM 2018), CRC Press, Copenhagen, Denmark. pp. 399–406.
Rasmussen, M.H., Hviid, C.A., Karlshøj, J., 2017a. Web-based topology queries on a BIM model, in: 5th Linked Data in Architecture and Construction Workshop, Univeristy of Burgundy, Dijon, France. doi:10.13140/RG.2.2.22298.95685.
Rasmussen, M.H., Lefrançois, M., Bonduel, M., Hviid, C.A., Karlshøj, J., 2018b. OPM: An ontology for describing properties that evolve over time, in: Poveda-Villalón, M., Pauwels, P., Roxin, A. (Eds.), Proceedings of the 6th Linked Data in Architecture and Construction Workshop, CEUR-WS.org, UCL, London, UK. pp. 24–33. URL: http://ceur-ws.org/Vol-2159/03paper.pdf. accessed September 2018.
Rasmussen, M.H., Pauwels, P., Hviid, C.A., Karlshøj, J., 2017b. Proposing a central AEC ontology that allows for domain specific extensions, in: Bosché, F., Brilakis, I., Sacks, R. (Eds.), Proceedings of the Joint Conference on Computing in Construction, Heriot-Watt University, Heraklion, Crete, Greece. doi:10.24928/jc3-2017/0153.
Rasmussen, M.H., Pauwels, P., Lefrançois, M., Schneider, G.F., Hviid, C., Karlshøj, J., 2017c. Recent changes in the Building Topology Ontology, in: 5th Linked Data in Architecture and Construction Workshop, Univeristy of Burgundy, Dijon, France. doi:10.13140/RG.2.2.32365.28647.
Reinisch, C., Kofler, M., Iglesias, F., Kastner, W., 2011. ThinkHome energy efficiency in future smart homes. EURASIP Journal on Embedded Systems 2011, 104617. doi:10.1155/2011/104617.
Rijgersberg, H., Van Assem, M., Top, J., 2013. Ontology of units of measure and related concepts. Semantic Web 4, 3–13.
Schneider, G.F., 2017. Towards aligning domain ontologies with the Building Topology Ontology, in: 5th Linked Data in Architecture and Construction Workshop, Univeristy of Burgundy, Dijon, France. doi:10.13140/RG.2.2.21802.52169.
Schneider, G.F., Rasmussen, M.H., Bonsma, P., Oraskari, J., Pauwels, P., 2018. Linked Building Data for Modular Building Information Modelling of a Smart Home, in: eWork and eBusiness in Architecture, Engineering and Construction (ECPPM 2018), CRC Press, Copenhagen, Denmark. pp. 407–414.
Studer, R., Benjamins, V., Fensel, D., 1998. Knowledge engineering: Principles and methods. Data & Knowledge Engineering 25, 161–197. doi:10.1016/s0169-023x(97)00056-6.
Terkaj, W., Pauwels, P., 2017. A method to generate a mod-

ular ifcOWL ontology, in: Borgo, S., Kutz, O., Loebe, F., Neuhaus, F. (Eds.), Proceedings of the 8th International Workshop on Formal Ontologies meet Industry, CEUR-WS.org, Bolzano, Italy. URL: http://ceur-ws.org/Vol-2050/FOMI_paper_3.pdf. accessed September 2018.

Törmä, S., 2013. IEEE, Irvine, CA, USA. pp. 412–419. doi:10.1109/icsc.2013.80.

Verborgh, R., Vander Sande, M., Colpaert, P., Coppens, S., Mannens, E., Van de Walle, R., 2014. Web-scale querying through linked data fragments, in: Bizer, C., Heath, T., Auer, S., Berners-Lee, T. (Eds.), Proceedings of the Workshop on Linked Data on the Web, Citeseer. CEUR-WS.org, Seoul, Korea. URL: http://ceur-ws.org/Vol-1184/ldow2014_paper_04.pdf. accessed September 2018.

Yabuki, N., Lebegue, E., Gual, J., Shitani, T., Zhantao, L., 2006. International Collaboration for Developing the Bridge Product Model "IFC-Bridge", in: n/a (Ed.), Proceedings of the W78 23rd Joint International Conference on Computing and Decision Making in Civil and Building Engineering, Montréal, Canada. pp. 1927–1936. URL: http://itc.scix.net/data/works/att/w78-2006-tf289.pdf. accessed September 2018.

6.8 OPM2: Managing Interrelated Project Information in AEC Knowledge Graphs

Under review. Submitted to Automation in Construction on January 7th 2019.

This paper describes OPM in a larger perspective and extends the terminology initially presented in paper OPM1. The main contribution is the introduction of classification for 'property reliability' and 'calculations'. The latter provide a formal way to document the reasoning logic behind derived properties. Another contribution is the proposal of a standardised way to generate parametric queries for operating an OPM-compliant Architecture, Engineering and Construction (AEC) KG. A JavaScript based query generator (OPM-QG) that follows these principles is presented. With an implementation it is demonstrated how OPM can be used to accomplish a design task on a construction project. The specific implementation deals with calculation of space heating demands and demonstrates how OPM keeps track of the interdependencies so that consequences of a design change can be assessed.

Managing interrelated project information in AEC Knowledge Graphs

Mads Holten Rasmussen[a,*], Maxime Lefrançois[b], Pieter Pauwels[c], Christian Anker Hviid[a], Jan Karlshøj[a]

[a] *Technical University of Denmark, Department of Civil Engineering, Denmark*
[b] *Ecole des Mines de Saint-Etienne, Connected-Intelligence team, France*
[c] *Ghent University, Department of Architecture and Urban Planning, Belgium*

Abstract

The construction industry is often criticised for its lacking progress in efficiency compared to other industries. One explanation could be the predominantly document-centric working-manner, where highly interrelated and rapidly changing design data in a complex network of decisions, requirements and product specifications is predominantly captured in static documents. In this paper, we consider a purely data-driven approach based on semantic web technologies and an earlier proposed Ontology for Property Management (OPM). The main contribution of this work consists of extensions for OPM for describing property reliability as well as a formalised way to document the reasoning logic behind derived properties. The secondary contribution is the specification of a standardised way to generate parametric queries for managing an OPM-compliant AEC Knowledge Graph (AEC-KG). A software library for operating an OPM-compliant AEC-KG is further presented in the form of an OPM Query Generator (OPM-QG). The library generates queries that follow OPM best practices for representation of construction project data formally, infer data, retrieve data, and provide answers to a set of pre-defined competency questions. The OPM ontology aligns with latest developments in the W3C Community Group on Linked Building Data and suggests an approach to working with design data in a distributed environment using separate graphs for explicit facts and for materialised, deduced data. Finally, we evaluate the suggested approach using a software artefact developed using OPM and OPM-QG. The particular design task evaluated is performing heat loss calculations for spaces of a future building using an AEC-KG described using domain- and project specific extensions of the Building Topology Ontology (BOT) in combination with OPM. With this work, we demonstrate how a typical engineering task can be accomplished and managed in an evolving design environment, thereby providing the engineers with insights to support decision making as changes occur. The application uses a strict division between the client viewer and the actual data model holding design logic, and can easily be extended to support other design tasks.

Keywords: Linked Data, Building Information Modeling, Complex design, Ontology, Inference, Information Exchange, BIM, AEC Knowledge Graph, Linked Building Data

1. Introduction

The architecture, engineering and construction (AEC) industry is a fragmented industry, with information spread between numerous, changing stakeholders, including architects, engineers, contractors, subcontractors, owners, and so forth. This is one of the reasons why Bertelsen (2003) describes it as a complex industry. All these stakeholders have varying levels of proficiency in digital technologies. With the advent of Building Information Modeling (BIM) tools, these varying levels of proficiency have become more apparent. They are often referred to as levels of maturity or levels of BIM adoption in the industry, and there exist frameworks for quantifying these (Succar, 2009; BSI, 2014).

1.1. Document Exchanges at the Heart of AEC Project Design Workflows

Winch (2010) describes that construction project teams can, in essence, be considered as informa-

*Corresponding author
Email address: mhoras@byg.dtu.dk (Mads Holten Rasmussen)

tion processing systems. This definition emphasises why an unobstructed information exchange between project teams is essential – an observation which is also backed up in various analyses of the construction industry (Gallaher et al., 2004; Egbu et al., 2004; McKinsey, 2017). The situation in the industry is, however, not without obstacles. Notwithstanding the significant shift towards BIM tools and digital tools in general in the construction industry, many project participants are still working in a highly document-centric manner where data is stored in a static, fragmented fashion in multiple heterogenous formats (Isikdag et al., 2007; Deshpande et al., 2014). Also, BIM-based workflows are often heavily document- or model-based, as the focus is on the exchange of files for achieving interoperability (BIM Level 2 as described in the PAS 1192 Specification (BSI, 2014)).

To make matters worse, design changes occur rapidly during the design, engineering and construction phases, so substantial rework must be done. Tracking design changes becomes a complicated task as these are often only documented in meeting memos, mail correspondences or, even worse, on a Post-it on the project manager's table or solely in the memory of the project participants (Kiviniemi, 2005; Tserng and Lin, 2004; Deshpande et al., 2014). As a result, the effects of a design change are so opaque that the consequences are hard to judge, sometimes resulting in critical after-effects, also in current BIM-based workflows. Further, substantial information losses occur each time an employee leaves the project (O'Leary, 1998).

Current BIM implementations do improve matters somewhat, as data structure and standardisation challenges are addressed. However, the fact that the majority of the industry is entirely document-centric, still has quite a big impact as well. No matter how much standardisation is put into file formats and exchanges, document-centric approaches result in parsing, interpretation, serialisation and deserialisation workflows that are bound to bring about inefficiencies and errors.

1.2. Shifting from the Exchange of Documents to the Management of AEC Knowledge Graphs

Considering that most documents are simply representations of data, we focus on the following research question in this paper:

How can we effectively store design data in a structured way, allowing interrelated data to maintain their relations intact as the project progresses, without losing the history of properties' progression?

Recent research efforts have proposed the use of semantic web technologies to overcome the document-based attitude and enhance interoperability (Santos et al., 2017). One main component of the semantic web is the design of *ontologies* which are by Studer et al. (1998) defined as *"a formal, explicit specification of a shared conceptualization."* 'Formal' refers to the fact that it must be machine-readable. 'Explicit' means that the concepts used and the constraints on their use are explicitly defined, and 'Shared' entails that it describes consensual knowledge which is accepted by a group. With this work, we investigate existing ontologies in order to satisfy the Data on the Web best practice that consists in reusing existing vocabularies where applicable (Lóscio et al., 2012). We consider what additional terminology is needed to answer the above research question, thereby providing such shared conceptualisation – this terminology, together with the project-specific assertions of a particular project, form what we in this article refer to as the *AEC-KG*.

1.3. Running Example: Calculation of Heating Demands for Spaces in a Building

We define a typical design task, conducted as part of a building design process that the research question can be evaluated against. The particular task is the calculation of heating demands for spaces in a building. The heating demand of a space is calculated for a steady state winter condition specified for the region in which the building is located and it has two components being (1) the infiltration heat loss which is constituted by the undesired ventilation through leaks, and (2) the transmission heat loss through the building envelope. The magnitude of the infiltration heat loss is dependent on the temperature difference between the air in the space and the outdoor as well as the ventilation rate which is typically estimated by the engineer as a function of the space volume. The total transmission heat loss of a space is the sum of the individual building envelope segments that face the space. Each transmission heat loss is dependent on the geometric properties of the segment, the thermal properties of the particular building element

and the temperature difference over the segment. During the design stages, the building's geometry occasionally changes, and this has consequences for both components of the space heat loss. Further, the resulting heating demands define the boundary conditions for the devices that heat up the spaces. These devices further constitute the boundary condition for the heating distribution system and hence all its sub-components. Therefore, this design task involves multiple information exchanges, and in a design practice with successive design iterations, it is a challenge that these exchanges are not handled dynamically. As the project evolves, this leads to inconsistencies, and it becomes a labour intensive task to assess the consequences.

1.4. Overview of the Outline of the Article and the Main Contributions

In this article, we will first give a brief overview of the state of the art in the use of semantic web technologies in the AEC industry along with an introduction to this topic (Section 2). Then, in Section 3, we will explain the design of the Ontology for Property Management (OPM) and give brief examples and indications of how the ontology, in combination with existing ontologies, can be used to keep track of the history, reliability and provenance of a property of some Feature of Interest (FoI). With FoIs, we refer to anything of relevance in a building to an AEC expert. This includes either spatial elements (spaces, zones, storeys), physical elements (walls, windows, heaters, sensors) or abstract elements (interfaces, systems, concepts). The core of OPM dealing with property change management was already presented in Rasmussen et al. (2018b), but we extend it here also to provide terminology for describing property reliability, derived properties and calculations for formalising reasoning logic (Section 3). The ontology extensions for describing property reliability and calculations along with best practice modelling examples are the first and main contribution of this work. The second contribution is the specification of a standardised way to generate parametric queries for managing an OPM-compliant AEC-KG. In Section 4, considerations concerned with this management is discussed in detail and in Section 5, a JavaScript-based Application Programming Interface (API) that facilitates the creation of uniform, reliable queries for retrieving, creating and updating OPM properties and calculations is presented. The API provides a crucial addition to the proposed OPM ontology. Namely, considering the expressiveness and complexity of OPM, end-user applications wishing to implement the proposed property management need a middeware that allows them to define the desired queries towards an OPM-compliant AEC-KG. The OPM Query Generator (OPM-QG) provides a reference implementation of such a middleware API. In Section 6, we demonstrate a Proof of Concept (PoC) implementation built on top of the OPM infrastructure which assesses the practical design case described above. The architecture and considerations in this regard are discussed in detail in this section. Finally, we conclude the work and present our visions for future work.

2. State of the Art

In this section, we first investigate existing research efforts and communities dealing with describing AEC knowledge in graphs. We then look into different approaches to property assignment and what benefits each of these possesses. Lastly, we look into existing efforts in dealing with the handling of property interdependencies and deduction of implicit knowledge from BIM models.

2.1. Web Ontologies for AEC Knowledge Graphs

Researchers in the linked data and semantic web domain have recently aimed at making building data available on the web, linking data rather than documents (Curry et al., 2013; Pauwels et al., 2018). This group of researchers in the AEC domain has been gathering behind the recent initiatives around linked data in architecture and construction, which includes the Linked Data Working Group (LDWG) in buildingSMART International (bSI)[1], and the Linked Building Data (LBD) Community Group[2] at the World Wide Web Consortium (W3C). In both standardisation bodies, ontologies are proposed for capturing building data using web technologies. The groups focus on the use of semantic web technologies, namely the Web Ontology Language (OWL) and the Resource Description Framework (RDF) (W3C OWL Working Group, 2012;

[1] `http://www.buildingsmart-tech.org/future/linked-data`

[2] `https://w3.org/community/lbd/`

Cyganiak et al., 2014), thus creating smaller aligned ontologies and putting them on a track towards standardisation. Aligned in this regard meaning that they extend or comply with terminology from the other ontologies.

buildingSMART is the standardisation organ who maintains the standard exchange format for BIM data models: the Industry Foundation Classes (IFC) standardised by ISO 16739 (ISO16739, 2013).

The LDWG remains entirely in the buildingSMART realm, focusing mainly on the production of the ifcOWL ontology[3] (Pauwels and Terkaj, 2016), which was initially proposed by Beetz et al. (2008). The ifcOWL ontology is set up to be a direct translation from the IFC schema represented in the EXPRESS data modelling language (ISO10303-11, 2004) into an OWL representation. This has resulted in an extensive ontology, much unlike many of the existing ontologies in various other domains, that are typically more narrow scoped and depend on extensions enabled by linked data principles. For this reason, there have also been various attempts on simplifying this ontology. For example, IFCWoD (IFC Web of Data) (Mendes de Farias et al., 2015), SimpleBIM (Pauwels and Roxin, 2016), and BimSPARQL (Zhang et al., 2018) all aim at providing simplified views over ifcOWL data. IFCWoD implements a set of rules within the triple store; SimpleBIM implements rewrite rules in code; and BimSPARQL simplifies ifcOWL data by extending the query language for RDF, SPARQL (Harris et al., 2013), with rewrite rules and geometry calculation algorithms. None of the simplification approaches proposes and defines an explicit OWL ontology.

The W3C LBD Community Group, on the other hand, has focused mostly on the creation of ontologies for capturing building information from close to scratch. By starting from close to scratch, ontology engineering best practices can be more easily maintained. For example, it is possible to reuse existing ontologies and develop minimal extensions in a modular fashion. This is possible because RDF uses International Resource Identifiers (IRIs) to denote resources (i.e. something in the world) (Cyganiak et al., 2014). The Linked Data rules (Berners-Lee, 2006) further requires the use of HyperText Transfer Protocol (HTTP) IRIs like the ones used when browsing the web.

[3] `http://www.buildingsmart-tech.org/ifcOWL/IFC4#`

The W3C group currently focuses on the development and maintenance of a Building Topology Ontology (BOT) (Holten Rasmussen et al., 2018) a PRODUCT ontology, and a PROPS ontology; with BOT being the most central one of those. Each of these ontologies captures a subdomain of data, being respectively building topology, products, and properties. They provide terminology to describe data in standardised structures, not placing any restrictions on the actual representation format or file format used.

2.2. Three Levels of Complexity for Design Property Descriptions

When assigning properties to some FoI, there are different considerations to illuminate. Is there a need to assign a physical unit? Is it necessary to capture metadata such as which property set a certain property belongs to? Is there a need to capture provenance data, such as, who created the property or what other properties or processes it was derived from? Should it be possible to change the value of this property, and in this case, should some record of state changes be maintained?

Bonduel (2018) provides a visual presentation, thereby comparing different modelling approaches for properties used in ifcOWL, IFCWoD and SimpleBIM. SimpleBIM uses the most straightforward approach by simply describing properties defined as OWL Data Properties (W3C OWL Working Group, 2012, §5.4) with a literal value assigned (Figure 1). IFCWoD relates the FoI to a property set which in turn describes each associated property as an objectified property assigned with an OWL Object Property (W3C OWL Working Group, 2012, §5.3). The objectified property then holds the value of the property described using EXPRESS datatypes like it is done in ifcOWL (Figure 1).

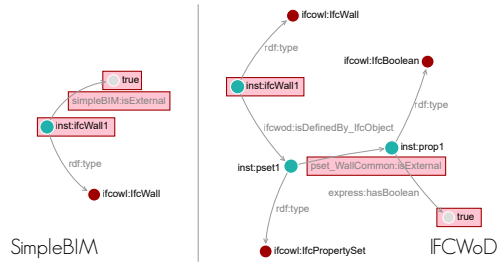


Figure 1: SimpleBIM and ifcWoD property assignment visualized using SPARQL visualizer[4].

Listing 1 shows how SimpleBIM and IFCWoD properties are encoded in the Turtle (Prud'hommeaux et al., 2014) serialisation format for RDF data models. In RDF, all statements are described in triples consisting of a *subject*, a *predicate* and an *object*. The first triple in the listing describes the instance (the *subject*) `inst:ifcWall1` which by the *predicate* `rdf:type` is assigned to the class (the *object*) `ifcowl:IfcWall`. The period marks the end of a triple. All IRIs are prefixed but the namespaces of the prefixes are not shown in the listing. `inst:` is a general prefix used to describe instances (in any namespace) and all other prefixes relate to ontologies that are either described in this section or discoverable at http://prefix.cc/{prefix}. The object of the second triple is a literal value. The IFCWoD triples use the 'a' token at the predicate position which is short for `rdf:type`. Further, it uses semicolon ';' to indicate that the subsequent triple concerns the same subject.

Listing 1: Property assignment in Turtle syntax.

```
# SimpleBIM
inst:ifcWall1 rdf:type              ifcowl:IfcWall .
inst:ifcWall1 simpleBIM:isExternal  "true" .

# IFCWoD
inst:ifcWall1 a ifcowl:IfcWall ;
              ifcwod:isDefinedBy_IfcObject inst:pset1 .
inst:pset1    a ifcowl:IfcPropertySet ;
              pset_WallCommon:isExternal inst:prop1 .
inst:prop1    a ifcowl:IfcBoolean ;
              express:hasBoolean "true" .
```

The most complex approach is used in ifcOWL (Figure 2), and this illustrates very well the relics of the complex IFC schema that are part of this ontology, because of the imposed backward compatibility constraint. The property is also here assigned through a property set, but the relationship between the FoI and the property set requires an intermediate node of type ifc:IfcRelDefinesByProperties. The property is also not directly assigned to the property set but requires an intermediate node of type ifc:IfcPropertySingleValue which refers to two different nodes holding the name and the value of the property - again as objectified properties holding the literals described using EXPRESS datatypes.

Examining the different approaches for property assignment reveals a high variance in complexity. The most simplistic form, direct assignment

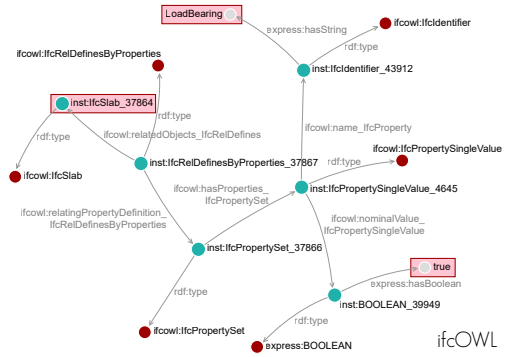


Figure 2: ifcOWL property assignment visualized using SPARQL visualizer[4].

of datatype properties, reduces the complexity of the queries and thereby makes it easier to navigate the graph, but it also sacrifices the opportunity of adding metadata.

Rasmussen et al. (2018b) describes three levels of complexity, where each level refers to the number of steps between the FoI and the actual object (literal or individual) that encodes the value of its property. L1 is equal to the approach used by SimpleBIM. L2 describes the property as an object, and thereby allows attaching metadata such as a unit of measure using a dedicated ontology like Quantities, Units, Dimensions and Types (QUDT) (Hodgson and Keller, 2011) or provenance data using the PROV Ontology (PROV-O) (Lebo et al., 2013). L3 is used by OPM and is inspired by the Smart Energy-Aware Systems (SEAS) evaluations Lefrançois (2017). By assigning multiple property states to one property, L3 property assignment allows the property value to change over time while keeping a record of how it evolved. The property assignment complexity in ifcOWL exceeds L3 by magnitudes but does not add functionality other than backward compatibility with IFC.

2.3. Handling Interdependent Properties

Isaac et al. (2013) suggest handling the complexity of construction projects by modelling the projects' topology in graphs. Other recent research projects have further illustrated how relationships in building data beyond the geometrical ones can be handled by the use of semantic web technologies. This further introduces the capability of using reasoning engines to infer implicit information

[4] `https://github.com/MadsHolten/sparql-visualizer`

that is not directly asserted in the AEC-KG, but that can be deduced from the facts that are present, thereby making them explicit facts. This technology is heavily used to make machines capable of 'reading between the lines' to provide enhanced results from search engines and so forth.

Deducing implicit facts from prose text using description logic is not remarkably different from the work of an engineer who puts various inputs into equations in order to generate new outputs. This is also in accordance with the analogy by Winch (2010) which compares construction project teams to information processing systems. The problem is that the information processing systems are today constituted by the knowledge workers and the software tools they use. This challenge was studied by Pauwels et al. (2011) who considers the use of rule-checking environments to formalise the knowledge of the human workers. In particular, semantic web technologies are used, to establish an AEC-KG consisting of (1) explicit building information parsed from an IFC file, (2) an ontology parsed from the IFC schema, and (3) a set of rule-sets describing implicit engineering knowledge. The concrete case of performing compliance checking of acoustic performance is presented with this work as a proof of concept. This novel approach demonstrates how a typical task for a knowledge worker can be explicitly described in reuseable rule-sets, and thereby it represents an entirely different data-driven approach to a labour-intensive job. A more recent study by Zhang et al. (2018) uses a similar approach with the purpose of (1) providing shortcuts to make an ifcOWL graph easier to query and (2) deduce geometric information from 3D geometry. The baseline of both studies is a final BIM model, and thereby it differs from the situation investigated with this work.

Both Pauwels et al. (2011) and Zhang et al. (2018) use rules to infer results at run-time. Providing derived properties at run-time rather than materialising them in the graph has the benefit that outdated or redundant information is avoided. Zamanian and Pittman (1999), however, argues that since AEC projects are performed by distributed teams, it is often desirable to have some consistent redundancies that are designed and managed to provide more efficient means to access and manipulate the information. Some party might wish to refer to an intermediate result of a derived property, and this is not immediately possible if that information is only described in a rule. Also, with OPM, the intention is that interdependencies are explicitly stated so that the knowledge workers are provided with insights. The initial work presented by Rasmussen et al. (2018b) suggests that every state of a property is saved in order to evaluate design changes, and this is only possible when materialising reasoning results.

3. An Ontology for Property Management (OPM)

The Ontology for Property Management (OPM) was initially introduced by Rasmussen et al. (2018b). It provides terminology for modelling complex properties in a design environment. Complexity, in this regard, entails that they (1) change over time (2) can be assumptions and hence comprise varying reliability, and (3) can be dependent on the value of other properties . The first contribution demonstrated how property assignment modelled according to this ontology can be used to answer a set of competency questions that are all concerned with managing evolving properties. Section 3.1 first overviews the concepts of the OPM ontology that was presented in the initial OPM paper. Then subsequent sections define additional sets of competency questions that deal with reliability (Section 3.2) and handling of interdependencies between properties (Section 3.3). We propose a set of new concepts extending the initial version of OPM to provide terminology to answer these questions. Like the original OPM ontology, the extensions depend on concepts from the SEAS Lefrançois (2017), schema.org and PROV-O ontologies and align well with the BOT, PROPS and PRODUCT ontologies of the W3C LBD Community Group. OPM uses the namespace `https://w3id.org/opm##` and the documentation is provided when this IRI is visited in a web browser.

3.1. Property History

Rasmussen et al. (2018b) described seven competency questions, all dealing with property history modelling are listed below.

CQ 1.1 How to semantically describe a property such that its value is changeable while its historical record is maintained?

CQ 1.2 How to revise a property value?

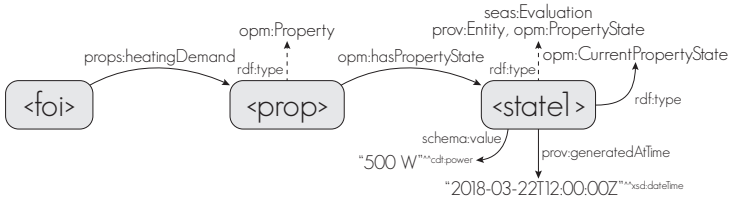


Figure 3: L3 property assignment using an opm:PropertyState for assigning the latest value and metadata describing when it was created (Rasmussen et al., 2018b).

CQ 1.3 How to delete a property while still being able to retrieve the history of it and not break all the links to derived properties that depend on it?

CQ 1.4 How to restore a deleted property?

CQ 1.5 How to retrieve the full history of how the value of a property has evolved over time?

CQ 1.6 How to retrieve only the latest value of a property?

CQ 1.7 How to simplify a complex OPM property (using states) for easier and faster querying?

To answer these questions, the authors describe three different levels of property assignment with varying expressivity. The level in this regard refers to *"the number of steps/relations between the FoI and the actual object (literal or individual) that encodes the value of its property."* The most expressive level, L3, is used in OPM to capture property changes over time. It uses the concept of property evaluations from SEAS. Figure 3 illustrates how the opm:PropertyState (subclass of seas:Evaluation) can be used to capture a state of a property. It is assigned to a property using the opm:hasPropertyState (sub-property of seas:evaluation) predicate, and the rdfs:range of this predicate implies that it belongs to the opm:PropertyState class. OPM depends on schema:value, schema:minValue and schema:maxValue to assign a value to a property state, and, as a minimum, the state must further have a prov:generatedAtTime predicate. Retrieving the most recent state of a property can be achieved by querying for the highest prov:generatedAtTime value, but this (1) increases the query complexity and (2) reduces query performance (Rasmussen et al., 2018b). Therefore, the opm:CurrentPropertyState class is always assigned to the most recent state and removed from the previous state when performing SPARQL (Harris et al., 2013) update queries on the AEC-KG. The opm:OutdatedPropertyState class can, in this case, be assigned to the outdated property state.

A different design pattern which is also supported by OPM is property assignment by classification. In this case, a generic property such as `props:hasProperty` is used as predicate and the object (the property) is classified according to the type of property. For example: `<foi> props:hasProperty <prop> . <prop> rdf:type props:NominalUA .`

As illustrated in Figure 4, changing a property's value is handled by assigning a new property state holding the new value and other metadata such as provenance. Thereby, it is possible to retrieve the full history of a property and restore previous properties if necessary. Changing and restoring properties can be handled with standardised SPARQL update queries that are executed against the AEC-KG by client applications.

In order to maintain the history of the project evolution and to be able to revert to an earlier stage, data should never be removed from the AEC-KG. Marking a deleted property state as an instance of both opm:CurrentPropertyState and opm:Deleted provides filtering options, and thereby the state can be stored while hidden from end users. A deletion is reverted by inferring a new state that inherits the properties of the most recent state with a value assigned to it. Having both the initial state, the deleted state and the restored state available allows for tracking when and by whom the different changes were conducted.

3.2. Property Reliability

The AEC industry is a complex industry where each project is constantly changing during the design and planning stages. As part of this work, we had discussions with industry professionals who described working methodologies that are dependent

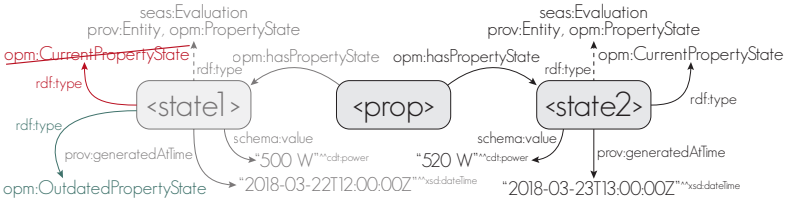


Figure 4: Changing a property is handled by assigning a new property state to the property and removing the opm:CurrentPropertyState from the state that was previously defined as the current property state. Assigning the previous state as opm:OutdatedPropertyState is optional (Rasmussen et al., 2018b).

on being able to assess the quality of any data in the project. As a result, we defined the following set of additional competency questions for property reliability:

CQ 2.1 How to describe that the value of some property is defined temporarily until the actual value is known?

CQ 2.2 How to document that some property has been confirmed and can hence be trusted not to change in the future?

CQ 2.3 How to describe that some property is derived from, and hence dependent on the value of some other property?

CQ 2.4 How to describe a property requirement?

All these questions require terminology that was not initially part of OPM, but in the following, each question will be answered by adding a specific class to the ontology.

CQ 2.1, How to describe that the value of some property is defined temporarily until the actual value is known?. OWL class opm:Assumed
In the early stages of any construction project, it is common practice to make temporary assumptions in order to progress with the design. Assumptions are typically managed in assumption lists, conducted as simple documents such as a spreadsheet, and it is the project manager's job to make sure that all assumptions are later clarified and confirmed by a person in charge of the specific domain. An assumption is thus defined here as a non final value with a high probability of changing over time as the project evolves. Classifying a property state as opm:Assumed describes that the value is temporary and must later be confirmed. A querying for all the states that belong to this class reveals which properties are yet to be confirmed.

CQ 2.2, How to document that some property has been confirmed and can hence be trusted not to change in the future?. OWL class opm:Confirmed
A confirmed value is approved by a person who has the authorisation to do so. Classifying a property state as opm:Confirmed indicates that its value can be trusted not to change in the future. For legal documentation purposes, a digital signature can be assigned to the property. Also, a link to documentation such as a mail, a scanned contract or similar can be attributed to the property state using the opm:documentation predicate. As it is common practice to set milestones in the course of a construction project, at which certain parameters are locked, these are obvious opportunities to mark all related object properties as confirmed. An example of a milestone is the date where the layout of the superstructure is frozen and concrete elements are ordered from the manufacturer. Changing the design after this date is possible, but it will likely lead to a cost penalty that someone needs to pay.

CQ 2.3, How to describe that some property is derived from, and hence dependent on the value of some other property?. OWL Class opm:Derived
Engineering is chiefly a discipline of gathering information, processing that information, typically by applying math and physics, and thereby deducing new information. A simple example is the area of a window, which is either directly deduced from its geometrical definition or by a product of the height and width properties. If the value of a property is dependent on other properties, it should have the opm:Derived class applied. Keeping derived properties up to date can be automated, but in many occasions it is desirable for the engineer to keep

the design as is, knowing that the property is no longer valid. Only when the consequence of the change is significant, the design is revised. It might also be that the consequences are not acceptable, and in this case, the party who made the initial change must instead be asked to revert the change. The engineer can be supported in this decision by dynamically deriving the new result while calculating the deviation from the static result. Also, it can be explicitly stated that the state belongs to the class opm:OutdatedPropertyState. Next section deals specifically with the handling of derived properties.

CQ 2.4, How to describe a property requirement?. OWL class opm:Required
Requirements can be assigned to abstract product or space models holding the prerequisites for a design. Use cases for this include space schemas holding functional requirements of the spaces of a future building, design models holding functional requirements for mechanical equipment, and so forth. Rasmussen et al. (2018a) implements this feature to compare client requirements to a building with the actual design. It is thereby possible to query the AEC-KG for all properties that do not fulfil the requirements that have been defined. OPM allows both requirements and design values to change over time, and therefore, the consequence of a violated requirement must ultimately be that either the requirement itself or the violating design value needs to adapt.

3.3. Property Interdependence

The opm:Derived class enables to describe that some property is derived from one or more other properties. There are, however, more things to consider when dealing with interdependent properties such as traceability and quality assurance. We defined the following set of additional competency questions to capture these:

CQ 3.1 How to associate a derived property to the properties from which it was derived?

CQ 3.2 How to identify that a derived property is outdated?

CQ 3.3 How to formally describe a calculation that can be applied to infer derived properties?

CQ 3.4 How to associate a derived property to the calculation or algorithm that formalises how it was derived?

CQ 3.5 How to check for circular dependencies in derived properties?

CQ 3.6 How to define the reliability of a derived property?

CQ 3.7 How to check which derived properties will be affected if a specific property is changed?

OPM does not restrict how derived properties are inferred and whether this is accomplished with runtime inference or by materialising the derived properties in the graph. It does, however, define a best practice approach for modelling interdependencies in a way that satisfies the answering of above competency questions. A derived property is modelled like any other OPM property, but it is classified as opm:Derived and its value is inferred instead of being typed.

CQ 3.1, How to associate a derived property to the properties from which it was derived?.
In order to associate a derived property with the properties from which it was derived each state of the derived property must be linked to the states of the properties from which it was derived (its arguments). Figure 5 illustrates how this is achieved using the prov:wasDerivedFrom predicate.

CQ 3.2, How to identify that a derived property is outdated?.
The most recent state of a derived property is classified as opm:CurrentPropertyState. Since this state is related to the states of the properties from which it was derived using the prov:wasDerivedFrom predicate it is possible to check whether these states are also classified as opm:CurrentPropertyState. If this is not the case, the derived property is outdated.

CQ 3.3, How to formally describe a calculation that can be applied to infer derived properties?. OWL class opm:Calculation
OPM includes the concept of calculations which formalises the specification of the reasoning logic. An instance of the opm:Calculation class holds such specification. As a minimum, a calculation must describe the IRI of the inferred property using predicate opm:inferredProperty, an expression using predicate opm:expression and a path from the FoI to each of the arguments using predicate opm:argumentPaths. The latter is described as an RDF list where each path is stated as a literal (See

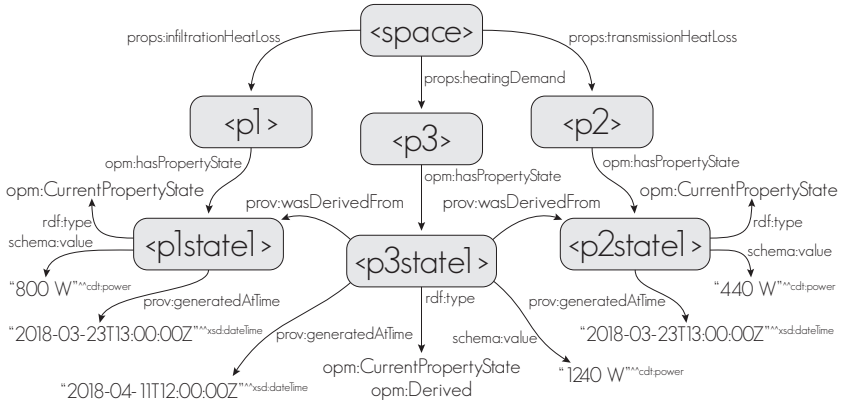


Figure 5: When inferring derived properties, they should be connected to the property states from which they are derived using a prov:wasDerivedFrom predicate.

example in Listing 2). Figure 6 shows the calculation that inferred the derived props:heatingDemand property from Figure 5. The calculation describes the heating demand as the sum of the infiltration heat loss and the transmission heat loss and since both these properties are directly assigned to the space itself, the calculation is relatively simple. The opm:expression defines the result as the sum of the two variables `?htr` and `?inf`. The paths from the space to the two arguments are defined as `"?foi props:transmissionHeatLoss ?htr"` and `"?foi props:infiltrationHeatLoss ?inf"`. The number of argument paths must correspond to the number of variables, but the paths can be extended to restrict the results further. For example it can be specified that the FoI must be an instance of bot:Space by extending the first path to `"?foi a bot:Space ; props:transmissionHeatLoss ?htr"`. Listing 2 shows how Figure 6 is described using the Turtle syntax for RDF.

Listing 2: Calculation from Figure 5 in Turtle syntax.

```
inst:c1  rdf:type                opm:Calculation .
inst:c1  opm:expression          "?htr+?inf" .
inst:c1  opm:inferredProperty    props:heatingDemand .
inst:c1  opm:argumentPaths (
           "?foi props:transmissionHeatLoss ?htr"
           "?foi props:infiltrationHeatLoss ?inf" ) .
```

The opm:expression is preferably defined in SPARQL 1.1 syntax, which includes methods that are sufficient for defining the simple calculations that are extensively used in engineering. Since the

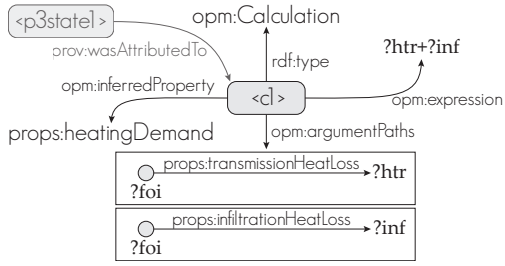


Figure 6: Describing a calculation with OPM. The dimmed part in the upper left corner shows how states inferred by the calculation can be attributed to the calculation from which they were inferred. `<p3state1>` refers to the inferred state illustrated in Figure 5.

expression is assigned as an OWL Data Property, it is also possible to describe more complex expressions. Expressions are expected to be encoded in the SPARQL 1.1 syntax, but other languages such as Javascript can be used as well. In this case, the datatype of the expression should be the JavaScript mediatype IRI iana:application/javascript[5].

The algorithm in Figure 7 shows the intended use of OPM calculations to generate reasoning results. The first step is to retrieve the calculation data. If the expression is not defined using a special datatype, it is expected to be described in SPARQL

[5] `https://www.iana.org/assignments/media-types/application/javascript`

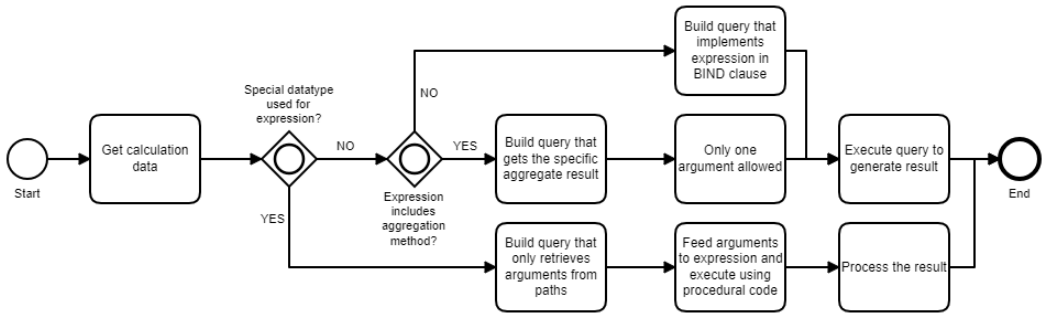


Figure 7: Algorithm to yield results from a opm:Calculation.

syntax. In this case, the expression is analysed, and if it contains an aggregation function such as sum, min, max, avg or count, an aggregation sub-query is constructed based on the single argument given by the opm:argumentPaths property. If it does not contain an aggregation function, the latest state of each argument is retrieved, and the expression is applied to the query using a BIND form, which allows the assignment of a value to a variable[6]. The naming of the variables used in the argument paths must match the arguments used in the expression (e.g. `?htr` and `?inf` in Listing 2). Listing 3 shows some examples of SPARQL 1.1 expressions that can be used in expressions. The result of the three first expressions can be bound to a `?result` variable using a BIND form. The latter requires an aggregation sub-query which can bind the result to a `?result` variable.

If the expression is some JavaScript procedural code, the arguments must first be retrieved using a query, and afterwards, the result can be calculated. This is the only valid approach if more complex calculations like simulations are needed.

Listing 3: Four example expressions for the opm:expression datatype property.

```
# Math expression
"math:sqrt( math:pow(?arg1, 2) + math:pow(?arg1, 2) )"
# String operation
"STRAFTER(?arg1,'/')"
# Conditional expression
"IF(?arg1 > ?arg2, 'true', 'false')"
# Aggregation expression
"SUM(?area)"
```

CQ 3.4, How to associate a derived property to the calculation or algorithm that formalises how it was derived?.
In order to associate a derived property with the calculation from which it was derived, a derived property should be linked to the particular opm:Calculation instance using the prov:wasAttributedTo predicate. This is illustrated in Figure 6.

CQ 3.5, How to check for circular dependencies in derived properties?.
Properties can be derived from properties which are themselves derived, and so forth. Since all property states are interlinked with the prov:wasDerivedFrom predicate, it is possible to retrieve the full set of property states from which a property is derived. A property cannot be derived from itself, and in order to check that no circular dependencies exist in the graph, the query from Listing 4 can be used. This query uses a SPARQL property path (OneOrMorePath, (Harris et al., 2013)) to ask for all arguments from which a property state is derived, and a filter to only return results where the state is dependent on itself.

Listing 4: Query to check for circular dependency.

```
SELECT ?propSt
WHERE {
    ?propSt prov:wasDerivedFrom+ ?arg .
    FILTER(?propSt = ?arg)
}
```

[6] https://www.w3.org/TR/sparql11-query/#bind

CQ 3.6, How to define the reliability of a derived property?.
The reliability of a derived property should also not be specified manually, but should instead be inferred from the properties from which it is derived (the arguments). If one or more of these belong to the class opm:Assumed, the same is the case for the derived property. Likewise, if an argument is of type opm:Deleted, the derived property is also no longer valid (i.e. deleted). Inheritance of the opm:Confirmed class to a derived property requires all the arguments to be confirmed.

CQ 3.7, How to check which derived properties will be affected if a specific property is changed?.
In a construction project, it is practically impossible to know the derived properties that other parties might have created. With OPM, however, the derived properties are related to the property from which they are derived, and hence it is possible to check what derived properties are affected when changing a property.

4. Reasoning with the OPM Ontology

This section describes the use of two main components of the semantic web in relation to OPM: reasoning (Section 4.1), and use of named graphs (Section 4.2). It further describes the use of SPARQL queries for inferencing (Section 4.3).

4.1. Materialising Derived Properties

Dealing with derived properties can be achieved either by inferencing upon request (deduce results at runtime) or by materialisation (saving the results) in the AEC-KG. Below, we justify why we recommend the latter approach for OPM. Table 1 compares the two approaches qualitatively.

Table 1: Runtime vs. materialised inferencing.

	Runtime	Materialised
Validity	●	◑
Performance	○	●
Traceability	○	●

Validity. When materialising the results of derived properties, there will at some point exist data that is no longer valid. Inferencing upon request will always provide the correct result based on the most recent state of all arguments. It is, however, possible to check whether a derived property is outdated. This check could be automated in order to infer the opm:OutdatedPropertyState class to the property states that are no longer valid using the logics described in *CQ 3.2*.

Performance. As properties can be derived from other derived properties, the dependency chain can become long, and hence the reasoning engine needs to loop over the data several times to saturate the graph. The resulting performance drop can be a significant drawback for inferencing upon request. As complexity grows, performance in materialising derived triples will also decrease, but this task can be performed in the background, thereby not sacrificing the user experience.

Traceability. Materialising every single state of a derived property and the specific states of the properties from which it was derived, provides valuable insights. This increases transparency and allows for more in-depth analyses of embedded consequences of particular changes. It is easy to imagine that the management of interdependencies will eventually become a chaotic task when everything is dynamic and automatically updating as the design changes. In a construction project, it might for some reason be desired to stick to the value as it is, knowing that the inputs have changed slightly for some reason not known to the reasoner. The missing traceability when having a calculation performed at runtime furthermore entails some legal implications concerning responsibility.

4.2. Separation of Explicit and Inferred Triples

To distinguish inferred triples from explicit triples, they can be stored in separate named graphs in the same database. Carroll et al. (2005) describes several purposes for named graphs. With this work, we suggest that they are used here to separate explicit triples from derived triples. The triples that are inferred from those in graph `IRId` are then stored in a second named graph denoted `IRId-I`. This provides a mechanism to remove all inferred triples and re-establish them from the most recent state of all arguments in cases where the history of derived properties is not important.

4.3. Inferencing with SPARQL queries

Materialising derived properties based on calculation data can be achieved with SPARQL update queries. Listing 5 shows a SPARQL update query that will append derived properties based on the calculation shown in Listing 2. The query appends the derived property props:heatingDemand in the graph of inferred triples <https://host/project−I> for all FoIs that satisfy the argument paths specified in Listing 2, but only if the property is not already assigned. The query further adds the opm:Derived and opm:Assumed classes to the newly generated property state, and the generation time.

As a matter of fact, this query may be generated automatically from the opm:Calculation instance of Listing 2 using the algorithm shown in Figure 7.

Materialised derived properties may become outdated, as they depend on arguments that could potentially change after materialisation. A similar query may be constructed to update derived properties where one or more of the arguments have changed.

These challenges may be addressed using incremental reasoning Barbieri et al. (2010), or defeasible reasoning approaches Antoniou and Bikakis (2007).

5. An API to interact with OPM Data

In the description of CQ 3.3 in Section 3.3 we described the algorithm illustrated on Figure 7 for automatically generating reasoning results from opm:Calculation instances. Listing 5 illustrates that parametric SPARQL queries can be used to work with OPM properties in general. This section introduces an API to interact with OPM data using parametric SPARQL queries.

5.1. Interacting with Properties through Parametric Query Generation

The OPM Query Generator (OPM-QG)[7] is a JavaScript library for simplifying the task of writing queries for doing Create, Read, Update and Delete (CRUD) operations on an OPM-compliant AEC-KG. This library eases the access to the concepts introduced in Sections 3 and 4 for people not so familiar with RDF. Further, it provides standardised methods for creating the complex queries (Listing 5), thereby ensuring that no unintended operations are performed on the graph.

Listing 5: SPARQL query to append a calculation.

```
PREFIX opm: <https://w3id.org/opm#>
PREFIX prov: <http://www.w3.org/ns/prov#>
PREFIX schema: <http://schema.org/>
PREFIX props: <https://w3id.org/props/>

INSERT {

  # insert in the graph of inferred triples
  GRAPH <https://host/project-I> {
    ?foi props:heatingDemand ?propertyIRI .
    ?propertyIRI opm:hasPropertyState ?stateIRI .
    ?stateIRI a opm:CurrentPropertyState ,
                  opm:Derived , opm:Assumed ;
        schema:value ?res ;
        prov:generatedAtTime ?now ;
        prov:wasDerivedFrom ?state1 , ?state2 .
  }

}
# using both the explicit and the inferred triples
USING NAMED <https://host/project>
USING NAMED <https://host/project-I>
WHERE {

    # get argument 1
    GRAPH ?g1 {
        ?foi props:transmissionHeatLoss ?htr_ .
        ?htr_ opm:hasPropertyState ?state1 .
        ?state1 a opm:CurrentPropertyState, opm:Assumed ;
            schema:value ?htr .
    }

    # get argument 2
    GRAPH ?g2 {
        ?foi props:infiltrationHeatLoss ?inf_ .
        ?inf_ opm:hasPropertyState ?state2 .
        ?state2 a opm:CurrentPropertyState, opm:Assumed ;
            schema:value ?inf .
    }

    # check that property state is not already inferred
    MINUS {
        GRAPH <https://host/project-I> {
            ?foi ?inferredProperty ?prop
        }
    }

    # perform calculation
    BIND((?htr+?inf) AS ?res)

    # create state and property IRIs
    BIND(IRI( CONCAT("https://host/project/",
        "state_", STRUUID() ) ) AS ?stateIRI)
    BIND(IRI( CONCAT("https://host/project/",
        "property_", STRUUID() ) ) AS ?propertyIRI)

    # get current time
    BIND(now() AS ?now)
}
```

[7] Query Generator - https://www.npmjs.com/package/opm-qg

Since the API is built in JavaScript, queries can be constructed and executed either directly from a web client application or a NodeJS[8]-based server-side application.

OPM-QG is divided into two interfaces, where one deals with properties and the other one with calculations: OPMProp and OPMCalc (See Figure 8, as well as Sections 5.2 and 5.3). Each interface contains a set of methods that return SPARQL queries (e.g. from Listing 2 to Listing 5).

	Create	Read	Update	Delete
OPMProp	postProp() postClassProp()	getProps()	putProp() restoreProp()	setReliability()
OPMCalc	postCalcData() postCalc()	getCalcData() getOutdated() getSubscribers()	putCalc()	
Interfaces	Methods			

Figure 8: OPM-QG interfaces and methods.

OPM-QG supports the separation of explicit and inferred triples in two named graphs as described in Section 4.2, but can also construct queries that operate only on the main graph. When instantiating one of the two interfaces from Figure 8, it is defined what `host` IRI is to be used. This information is necessary when constructing IRIs for new resources. The two BIND forms in the query illustrated in Listing 5 (lines 52-55) show how OPM-QG creates new IRIs as a concatenation of `{host}/{type}_{UUID}` where variable `{type}` is typically either *state* or *property*.

All read queries can be generated either as SELECT or CONSTRUCT queries and create, update and delete queries can be generated either as INSERT or CONSTRUCT queries. The API can thereby be used for both runtime inferencing and materialising derived triples. The contained methods answer to most of the competency questions described in Section 3.

In the following subsections, it is described how OPM-QG can be used to generate parametric queries for accessing and manipulating properties and calculations respectively (Sections 5.2 and 5.3).

5.2. Properties

The OPMProp interface provides methods for dealing with OPM properties. In the following, the methods illustrated in Figure 8 are described in detail.

Create. Properties can be assigned either as instance properties or as property restrictions to an owl:Class instance. The latter approach was used by Rasmussen et al. (2018a) to specify space requirements at type level that would then be inherited by all instances of that class. Listing 6 shows an example where a property restriction is applied to a project-specific wall class. The property restriction is for the U-value (props:thermalTransmittance), and it restricts its value to a specific property, `inst:heavyWall_r1_s1`. This property is inherited by all instances of the wall class and it can be changed using the general OPM principles, since the property is described with an OPM property state.

Listing 6: OWL property restriction.

```
# project-specific class with property restriction
inst:heavyWall rdfs:subClassOf prod:Wall ;
    rdfs:subClassOf inst:heavyWall_r1 .
# property restriction
inst:heavyWall_r1 rdf:type owl:Restriction ;
    owl:onProperty props:thermalTransmittance ;
    owl:hasValue inst:heavyWall_r1_s1 .
# first state of property restriction
inst:heavyWall_r1_s1 rdf:type opm:CurrentPropertyState ,
        opm:Required ;
    schema:value "0.21 W/(m2.K)"^^cdt:ucum ;
    prov:generatedAtTime "2018-10-31T.."^^xsd:dateTime .
```

OPM-QG includes two methods `postProp()` and `postClassProp()` that generates queries for assigning a new property and an associated property state to some FoI. The methods take the IRI of the new property and the value as arguments. For both methods, the instance/class to which the property should be assigned, can be specified by providing the IRI of a specific FoI or by providing a triple path that must return a match. The path is described like the opm:argumentPaths from Listing 2. Optionally, a `reliability`, a `userIRI` and a `comment` can be provided.

Providing the object shown in Listing 7 to the `postProp()` method returns the query shown in Listing 8. It is important to note that the path is commented out. Either a `foiIRI` or a `path` must be provided. Also, the last three attributes are optional, and leaving them out would have omitted lines 5, 7-8 and 20-21 from the query. Supplying a `path` instead of the `foiIRI` would have omitted line 19 and replaced line 24 with the `path`. Since it is a create method, it should only apply the new property to the space if it does not already have the property assigned.

[8] https://nodejs.org/

Listing 7: Input object to `postProp()` or `postClassProp()`.

```
{
    foiIRI:     'https://host/project/space_1',
    //path:     '?foi a bot:Space',         //alternative
    property:   'props:area',
    value:      '"20 m2"^^cdt:area',
    comment:    'Just a test',              //optional
    userIRI:    'https://niras.dk/mhra',    //optional
    reliability: 'assumed'                  //optional
}
```

Read. The generic method, `getProps()`, can be used to get all properties in the AEC-KG. The result can, however, also be restricted by providing a specific `foiIRI` and/or a specific `propertyType` and/or a specific `propertyIRI`. The results can also be restricted to only include the latest property state or property states with a specific restriction, such as all deleted properties.

Listing 8: Result when providing Listing 7 to `postProp()` (main graph).

```
CONSTRUCT {
    ?foi props:area ?propertyIRI .
    ?propertyIRI a opm:Property ;
        opm:hasPropertyState ?stateIRI .
    ?stateIRI a opm:Assumed .
    ?stateIRI a opm:CurrentPropertyState ;
        prov:wasAttributedTo ?userIRI ;
        rdfs:comment ?comment ;
        schema:value ?val ;
        prov:generatedAtTime ?now .
}
WHERE {
    # create state and property iris
    BIND( IRI( CONCAT( "https://host/project/",
        "state_", STRUUID() ) ) AS ?propertyIRI )
    BIND( IRI( CONCAT( "https://host/db/architect/",
        "property_", STRUUID() ) ) AS ?propertyIRI )
    BIND(now() AS ?now)
    BIND(<https://host/project/space_1> AS ?foi)
    BIND(<https://niras.dk/employees/mhra> AS ?userIRI)
    BIND("Just a test" AS ?comment)
    BIND("20 m2"^^cdt:area AS ?val)
    # foi must exist
    ?foi ?p ?o .
    # the foi cannot have the property assigned already
    MINUS { ?foi props:area ?prop . }
}
```

Update. The `putProp()` method for updating a property by assigning a new state is comparable to `postProp()`, but separate methods exist for setting the reliability or restoring a specific property (as described in (Rasmussen et al., 2018b)). Restoring a property (method `restoreProp()`) or setting the reliability (method `setReliability()`) requires the IRI of a specific property (`propertyIRI`) as argument. The `putProp()` method, however, accepts also a set of the IRI of a specific FoI (`foiIRI`) and a `propertyType` or a `path`.

The query generated by the `putProp()` method is comparable with the query in Listing 8. It contains `MINUS` clauses so that it will only create a new property if the previous state is not an instance of opm:Confirmed, opm:Derived or opm:Deleted. The first because a confirmed property should not be changed, the second because a derived property should be changed by the algorithm which it was generated by and the latter because a deleted property should first be restored using the `restoreProp()` method which restores the previous state. If the existing value is equal to the new one, it will also not be updated.

Delete. Creating a query to delete a property is achieved by using the `setReliability()` method to set the reliability to deleted. A helper method, `deleteProperty()` also exists. This method only takes the `propertyIRI` as an argument and preferably a `userIRI` and a `comment`.

5.3. Calculations

The OPMCalc interface provides methods for dealing with OPM calculations. Similar to OPMProp, it contains methods to generate queries for CRUD operations on the AEC-KG. When dealing with calculations, however, there are both the management of opm:Calculation instances and the operations to be performed on the AEC-KG in order to infer derived properties.

Create. The `postCalcData()` method returns a query for creating a new opm:Calculation instance. As a minimum, it takes a `label`, an `expression`, a set of `argumentPaths` and the type of the inferred property (`inferredProperty`) as arguments. The number of variables used in the `expression` is compared to the number of `argumentPaths` to ensure that the two match and it is checked that the variable names match. For the parametric queries to function, the name used for the first variable in each argument path is replaced by `?foi`. This means that `?s props:area ?a` is automatically changed to `?foi props:area ?a`. Optional arguments include `userIRI`, a FoI restriction (opm:foiRestriction) which will restrict the calculation to only be applied to a specific FoI and a path restriction (opm:pathRestriction) which will restrict the calculation only to be applied where a specific path is matched. The generated query creates a resource with the above properties assigned (similar to Listing 2).

The `postCalc()` method takes all the arguments that are available on an opm:Calculation instance including the `calculationIRI`, and performs the same validation of the arguments and generates a query like the one illustrated in Listing 5. As described in Section 3.3, lines 18-29 and 31-42 retrieve the arguments. OPM-QG generates these according to the number of arguments given in the calculation. Each argument path is first appended with an underscore suffix for the argument variable name. This is because the variable name is instead used to describe the value of the most recent property state. The state is itself saved in a variable, and all these are appended in line 10 where it is specified what property states the specific derived property state was derived from.

If the expression contains an aggregation function, the structure of the calculation is quite different. OPM-QG will recognise either of these and instead generate a query like the one shown in Listing 9. For aggregation functions, it is further checked that the list of opm:argumentPaths only contains one item.

The query in Listing 9 is a `CONSTRUCT` query, so it will generate all the new derived properties and return the full graph without materialising it in the AEC-KG. This enables the end user to evaluate the results before making a final change and thereby provides insights for the engineers to compare and assess changes continuously without having everything being updated automatically.

The sub-query in lines 19-25 assigns the FoI to variable `?foi` and all the latest states of the properties that match the path to variable `?state1`. This sub-query is needed in order to get the individual states for assigning the prov:wasDerivedFrom predicate to the derived property (line 10). The actual sum is calculated in the next sub-query. This query generates a result for each `?foi` and creates IRIs for the new derived properties while doing so. At line 55 the expression is applied. For this particular query, it would be enough to just assign the value of `?htr` directly, but this approach allows for post-processing such as formatting the result or adding 10 % (`sum(?htr)*1.1`). OPM-QG takes care of removing the `sum()` function from the expression since `?htr` already holds the sum. The property will not be assigned to FoIs already having the property assigned.

Listing 9: SUM query returned by `postCalc()`.

```
1  CONSTRUCT {
2      ?foi ?inferredProperty ?propertyIRI .
3      ?propertyIRI a opm:Property ;
4          opm:hasPropertyState ?stateIRI .
5      ?stateIRI a opm:CurrentPropertyState ,
6                          opm:Derived , ?reliability ;
7          schema:value ?res ;
8          prov:generatedAtTime ?now ;
9          prov:wasAttributedTo <https://host/project/c2> ;
10         prov:wasDerivedFrom ?state1 .
11 }
12 USING NAMED <https://host/project>
13 USING NAMED <https://host/project-I>
14 WHERE {
15     BIND(props:transmissionHeatTransferRate AS
16                                 ?inferredProperty)
17     # GET THE MOST RECENT STATES OF THE ARGUMENTS
18     GRAPH ?g {
19         { SELECT ?foi (?state AS ?state1) WHERE {
20             ?foi a ice:ThermalEnvironment ;
21                 ^ice:surfaceInterior ?i .
22             ?i props:totalHeatTransferRate ?htr_ .
23             ?htr_ opm:hasPropertyState ?state .
24             ?state a opm:CurrentPropertyState
25         }}
26         # CALCULATE THE SUM
27         { SELECT ?foi (SUM(?res_) AS ?htr)
28             (IRI(CONCAT("https://host/project/",
29                 "state_", STRUUID())) AS ?stateIRI)
30             (IRI(CONCAT("https://host/project/",
31               "property_", STRUUID())) AS ?propertyIRI)
32             (now() AS ?now)
33           WHERE {
34             ?foi a ice:ThermalEnvironment ;
35                 ^ice:surfaceInterior ?i .
36             ?i props:totalHeatTransferRate ?htr_ .
37             ?htr_ opm:hasPropertyState ?state1 .
38             ?state1 schema:value ?htr__ .
39             BIND(?htr__ AS ?res_)
40           } GROUP BY ?foi
41         }
42         # INHERIT CLASS OPM:ASSUMED OR OPM:DELETED
43         OPTIONAL {
44             ?state1 a ?reliability .
45             FILTER( ?reliability = opm:Assumed ||
46                     ?reliability = opm:Deleted )
47         }
48     }
49     # DO NOT APPEND IF PROPERTY ALREADY DEFINED
50     MINUS {
51         GRAPH <https://host/project-I> {
52             ?foi ?inferredProperty ?prop }
53     }
54     # APPLY EXPRESSION
55     BIND((?htr) AS ?res)
56 }
```

Read. Getting calculation data is achieved with the `getCalcData()` method. If no arguments are provided, it will return a query to get all calculations. Providing a `calculationIRI` will return the data for that specific calculation. Providing a `label` will return the calculation matching that particular label. Providing a `foiIRI` and a `propertyType` will return data on the calculation which inferred the particular derived property.

`getOutdated()` is a method for retrieving all derived properties where one or more of the arguments

is no longer the opm:CurrentPropertyState. It can be restricted to only return derived properties of a specific FoI.

`getSubscribers()` returns a list of derived properties that are dependent on a specific property. This can be used to evaluate whether other parties will be influenced before a change to the property is conducted. Instead of providing an IRI for the property, it is also possible to provide a `foiIRI` and the `propertyType`.

Getting the latest state or all the states of a particular derived property is not different from getting a typed property. The OPMProp interface contains methods for this purpose.

Update. Calculations do not use OPM for managing the specifications, and hence the calculations themselves cannot be updated. It is, however possible to support this by simply assigning property states to the expression, argument paths and so forth (`putCalc()` method). This will, however, increase complexity, since a derived property will not only be outdated when one of its arguments has changed, but also if the calculation which it was attributed to has changed.

Similar to the `postCalc()` method, the `putCalc()` method takes all the arguments that are available on an opm:Calculation instance, including the `calculationIRI`, and generates an update query. The only difference is that this query will apply new states to existing derived properties where at least one argument has changed.

Delete. A derived property automatically inherits the opm:Deleted state from any of its arguments and thereby automatically becomes deleted itself.

6. Proof of Concept

In order to demonstrate the capabilities of the OPM architecture, a Proof of Concept (PoC) application was developed. The application performs the particular design task of calculating heating demand as described in the introduction. It uses a generic approach; however, that is transferable to other design tasks in the future.

6.1. System Architecture

Since the AEC-KG was stored in a triplestore that exposes a SPARQL endpoint (Ogbuji, 2013), it would have been possible for a client application to perform queries directly through HyperText Transfer Protocol (HTTP) requests. However, it was decided to make a middleware on a backend server that handles communication with the server. As a result, the frontend application can be developed by developers with no knowledge of RDF and OPM, and they are hence protected from the complex queries demonstrated in the previous sections. This is particularly practical for tasks that require several queries to the triplestore and further entails that complicated requests can be used across several client applications.

The backend is built as a Representational state transfer (REST) API which exposes a set of routes to which client applications can send HTTP request in order to do CRUD operations on the AEC-KG (see Section 5). Some of the routes are generic and will, for example, return all properties assigned to a FoI, change a class assigned to a FoI, update a property or return a full list of calculations. Others are provided for a particular application and will, for example, return all the parts of the building envelope that face a specific room.

The frontend is built in a JavaScript framework called Angular[9] and it is structured so that a service component takes care of the communication with the backend whereas a set of controllers take care of building the view based on the data returned by the service.

In the next sections, some of the core functionality of the backend is described, but only at an overall level. The frontend is not described and it is generally not the intention to describe the architecture in detail. The purpose is to communicate the main design considerations in relation to OPM.

6.2. Interface to the Architect

Typical practice in engineering companies is that new drawings a BIM model are received from the architect on a weekly basis. The architectural model is then mainly used as an underlay for the mechanical and structural design models. With this work, we suggest an approach where all the valuable information is extracted for further use, thereby relying on the set of ontologies and methods proposed within the W3C LBD Community Group. For example, using the Revit-BOT-exporter [10] with an extension for extracting the building envelope and for communicating with the REST API instead of

[9] `https://angular.io/`

[10] `https://github.com/MadsHolten/revit-bot-exporter`

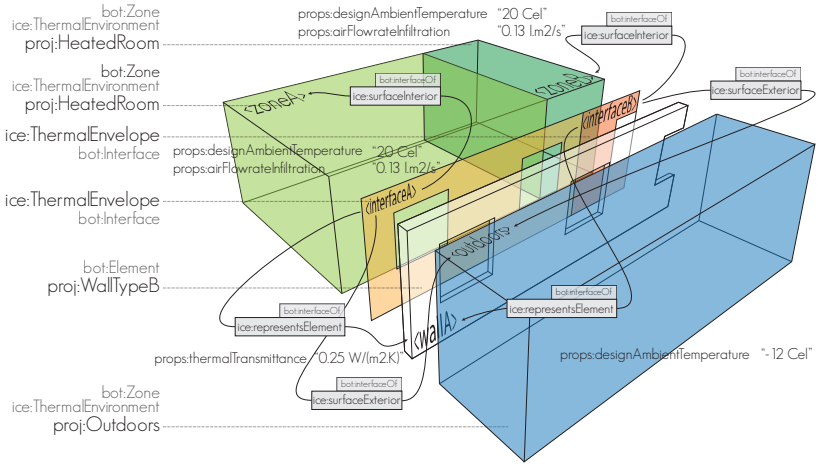


Figure 9: Data model for heat loss calculation.

writing triples to a file, it was possible to establish direct communication between the native BIM tool and the backend. All topological relationships and properties are extracted and sent via an OPM batch upload route to the server through this setup.

In this OPM batch upload route, all the zone and element instances and their topological relationships are written to the AEC-KG, as soon as data from the BIM software is received. These instances and relationships are not managed with OPM. Only properties such as space names and numbers, areas, volumes, 2D space boundaries and object mesh models are managed with OPM. All properties are received in complexity level L1 (see Section 2.2), and these are loaded into a temporary named graph in the AEC-KG (see Section 4.2). Then, a comparison is conducted between what is already in the AEC-KG and what new properties have been received. Finding the new properties is handled by searching the temporary graph for matches to the triple path `?foi ?prop ?val` while leaving out results in the project graph meeting the triple pattern `?foi ?prop ?x`. New properties and states are created using the API discussed in Section 5, and the triples are added to the project graph. Next, the backend checks for updated triples by finding the value of the latest property state and comparing it to the new value. Only if states are different, they are added to the project graph in the form of new property states for specific updated properties.

6.3. Appending Engineering Properties

With the initial graph in place, starting from the architectural design model, the engineer then defines and assigns a set of project-specific classes for element and zone types with OWL property restrictions similar to what was shown in Listing 6. For heat loss calculations, most spaces are simply seen as heated or unheated, and the exterior can be either ground or air. The data model and the inherited properties are illustrated in Figure 9. It is based on BOT with Indoor Climate and Energy (ICE) specific extensions and even more specific project extensions. The two rooms are instances of the project-specific `proj:HeatedRoom` ($\sqsubseteq$ `ice:ThermalEnvironment` $\sqsubseteq$ bot:Zone) and inherit a props:designAmbientTemperature of 20 $°C$ as well as an props:airFlowRateInfiltration of 0.13 $l.m^2/s$. The outdoor environment is an instance of the project-specific `proj:Outdoors` class, thereby inheriting a props:designAmbientTemperature of -12 $°$ C. The wall is an instance of the project-specific `proj:WallTypeB` class, thereby inheriting a props:thermalTransmittance of 0.25 W/m^2K. Each `ice:ThermalEnvelope` ($\sqsubseteq$ bot:Element) has a props:heatTransferSurfaceArea assigned explicitly, and each individual room also has an area. Hence, all the necessary information for deriving the heating demand is available, and a set of opm:Calculations can be defined to do so.

Pre-defined calculations. Table 2 lists all the calculations that were defined in the project, including their opm:inferredProperty, opm:expression and opm:argumentPaths. A quick examination of the inferred properties and the argument paths reveals that there are some internal interdependencies. As long as there are no circular dependencies, it is not a problem, and therefore a check needs to be performed on the backend to assure that no circular dependencies will be inferred by any new calculation before it is created.

Table 2: Predefined opm:Calculations.

props:nominalUA	
`?u*?a`	'?foi props:heatTransferSurfaceArea ?a', '?foi ice:representsElement ?el . ?el props:thermalTransmittance ?u'
props:designTemperatureDifference	
`?te-?ti`	'?foi ice:surfaceInterior ?si . ?si props:designAmbientTemperature ?ti', '?foi ice:surfaceExterior ?se . ?se props:designAmbientTemperature ?te'
props:totalHeatTransferRate	
`?ua*?dt`	'?foi props:designTemperatureDifference ?dt', '?foi props:nominalUA ?u'
props:transmissionHeatTransferRate	
`sum(?htr)`	'?foi a ice:ThermalEnvironment ; ^ice:surfaceInterior ?int', '?int props:totalHeatTransferRate ?htr'
props:infiltrationHeatTransferRate	

`?a*?inf*`
`1.166*1.0075`
`*(?ti-(-12))`
	'?sp props:netFloorArea ?a',
	'?sp props:designAmbientTemperature ?ti',
	'?sp props:airFlowrateInfiltration ?inf'

props:heatingDemand	
`?tr+?inf`	'?foi props:transmissionHeatTransferRate ?tr', '?foi props:infiltrationHeatTransferRate ?inf'

Appending calculations. When sending a `POST` request to the IRI of a calculation, new derived properties are appended. The `getCalcData()` method of the OPM-QG is used to get the calculation data and subsequently, the `postCalc()` method is used to infer the derived properties where the arguments are matched but the derived property is not already appended.

Each time a new BIM model is received from the architect, there are potentially new matches to the calculations, and, therefore, the newly derived properties must be appended. Because of the interdependencies between derived properties, it might require several loops for all the derived properties to be inferred, so some pre-processing is done. The whole network of interdependencies is explicitly stated in the graph, so it is possible first to calculate an execution order. As a result, the server load is reduced dramatically.

A dedicated route on the backend calculates the full tree of all calculations using the algorithm illustrated in the BPMN diagram in Figure 10. The depth of a calculation denotes its position in the dependency chain. A calculation having depth 0 has no dependencies and can hence be executed directly. Table 3 shows the depths of all the properties inferred by the calculations in Table 2. The first three are independent on output from the other calculations, but three other properties must first be inferred by calculations in order to calculate the props:heatingDemand. All calculations located at the same depth can be executed in parallel.

Table 3: Calculation depths for calculations in Table 2.

Inferred property	depth
props:infiltrationHeatTransferRate	0
props:designTemperatureDifference	0
props:nominalUA	0
props:totalHeatTransferRate	1
props:transmissionHeatTransferRate	2
props:heatingDemand	3

Updating calculations. When sending a `PUT` request to the IRI of a calculation, existing derived properties are updated. The approach is similar to appending calculations, but instead uses the `putCalc()` method to update the derived properties where at least one of its arguments has changed.

The task of updating derived properties is not automated since the engineer must make the decision of conducting a change. The engineer is, however, provided with an overview of the outdated properties along with insights concerning the consequences of conducting a particular change. Thereby, the engineer is provided with supporting tools for decision making.

In the user interface, an outdated derived property is highlighted, and clicking an icon will request the backend for a full dependency tree of the particular outdated property. The algorithm is a bit different from what is illustrated in Figure 10 and it uses a recursive "follow your nose" approach for retrieving a full list of arguments. Figure 11 shows an example of a derived property

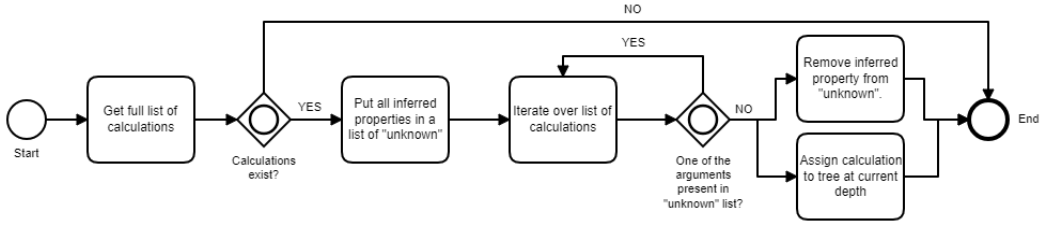


Figure 10: Algorithm to calculate complete execution tree for calculations.

with a rather long dependency chain inferred by the calculations shown in Table 2. The figure reveals that the props:heatingDemand is outdated because the props:heatTransferSurfaceArea of one of the thermal envelope segments facing the particular space is no longer valid. This consequently means that the props:nominalUA and hence also the props:totalHeatTransferRate of that particular segment are outdated, which in turn means that the props:transmissionHeatTransferRate of the space is also outdated.

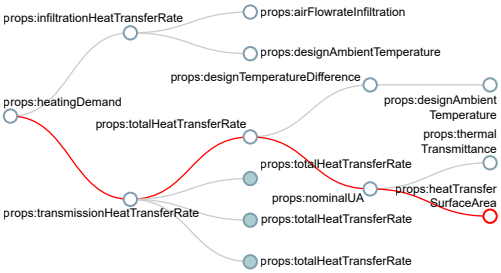


Figure 11: When the user clicks an outdated derived property, the full calculation tree is shown to provide insights.

Performing a `PUT` request on the derived props:heatingDemand will not have any influence as long as the derived props:transmissionHeatTransferRate has not been revised. Therefore, updating the props:heatingDemand will require that all the intermediate derived properties are also updated starting with the one closest to the typed property (props:heatTransferSurfaceArea). Since all these properties belong to the engineer's design, she can decide to update the whole tree, which can be achieved by sending a `PUT` request to the calculation IRI with a query parameter specifying that the whole chain should be updated. If one or more properties had belonged to another stakeholder, the decision of updating the whole chain would require involving this 3rd party.

Checking for outdated. It was previously described how an outdated property is identified by checking if any of the arguments (or arguments' arguments) are no longer an instance of opm:CurrentPropertyState. This can be achieved with a SPARQL property path and a `MINUS` clause: `?prop prov:wasDerivedFrom+ ?arg . MINUS{?arg a opm:CurrentPropertyState}`. This, however, causes problems when depending on OWL restrictions for property inheritance. If a wall instance is, for example, changed from `proj:WallHeavy` to `proj:WallLight`, the inherited U-value will change accordingly. The derived property for props:nominalUA is dependent on the U-value, but, since the latest state of this property is related to the most recent state of the U-value of `proj:WallHeavy` (by prov:wasDerivedFrom), the property path above will not recognise that the derived property is no longer valid. Re-calculating the result with a `PUT` request to the calculation IRI will retrieve the new result, and comparing this with the current result will reveal that the property is outdated. The comparison approach is more resource-intensive than the one that looks for states that are not classified as opm:CurrentPropertyState. This resource-intensive query can, however, be performed on the server as a scheduled job, thereby explicitly inferring the opm:OutdatedPropertyState class. Also, it is possible to do this check each time the class of an instance is changed.

6.4. Interface to Other Engineers

The props:heatingDemand of each space sets the boundary conditions for the heater serving that space. Generating heaters can be automated with a SPARQL update query which searches for any

space classified as a `HeatedSpace` with a heating demand above a certain threshold that does not already have a heater assigned. Calculations specific to the flow system can then be set up by the HVAC engineer similar to the procedure demonstrated for the ICE engineer in Section 6.3.

7. Conclusions and Future Work

7.1. Results

With this work, we showed a novel approach for working with interconnected data in a construction project using semantic web technologies. We did so by extending the initial work on OPM presented in Rasmussen et al. (2018b) with the concepts of reliability and calculations. Along with the new concepts, we presented an API for straightforward interaction with the OPM data. Furthermore, through a PoC implementation, it was demonstrated how this could be implemented. The implementation exposes the benefits of having all interconnections between project properties explicitly connected in an AEC-KG. By providing traceability and allowing for consequence analysis of a property change, the foundation for future software tools to better support the design engineers' decision making has been laid.

With the implementation, we demonstrated a system that answers the initial research question by storing design data in a structured way, allowing interrelated data to maintain their relations intact as the project progresses. The full history persists, and it is hence possible, at any point in time, to analyse the background of a specific design change. It was endeavoured to use terminology which is already widely adopted to describe the AEC-KG. Property inferencing is based on standard OWL reasoning and the widely adopted ontologies PROV-O and schema.org are used for describing properties.

The PoC shows an alternative approach for working with building data. It is a general experience in the industry that valuable data is trapped in proprietary BIM models and, with this work, we demonstrate how data from the other stakeholder's (architect's) model can be made accessible as RDF triples, directly from within the designer's toolchain and not through an intermediate file format.

7.2. Conclusions

Although a prototype of the system is currently being implemented in the Danish consulting engineering company Niras, it is still at an early stage, and evaluating how it performs on large projects is still unknown. There were, however, already several valuable discoveries. In the W3C LBD Community Group it has been discussed whether to model properties at level L1, L2 or L3. From our experiences, it is sufficient to exchange L1 with other practitioners, but internally in a company, there is much value to storing the full history at L3. With the implementation, data from the architect's model was exchanged at L1, but it was stored in the AEC-KG at L3. In other words, expressivity was added because it created value for the receiver. Even though BIM research tries to bring down the silos, it might still be desired for companies to retain part of the silo, and this can be achieved by exchanging L1 without all the metadata that is used internally in the individual companies.

The presented infrastructure is fundamentally different from how engineers currently work. The intention is that calculations and inferencing, which are currently done in different tools by different people, is described explicitly and in an interoperable manner. As soon as all the arguments for a particular calculation are available, so is the result. In such a setup, the primary task of an engineer is to make sure that all arguments are provided by the other practitioners. Further, since consequence analyses can be performed much faster, the engineer could potentially work with multiple parallel concepts for each design until sufficient knowledge is available for making a final choice.

Making calculations and hence design tasks reusable entails that the knowledge of the company as a whole grows over time in contrast to today where it is mainly the knowledge of the employees that evolves. Over time, the growing AEC-KG could potentially overcome the still existing challenge that companies lose access to large quantities of critical knowledge as employees turn over as it was implied by O'Leary (1998).

Expanding the functionality of the PoC is handled by extending the backend with new routes using the presented methodology. This demonstrates how the system can grow organically with the project and with the introduction of other design disciplines. Since all calculations and inferences are described in a formally structured and open manner, they can be copied to the next project, and this allows the company to grow the capabilities of the system over time.

Generating the calculation trees currently re-

quires some post-processing on the backend, because SPARQL does not return the depth when querying property paths. There do exist commercial extensions of SPARQL for applying general graph theory in queries[11], but no standard exists yet.

7.3. Future Outlook

With the implementation, the AEC-KG is distributed into named graphs each of which contains explicit and implicit data from several projects. Having more projects stored in the same database results in a significant increase in reasoning time for property inference. This is, however, highly dependent on the triplestore that is used, and more research and benchmarking in this field should be conducted. Alternatively, one could opt to store each project in a separate database.

In the PoC, all calculations were performed at the backend using an approach similar to the one illustrated in Figure 7. It would be interesting to investigate how some calculations could be performed by the client application while still following the OPM principles described in Section 3 when writing the resulting triples to the AEC-KG. It would also be interesting to investigate how more complex calculations such as thermal simulations could be implemented in practice.

Acknowledgements

Special thanks to the NIRAS ALECTIA Foundation and Innovation Fund Denmark for funding. Also thanks to Niras for allowing open distribution of the developed artifacts[12]. It is a fundamental necessity for the future growth of the proposed standards that they are adopted and further developed by the community.

References

Antoniou, G., Bikakis, A., 2007. DR-Prolog: A System for Defeasible Reasoning with Rules and Ontologies on the Semantic Web. IEEE Transactions on Knowledge and Data Engineering 19, 233–245. doi:10.1109/tkde.2007.29.

Barbieri, D.F., Braga, D., Ceri, S., Valle, E.D., Grossniklaus, M., 2010. Incremental Reasoning on Streams and Rich Background Knowledge, in: Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 1–15. doi:10.1007/978-3-642-13486-9_1.

Beetz, J., van Leeuwen, J., de Vries, B., 2008. IfcOWL: A case of transforming EXPRESS schemas into ontologies. Artificial Intelligence for Engineering Design, Analysis and Manufacturing 23, 89. doi:10.1017/s0890060409000122.

Berners-Lee, T., 2006. Linked data. W3C recommendation URL: https://www.w3.org/DesignIssues/LinkedData.html. accessed November 2018.

Bertelsen, S., 2003. Construction as a complex system, in: Proceedings of the 11th Annual Conference of the International Group for Lean Construction, pp. 143–168. URL: http://www.iglc.net/papers/details/231.

Bonduel, M., 2018. Towards a PROPS ontology. URL: https://github.com/w3c-lbd-cg/lbd/blob/gh-pages/presentations/props/presentation_LBDcall_20180312_final.pdf. accessed November 2018.

BSI, 2014. PAS 1192-3: 2014. Specification for information management for the operational phase of assets using Building Information Modelling. URL: https://shop.bsigroup.com/Sandpit/PAS-old-forms/PAS-1192-3/. accessed December 2018.

Carroll, J.J., Bizer, C., Hayes, P., Stickler, P., 2005. Named graphs, provenance and trust, in: Ellis, A., Hagino, T. (Eds.), Proceedings of the 14th international conference on World Wide Web, ACM Press. doi:10.1145/1060745.1060835.

Curry, E., O'Donnell, J., Corry, E., Hasan, S., Keane, M., O'Riain, S., 2013. Linking building data in the cloud: Integrating cross-domain building data using linked data. Advanced Engineering Informatics 27, 206–219. doi:10.1016/j.aei.2012.10.003.

Cyganiak, R., Wood, D., Lanthaler, M., 2014. RDF 1.1 concepts and abstract syntax. W3C recommendation URL: http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/. accessed November 2018.

Deshpande, A., Azhar, S., Amireddy, S., 2014. A framework for a BIM-based knowledge management system. Procedia Engineering 85, 113–122. doi:10.1016/j.proeng.2014.10.535.

Egbu, C., Hayles, C., Quintas, P., Hutchinson, V., Anumba, C., Ruikar, K., 2004. Knowledge Management for Sustainable Construction Competitiveness. Knowledge Management for Sustainable Construction Competitiveness Project, Partners in Innovation (CI 39/3/709) Work Package .

Mendes de Farias, T., Roxin, A., Nicolle, C., 2015. IfcWoD, semantically adapting IFC model relations into OWL properties, in: Proceedings of the 32nd CIB W78 Conference on Information Technology in Construction. URL: https://arxiv.org/abs/1511.03897. accessed March 2018.

Gallaher, M.P., O'Connor, A.C., John L. Dettbarn, J., Gilday, L.T., 2004. Cost Analysis of Inadequate Interoperability in the U.S. Capital Facilities Industry. Technical Report. National Institute of Standards and Technology. doi:10.6028/nist.gcr.04-867.

Harris, S., Seaborne, A., Prudhommeaux, E., 2013. SPARQL 1.1 query language. W3C recommendation 21. URL: https://www.w3.org/TR/sparql11-query/. accessed December 2018.

Hodgson, R., Keller, P.J., 2011. QUDT -quantities, units, dimensions and data types in OWL and XML. URL: http://www.qudt.org. accessed December 2018.

Holten Rasmussen, M., Pauwels, P., Lefrançois, M., 2018.

[11] https://www.stardog.com/blog/a-path-of-our-own/

[12] https://github.com/w3c-lbd-cg/opm

Building Topology Ontology. W3C Draft Community Group Report. W3C. URL: https://w3c-lbd-cg.github.io/bot/.

Isaac, S., Sadeghpour, F., Navon, R., 2013. Analyzing Building Information Using Graph Theory, in: Proceedings of the 30th International Symposium on Automation and Robotics in Construction and Mining (ISARC 2013): Building the Future in Automation and Robotics. doi:10.22260/isarc2013/0111.

Isikdag, U., Aouad, G., Underwood, J., Wu, S., 2007. Building information models: a review on storage and exchange mechanisms, in: Proceedings of the 24th CIB W78 Conference. URL: http://itc.scix.net/data/works/att/w78-2007-020-068b-Isikdag.pdf. accessed December 2018.

ISO10303-11, 2004. Industrial automation systems and integration – Product data representation and exchange – Part 11: Description methods: The EXPRESS language reference manual. Standard. International Organization for Standardization. Geneva, CH.

ISO16739, 2013. Industry Foundation Classes (IFC) for data sharing in the construction and facility management industries. Standard. International Organization for Standardization. Geneva, CH.

Kiviniemi, A., 2005. Requirements Management Interface to Building Product Models. Ph.D. thesis. Stanford, CA, USA. AAI3162341.

Lebo, T., Sahoo, S., McGuinness, D., Belhajjame, K., Cheney, J., Corsar, D., Garijo, D., Soiland-Reyes, S., Zednik, S., Zhao, J., 2013. PROV-O: The PROV ontology. W3C recommendation URL: https://www.w3.org/TR/prov-o/. accessed December 2018.

Lefrançois, M., 2017. Planned ETSI SAREF Extensions based on the W3C&OGC SOSA/SSN-compatible SEAS Ontology Patterns, in: Fensel, A., Daniele, L. (Eds.), Proceedings of Workshop on Semantic Interoperability and Standardization in the IoT, SIS-IoT,, CEUR-WS.org. URL: http://ceur-ws.org/Vol-2063/sisiot-paper2.pdf. accessed December 2018.

Lóscio, B.F., Burle, C., et al., N.C., 2012. Data on the Web Best Practices. W3C recommendation URL: https://www.w3.org/TR/2017/REC-dwbp-20170131/. accessed November 2018.

McKinsey, G.I., 2017. Reinventing construction: A route to higher productivity. URL: https://www.mckinsey.com/industries/capital-projects-and-infrastructure/our-insights/reinventing-construction-through-a-productivity-revolution. accessed November 2018.

Ogbuji, C., 2013. SPARQL 1.1 Protocol. W3C recommendation URL: http://www.w3.org/TR/2013/REC-sparql11-protocol-20130321/. accessed November 2018.

O'Leary, D., 1998. Enterprise knowledge management. Computer 31, 54–61. URL: https://doi.org/10.1109%2F2.660190, doi:10.1109/2.660190.

Pauwels, P., Deursen, D.V., Verstraeten, R., Roo, J.D., Meyer, R.D., de Walle, R.V., Campenhout, J.V., 2011. A semantic rule checking environment for building performance checking. Automation in Construction 20, 506–518. doi:10.1016/j.autcon.2010.11.017.

Pauwels, P., McGlinn, K., Törmä, S., Beetz, J., 2018. Linked data, in: Borrmann, A., Knig, M., Koch, C., Beetz, J. (Eds.), Building Information Modeling. Springer, pp. 181–197. doi:10.1007/978-3-319-92862-3.

Pauwels, P., Roxin, A., 2016. SimpleBIM : From full ifcOWL graphs to simplified building graphs, in: Christodoulou, S.E., Scherer, R. (Eds.), eWork and eBusiness in Architecture, Engineering and Construction (ECPPM 2016), CRC Press, Limassol, Cyprus. pp. 11–18. doi:10.1201/9781315386904.

Pauwels, P., Terkaj, W., 2016. EXPRESS to OWL for construction industry: Towards a recommendable and usable ifcOWL ontology. Automation in Construction 63, 100–133. doi:10.1016/j.autcon.2015.12.003.

Prud'hommeaux, E., Carothers, G., Beckett, D., Berners-Lee, T., 2014. RDF 1.1 Turtle - Terse RDF Triple Language. W3C recommendation URL: http://www.w3.org/TR/2014/REC-turtle-20140225/. accessed November 2018.

Rasmussen, M.H., Bonduel, M., Hviid, C.A., Karlshøj, J., 2018a. Managing Space Requirements of New Buildings Using Linked Building Data Technologies, in: eWork and eBusiness in Architecture, Engineering and Construction (ECPPM 2018), CRC Press. pp. 399–406.

Rasmussen, M.H., Lefrançois, M., Bonduel, M., Hviid, C.A., Karlshøj, J., 2018b. OPM: An ontology for describing properties that evolve over time, in: Poveda-Villalón, M., Pauwels, P., Roxin, A. (Eds.), Proceedings of the 6th Linked Data in Architecture and Construction Workshop, CEUR-WS.org. pp. 24–33. URL: http://ceur-ws.org/Vol-2159/03paper.pdf. accessed September 2018.

Santos, R., Costa, A.A., Grilo, A., 2017. Bibliometric analysis and review of Building Information Modelling literature published between 2005 and 2015. Automation in Construction 80, 118–136. doi:10.1016/j.autcon.2017.03.005.

Studer, R., Benjamins, V., Fensel, D., 1998. Knowledge engineering: Principles and methods. Data & Knowledge Engineering 25, 161–197. doi:10.1016/s0169-023x(97)00056-6.

Succar, B., 2009. Building information modelling framework: A research and delivery foundation for industry stakeholders. Automation in Construction 18, 357–375. doi:10.1016/j.autcon.2008.10.003.

Tserng, H.P., Lin, Y.C., 2004. Developing an activity-based knowledge management system for contractors. Automation in Construction 13, 781–802. doi:10.1016/j.autcon.2004.05.003.

W3C OWL Working Group, 2012. OWL 2 Web Ontology Language Document Overview (Second Edition). W3C recommendation URL: http://www.w3.org/TR/2012/REC-owl2-overview-20121211/. accessed November 2018.

Winch, G.M., 2010. Managing construction projects. John Wiley & Sons.

Zamanian, M., Pittman, J.H., 1999. A software industry perspective on AEC information models for distributed collaboration. Automation in Construction 8, 237–248. doi:10.1016/s0926-5805(98)00074-0.

Zhang, C., Beetz, J., de Vries, B., 2018. BimSPARQL: Domain-specific functional SPARQL extensions for querying RDF building data. Semantic Web 9, 829–855. doi:10.3233/sw-180297.

Appendix A

Software artefacts

Table A.1: Web-app modules

Name	Description
ng-plan[1]	Angular module for viewing 2D plan geometry of bot:Zones as an interactive Scalable Vector Graphics (SVG). Live example provided with the BOT-Duplex-house[2]
ng-mesh-viewer[3]	Angular module for viewing 3D mesh geometry of bot:Elements and bot:Zones in a web application. Live example provided with the BOT-Duplex-house[2]

[1] Module: https://www.npmjs.com/package/ng-plan
[2] Demo: https://madsholten.github.io/BOT-Duplex-house/
[3] Module: https://www.npmjs.com/package/ng-mesh-viewer

Table A.2: Web applications

Name	Description
Ng-Forge App[1,2]	An AutoDesk Forge based web-viewer that allows for querying based on Building Topology Ontology (BOT) topology. The tool is documented in paper FORGE (M. H. Rasmussen et al., 2017a).
SPARQL Protocol and RDF Query Language (SPARQL) visualizer[3,4]	Visualisation tool to communicate the content of an ontology and how it is used. Query results are visualised either in a table or in an node-link graph. The tool is documented in (Bonduel et al., 2018b).
DK Owner[5,6]	Part of a "Distributed Knowledge" concept. It is a simple Create, Read, Update and Delete (CRUD) application made for illustrating that a data model can be created without necessarily having any 3D geometry. The intention was to include it as part of paper REQ (M. H. Rasmussen et al., 2018a).

[1] Repository: https://github.com/MadsHolten/forge-sparql
[2] Demo: https://forge-sparql.herokuapp.com/
[3] Repository: https://github.com/MadsHolten/sparql-visualizer
[4] Demo: https://madsholten.github.io/sparql-visualizer/
[5] Repository: https://github.com/MadsHolten/dk-owner
[6] Demo: https://madsholten.github.io/dk-owner/

Table A.3: Tools

Name	Description
PySPARQL2D3[1]	Simple Python tool to make a node-link graph. The graph is generated using a Data driven documents (D3)-based implementation by Chawuthai and Takeda (2015).
revit-bot-exporter[2]	The exporter has been developed in collaboration with Jonas Eik Bacher-Jacobsen, Niras. The initial version is documented in M. H. Rasmussen et al., 2017a, but it has later been extended.
opm-qg[3]	Query Generator to generate querys to be performed on an Ontology for Property Management (OPM)-compliant dataset. The tool is documented in paper OPM2 (M. Rasmussen et al., 2019a).
props ontology generator	A GET request at https://objprops-gen.herokuapp.com/id/:wikiID[4] will return a dynamically generated ontology containing information about the specific property retrieved from Wikidata[5]. The approach is inspired by the product ontology[6] by Martin Hepp.

[1] Repository: https://github.com/MadsHolten/PySPARQL2D3
[2] Repository: https://github.com/MadsHolten/revit-bot-exporter
[3] Library: https://www.npmjs.com/package/opm-qg
[4] Example: https://objprops-gen.herokuapp.com/id/area
[5] Wikidata: https://www.wikidata.org/
[6] Web-app: http://www.productontology.org/

Acronyms

1D	One-Dimensional
2D	Two-Dimensional
3D	Three-Dimensional
AAU	Aalborg University Denmark
AEC	Architecture, Engineering and Construction
AEC-KG	AEC Knowledge Graph
AHU	Air Handling Unit
AI	Artificial Intelligence
API	Application Programming Interface
BACS	Building Automation and Control System
BCF	BIM Collaboration Format
BDS	Building Description System
BI	Business Intelligence
BIM	Building Information Modelling
BIMDO	BIM Design Ontology
BIMSO	BIM Shared Ontology
BMS	Building Management System
BOT	Building Topology Ontology
BPS	Building Performance Simulation
bSDD	buildingSMART Data Dictionary
bSI	buildingSMART International
CAD	Computer Aided Design
CCS	Cuneco Classification System
CDE	Common Data Environment

CDT	Custom DataTypes
CEDR	Conference of European Directors of Roads
CM	Construction Management
COMBINE	COmputer Models for the Building INdustry in Europe
CRUD	Create, Read, Update and Delete
CS	Computer Science
CSS	Cascading Style Sheets
CWA	Closed World Assumption
D3	Data driven documents
DAML	DARPA Agent Markup Language
DARPA	Defense Advanced Research Projects Agency
DB	Design-Bid
DBB	Design-Bid-Build
DBIA	Design-Build Institute of America
DBMS	Database Management System
DDS	Data Design System
DeSemWeb	Decentralized Semantic Web
DHW	Domestic Hot Water
DIKW	Data, Information, Knowledge, Wisdom
DL	Description Logic
DS	Dansih Standard
DTU	Technical University of Denmark
EAV	Entity-Attribute Value
ECPPM	European Conference on Product and Process Modelling
EEPSA	Energy Efficiency Prediction Semantic Assistant
ES	Expert System
ETH	Eidgenössische Technische Hochschule Zürich
EU	European Union
FEM	Finite Element Method
FM	Facility Management
FOAF	Friend Of A Friend ontology
FSO	Flow System Ontology
GARM	General AEC Reference Model
gbXML	green building XML
GIS	Geographic Information System
GML	Geography Markup Language
HTML	HyperText Markup Language

HTTP	HyperText Transfer Protocol
HVAC	Heating, Ventilation and Air Conditioning
IA	Intelligent Agent
IaaS	Infrastructure as a Service
IAI	International Alliance for Interoperability
ICE	Indoor Climate and Energy
ICT	Information and Communications Technology
IDM	Information Delivery Manual
IETF	Internet Engineering Task Force
IFC	Industry Foundation Classes
ifcOWL	IFC OWL
ifcWoD	IFC Web of Data
IMSvr	IFC Model Server
IPD	Integrated Project Delivery
IPI	Integrated Project Insurance
IRI	International Resource Identifier
ISO	International Organization for Standardization
ISQ	International System of Quantities
IT	Information Technology
JS	JavaScript
JSON	JavaScript Object Notation
JSON-LD	JSON Linked Data
KBS	Knowledge-Based System
KG	Knowledge Graph
KM	Knowledge Management
LBD	Linked Building Data
LCA	life Cycle Assessment
LDAC	Linked Data in Architecture and Construction
LDWG	Linked Data Working Group
LHS	Left Hand Side
LOD	Linked Open Data
LoD	Level of Detail
LPG	Labeled Property Graph
MEP	Mechanical, Electrical and Plumbing
MSc	Master of Science
MVD	Model View Definition
NASA	National Aeronautics and Space Administration
NIST	National Institute of Standards and Technology

NLP	Natural Language Processing
NoSQL	Not Only SQL
NPM	Node.js Package Manager
NRA	National Road Authorities
OGC	Open Geospatial Consortium
OIL	Ontology Inference Layer
OPM	Ontology for Property Management
OSH	Open Smart Home
OTL	Object Type Library
OWA	Open World Assumption
OWL	Web Ontology Language
PA	Project Alliancing
PAS	Publicly Available Specification
PDF	Portable Document Format
PG	Property Graph
PI	Piping & Instrumentation
PP	Project Partnering
PROV-O	Provenance Ontology
PRQ	Primary Research Question
QUDT	Quantities, Units, Dimensions, and Data Types
RDF	Resource Description Framework
RDFa	RDF in Attributes
RDFS	RDF Schema
REST	Representational State Transfer
RHS	Right Hand Side
RIF	Rule Interchange Format
RPDA	Relational Project Delivery Arrangement
RQ	Research Question
RT	Research Task
SaaS	Platform as a Service
SaaS	Software as a Service
SABLE	Simple Access to the Building Lifecycle Exchange
SAREF	Smart Appliances REFerence
SC	Subcommittee
SEAS	Smart Energy-Aware Systems
SHACL	Shapes Constraint Language
SKOS	Simple Knowledge Organization System

SOSA	Sensor, Observation, Sample, and Actuator Ontology
SPARQL	SPARQL Protocol and RDF Query Language
SQL	Structured Query Language
SSN	Semantic Sensor Networks Ontology
STEP	STandard for the Exchange of Product model data
SVG	Scalable Vector Graphics
TC	Technical Committee
UCUM	Unified Code for Units of Measure
UGent	Ghent University
UI	User Interface
UK	United Kingdom
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
US	United States
UX	User Experience
W3C	World Wide Web Consortium
W3C LBD CG	W3C LBD Community Group
WebGL	Web Graphics Library
WKT	Well-Known Text
www	World Wide Web
XML	Extensible Markup Language
XSD	XML Schema Definition

Glossary

Agent	In Computer Science (CS), an agent is a software that acts on the behalf of a user. For example a web or mobile application through which a user communicates with some resource or an autonomous agent - a so-called Intelligent Agent (IA). 14, 31, 32, 51, 87
Angular	TypeScript based framework for building single page web applications. It is open-source and the development is led by Google. Available at https://angular.io/ 69, 85, 87, 201
ArchiCAD	Building Information Modelling (BIM) authoring tool primarily scoped for architects and landscape. Available at https://www.graphisoft.com/archicad/. 19
Assertion	Assertional knowledge deals with concrete objects of the considered domain rather than general notions. There exists a certain analogy between assertional information and database content. There are, however some significant differences. For example that databases typically operate with a Closed World Assumption (CWA) where the semantic web follows an Open World Assumption (OWA) (Hitzler et al., 2012). The assertion layer of a knowledge graph is sometimes referred to as the ABox. x, 35, 40–43, 51, 53, 57, 80, 91, 93

AutoDesk — Software vendor iii, xi, xviii, 18, 21, 28, 68, 71, 72, 89, 113, 202, 212–216

Axiom — The basic statements that an ontology expresses 39

BIM360 — Construction Management (CM) platform by AutoDesk. Available at https://bim360.autodesk.com/. 28

BIM9 — Commercial BIM cloud solution. 28

BIMcloud — Commercial BIM cloud solution. Available at https://www.graphisoft.com/bimcloud/. 28

BIMServer — Open source BIM cloud solution. Available at http://bimserver.org/. 28, 29, 43

BIMx — Commercial BIM cloud solution. 28

BuildingSMART — Standardisation organisation for BIM started as International Alliance for Interoperability (IAI) in 1996. The purpose of the organisation is to facilitate interoperability in the Architecture, Engineering and Construction (AEC) industry by enabling full information exchange between the various software programs used. The Industry Foundation Classes (IFC) exchange format and BIM Collaboration Format (BCF) are examples of developments that are maintained by buildingSMART. 18, 20, 26, 71, 83, 88, 92–94, 205, 220

Cadd Force — Commercial BIM cloud solution. Available at http://caddforce.com/ 28

Datacubist SimpleBIM — BIM tool for model validation. Available at http://www.datacubist.com/. 21, 25, 45

DDS-CAD — Data Design System (DDS) Computer Aided Design (CAD) is a BIM authoring tool for Mechanical, Electrical and Plumbing (MEP) design. Available at https://www.dds-cad.net/. 19

Directed Graph — A directed graph is a graph where all the edges are unidirectional. If an edge describes a relationship, it is therefore a relationship going from one node/vertex to another (See Figure 2.9). x, xxx, 14, 34, 35

Dynamo — Visual programming tool for Revit. Available at http://dynamobim.org/. 86, 89

Entity	Entities of an ontology are elements used to refer to real-world objects 34
Forge	Platform by AutoDesk aimed at moving BIM to the cloud. The Forge viewer is a ThreeJS-based Web Graphics Library (WebGL) viewer to render a BIM model in the browser. Available at https://forge.autodesk.com/ xi, xviii, 71, 72, 89, 113, 202
Git	Git is a version control system created by Linus Torvalds for development of the Linux kernel. 213
GitHub	GitHub is a web-based hosting service for version control using Git 57, 85
Graph	A graph is a mathematical concept that consists of nodes/vertices that are connected by edges (See Figure 2.9). Graphs are used to represent networks of entities (e.g. roads, social networks and so forth). 36–38, 44, 70–72, 77, 83, 85, 88, 89, 202, 203, 212, 214
GraphDB	Commercial triplestore. Available at http://graphdb.ontotext.com/. 87
Grasshopper	Visual programming tool for Rhino. Available at https://www.grasshopper3d.com/. 86, 89
IDA ICE	Building Performance Simulation (BPS) tool. Available at https://www.equa.se/en/. 21
IES Virtual Environment	BPS tool. Available at https://www.iesve.com/. 21
Isothermally	If air is supplied isothermally to a space the temperature remains constant. Therefore, no energy exchange occurs. 64
JavaScript	Programming language that along with HyperText Transfer Protocol (HTTP), HyperText Markup Language (HTML) and Cascading Style Sheets (CSS) is a World Wide Web (www) standard. It is supported both by web browsers and desktop environments xvii, 69, 81, 176, 207, 215, 217

Knowledge Graph	A knowledge graph represents facts about the world in the structure of a directed graph. These facts are described explicitly by using classes and properties from ontologies. xi, xvi, 14, 44, 52, 64, 73, 81, 93, 151, 205, 207, 218
Metadata	Data that provides information about other data. 16, 37, 75, 215
Microsoft Excel	Spreadsheet with calculation and graphing capabilities. iii
Model	In a BIM context a model is often understood as a Three-Dimensional (3D) geometry model of the building, but a model does not have to be geometrical. A data model is is an abstract model that describes real world objects and their relationships to properties and to other real world objects. vi, x, xi, xvi, xviii, 4, 8, 11, 17–22, 24–28, 30, 34, 42–44, 46–48, 50, 52, 57, 60, 62, 68–71, 73, 78, 83–85, 89–91, 113, 142, 151, 207, 212, 215, 216
MongoDB	Not Only SQL (NoSQL) database that uses JavaScript Object Notation (JSON)-like documents with schemata. Available at url-https://www.mongodb.com/. 89
Navisworks	BIM tool by AutoDesk for CM and clash detection. Available at https://www.autodesk.dk/products/navisworks/. 21
Neo4J	Commercial graph database. Available at https://neo4j.com/. 88
Neptune	Commercial triplestore. Available at https://aws.amazon.com/neptune/. 87
Niras	Danish consulting engineering company with 2,200 employees in 51 offices in 27 countries at the time of writing. Niras operate in a wide area of engineering disciplines from processing plants and construction over energy, environment, and infrastructure to development aid and urban planning. This Industrial PhD was conducted in a partnership between Niras and Technical University of Denmark (DTU). i, 54, 78, 85, 88, 220

Node.js — Node.js is an open-source, cross-platform, runtime environment for creating server-side applications in JavaScript. Available at https://nodejs.org/. 70, 85, 208

OBJ — Wavefront OBJ is a geometry format for describing 3D mesh geometry. 68, 69

Ontology — An ontology is defined as *"a formal, explicit specification of a shared conceptualization"* (Studer et al., 1998). It introduces a vocabulary and terminology relevant to the domain and describes the intended meaning of the vocabulary. x, xi, xv–xviii, xxix, 31, 32, 38, 39, 41–47, 51, 54–58, 60, 62–64, 66–68, 74, 78, 80–84, 88, 89, 92–94, 96, 105, 122, 133, 142, 151, 202, 203, 205, 206, 208, 209, 214, 216, 217, 219

Onuma System — Commercial BIM cloud solution. Available at http://www.onuma-bim.com/. 28, 29

Payload — The intended message of the transmitted data excluding headers and metadata 27, 28

Python — Programming language. Available at https://www.python.org/. 203

React — JavaScript (JS)-based framework for building single page web applications. It is open-source and the development is led by Facebook. Available at https://reactjs.org/ 87

Reasoning — the act of deducing implicit facts from explicitly stated facts when taking reasoning logic (rules and models) into account. 31, 217

Resource — Web jargon for any entity. Includes Web pages, parts of a Web page, devices, people and more (Berners-Lee et al., 2001). x, xxix, 27–29, 31, 33–36, 38, 39, 42, 52, 68, 69, 74, 89, 93, 208, 217

Revit — BIM authoring tool by AutoDesk. Available at https://www.autodesk.eu/products/revit. iii, xi, xviii, xix, 19, 68, 69, 86, 89, 113, 212, 220

Rhino — Rhinoceros is typically abbreviated Rhino, or Rhino3D and is a commercial CAD application. Available at https://www.rhino3d.com/. 86, 89, 213

RIB iTwo	BIM tool for cost estimation and time scheduling. Available at https://www.rib-software.co.uk/. 21
Robot	Finite Element Method (FEM) software by AutoDesk. Available at https://www.autodesk.com/products/robot-structural-analysis/. 21
Semantic	In ontology engineering, semantics refer to the meaning of concepts (classes), properties, and relationships that formally represent real-world entities in a logical underpinning. vi, x, xvi, xix, 14, 25, 31, 32, 41, 42, 44, 47, 50–52, 54, 80, 83, 87, 89, 93, 151
Sigma Estimates	BIM tool for cost estimation and time scheduling. Available at https://sigmaestimates.com/. 21
Solibri Model Checker	BIM tool for clash detection and model validation. Available at https://www.solibri.com/. 21, 26
Speckle	Open source collaboration and communication platform that connects parametric geometry models through connections in the native BIM environments using web technologies. Available at https://speckle.works/ 89
Stardog	Commercial triplestore. Available at https://www.stardog.com/. 89
Strusoft FEM-Design	FEM software. Available at https://strusoft.com/products/fem-design. 21
Synchro	BIM tool for cost estimation and time scheduling. Available at https://www.synchroltd.com/. 21
Tekla Structures	BIM authoring tool for structural design. Available at https://www.tekla.com/products/tekla-structures. 19

Terminology	A terminology provides a vocabulary together with information on terms are interrelated. Terminological knowledge constitutes an essential part of anWeb Ontology Language (OWL) document. There exists a certain analogy between terminological information and database schemata. There are, however some significant differences. For example that databases typically operate with a CWA where the semantic web follows an OWA (Hitzler et al., 2012). The terminology layer of a knowledge graph is sometimes referred to as the TBox. 14, 25, 38, 42, 43, 46, 51–53, 55–57, 60, 63, 65, 68, 71, 74, 80, 81, 84, 93, 215
This	Just a test for *Structure of the Thesis*. xxx
ThreeJS	JS library for WebGL. Available at https://threejs.org/. 85, 213
Trimble Connect	Commercial BIM cloud solution. Available at https://connect.trimble.com/. 28
Triple	Any statement in an Resource Description Framework (RDF) graph is described as a triple. A triple consists of a subject, a predicate and an object. x, 34–37, 40–42, 52, 68, 70, 77, 217
Triplestore	Triplestores are used to store and query triples using SPARQL Protocol and RDF Query Language (SPARQL). These typically also offer some reasoning capability. 84, 85, 87, 89, 213, 214, 216
Turtle	Turtle (Terse RDF Triple Language) is an RDF serialization. It is the triple pattern syntax of SPARQL. 34, 36, 40, 41
TypeScript	TypeScript is a programming language which transpiles to JavaScript. One of the main features is that it adds optional static typing. It is open-source and is developed and maintained by Microsoft. 211
Vectorworks	General purpose BIM authoring tool. Available at https://www.vectorworks.net/en. 19
Vico Office	BIM tool for cost estimation and time scheduling. Available at https://gc.trimble.com/product-categories/bim-solutions. 21

Vocabulary	A vocabulary includes a set of terms describing a certain domain of interest, thereby fixing their meaning (Hitzler et al., 2012). 43, 215, 217
Well-Known Text	Well-Known Text (WKT) is a text markup language for representing vector geometry such as points, lines and polygons. 68, 209, 218
Wikidata	A collaboratively edited Knowledge Graph (KG) hosted by the Wikimedia Foundation. In January 2019 it contained 53,685,800 items. 92
Wikipedia	A collaboratively edited encyclopedia hosted by the Wikimedia Foundation. 92
World Wide Web Consortium	The World Wide Web Consortium (World Wide Web Consortium (W3C)) is an international community led by Web inventor and Director Tim Berners-Lee. Member organisations, a full-time staff, and the public work together through this organisation to develop Web standards. xvi, 32, 54, 92, 105, 209, 218, 219

Acknowledgements

Special thanks to the NIRAS ALECTIA Foundation and Innovation Fund Denmark for funding. Also thanks to Niras for allowing open distribution of the developed artifacts. It is a fundamental necessity for the future growth of the proposed standards that they are adopted and further developed by the community.

I would like very much to thank Assistant Professor Pieter Pauwels at Ghent University (UGent). Pieter has made great contributions to the research community, and this was the reason why I initially contacted him. Pieter has been a great mentor throughout the project and as early as the second semester of the research he hosted me at a research stay at UGent. Pieter also invited me into the World Wide Web Consortium (W3C) Linked Building Data (LBD) community group and the Linked Data in Architecture and Construction (LDAC) workshops. We have had a good time, have enjoyed each others company and have had a great benefit from being able to discuss thoughts and ideas on the topic. During the stay at UGent we had a short research stay together at Technion. In this regard, I would also like to thank Professor Rafael Sacks for being a great host. We had a rewarding week and great workshops with his research team.

I would also like to thank Associate Professor Maxime Lefrançois. First of all for hosting me at a short research stay at École des Mines de Saint-Étienne where we had a workshop in relation to the Smart Energy-Aware Systems (SEAS) ontologies. Throughout the research project, Maxime has provided competent feedback, mainly in regard to the development of the Ontology for Property Management (OPM), and is has been a great pleasure to collaborate.

Thanks to all the members of the W3C Community Group for all the valuable discussions we have had during the biweekly meetings. In particular, I would like to thank Mathias Bonduel, Georg Ferdinand Schneider and Jyrki Oraskari, who I have been working with on research projects.

Thanks to my supervisors from Technical University of Denmark (DTU): Associate Professor Jan Karlshøj and Associate Professor Christian Anker Hviid and my company supervisor: Senior Engineer Morten Vammen Vendelboe. The project was initiated by Christian, who was at that time also employed part-time by the hosting company and part-time at DTU. Initially, Christian had a hard time in convincing me to conduct this project but today I am happy that he managed to convince me. Jan, who also supervised me in my Master's thesis agreed to be the main supervisor on the project. With his history in buildingSMART he has provided valuable knowledge in the domain of Building Information Modelling (BIM) and standardisation. Morten has a strong knowledge in the processes around the planning of building services, and because his initial knowledge in the scope of BIM was very limited, it has been great to see his conservatism fade as he gained insights. I am convinced that Morten will continue to be a great sounding board as we continue to implement these promising technologies at Niras.

Thanks to all my Niras colleagues with whom I have had rewarding conversations at lunch, over the coffee machine and through projects. In particular I would like to thank Stig Brinck who has been the project owner internally in the company. Stig has been participating in supervisor meetings, and since he is engaged in the technological development of the company, it has been confirming to have his support. This support was materialised when he employed the company's first industrial master's student: Christian Aaskov Frausing, who has developed the frontend of the radiator sizing application. Christian also studied Architectural Engineering, but his master's is in Computer Science. It has been valuable to discuss the software architecture with Christian, and we are both looking forward to continue to develop new applications for the platform. In this regard, I would also like to thank Jonas Eik Bacher-Jacobsen who, with his knowledge in Revit's Application Programming Interface (API), has been a great help in developing the revit-bot-exporter.

Lastly, I would like to thank my girlfriend and friends and family for being a great support through the course of the project.

Bibliography

Ackoff, R. L. (1989). From data to wisdom. *Journal of Applied Systems Analysis*, *16*, 3–9.

Adida, B., Birbeck, M., McCarron, S., & Herman, I. (2015). RDFa Core 1.1 - Third Edition. *W3C recommendation*. Accessed November 2018. Retrieved from http://www.w3.org/TR/2015/REC-rdfa-core-20150317/

Alavi, M. (1984, June). An assessment of the prototyping approach to information systems development. *Communications of the ACM*, *27*(6), 556–563. doi:10.1145/358080.358095

Ashburner, M., Ball, C. A., Blake, J. A., Botstein, D., Butler, H., Cherry, J. M., ... Eppig, J. T. et al. (2000). Gene Ontology: tool for the unification of biology. *Nature genetics*, *25*(1), 25.

Augenbroe, G. (1994). An overview of the COMBINE project. In R. J. Scherer (Ed.), *Proceedings of the First European Conference on Product and Process Modeling in the Building Industry (ECPPM'94)*, October 5–7, 1994 (p. 547). Citeseer. Dresden, Germany.

Azhar, S. (2011, July). Building information modeling (BIM): trends, benefits, risks, and challenges for the AEC industry. *Leadership and Management in Engineering*, *11*(3), 241–252. doi:10.1061/(asce)lm.1943-5630.0000127

Balaji, B., Bhattacharya, A., Fierro, G., Gao, J., Gluck, J., Hong, D., ... Whitehouse, K. (2016). Brick: Towards a Unified Metadata Schema For Buildings. In *Proceedings of the 3rd ACM International Conference on Systems for Energy-Efficient Built Environments*, November 16–17, 2016 (pp. 41–50). BuildSys '16. Palo Alto, CA, USA: ACM. doi:10.1145/2993422.2993577

Beetz, J., van den Braak, W., Botter, R., Zlatanova, S., & de Laat, R. (2014, August). Interoperable data models for infrastructural artefacts: a novel

IFC extension method using RDF vocabularies exemplified with quay wall structures for harbors. In *eWork and eBusiness in Architecture, Engineering and Construction (ECPPM 2014)* (pp. 135–140). Vienna, Austria: CRC Press. doi:10.1201/b17396-26

Beetz, J., van Berlo, L., de Laat, R., & van den Helm, P. (2010). BIMserver.org– An open source IFC model server. In *Proceedings of the 27th CIB W78 Conference*, November 16–19, 2010. Cairo,Egypt.

Beetz, J., van Leeuwen, J., & de Vries, B. (2009). IfcOWL: A case of transforming EXPRESS schemas into ontologies. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, *23*(1), 89–101. doi:10.1017/S0890060409000122

Belleau, F., Nolin, M.-A., Tourigny, N., Rigault, P., & Morissette, J. (2008, October). Bio2RDF: Towards a mashup to build bioinformatics knowledge systems. *Journal of Biomedical Informatics*, *41*(5), 706–716. doi:10.1016/j.jbi.2008.03.004

Berners-Lee, T. (2006). Linked data. *W3C recommendation*. Accessed November 2018. Retrieved from https://www.w3.org/DesignIssues/LinkedData.html

Berners-Lee, T., Hendler, J., & Lassila, O. (2001, May). The semantic web. *Scientific American*, *284*(5), 34–43. doi:10.1038/scientificamerican0501-34

Bersin, J. (2004). *The blended learning book: best practices, proven methodologies, and lessons learned*. Jossey-Bass/Pfeiffer.

Bertelsen, S. (2003). Construction as a complex system. In *Proceedings of the 11th Annual Conference of the International Group for Lean Construction* (Vol. 11, pp. 143–168). Accessed December 2018. Virginia, USA. Retrieved from http://iglc.net/Papers/Details/231

Bew, M. & Richards, M. (2008). Bew-Richards BIM maturity model. In *Buildingsmart construct it autumn members meeting*. Brighton, UK.

Bhatt, D. (2000). EFQM excellence model and knowledge management implications. *Published by EFQM Organization*, *8*.

Bizer, C., Heath, T., & Berners-Lee, T. (2009, July). Linked data - the story so far. *International Journal on Semantic Web and Information Systems*, *5*(3), 1–22. doi:10.4018/jswis.2009081901

Björk, B.-C. (1989, March). Basic structure of a proposed building product model. *Computer-Aided Design*, *21*(2), 71–78. doi:10.1016/0010-4485(89)90141-3

Bonduel, M., Rasmussen, M. H., Pauwels, P., Vergauwen, M., & Klein, R. (2018a). A novel workflow to combine BIM and Linked Data for existing buildings. In *eWork and eBusiness in Architecture, Engineering and Construction (ECPPM 2018)*, September 12–15, 2018 (pp. 407–414). Copenhagen, Denmark: CRC Press.

Bonduel, M., Rasmussen, M. H., Pauwels, P., Vergauwen, M., & Klein, R. (2018b). SPARQL-visualizer: A Communication Tool for Collaborative

Ontology Engineering Processes. Rejected. doi:10.13140/RG.2.2.13687.93603

Bonino, D. & Corno, F. (2008). DogOnt - Ontology Modeling for Intelligent Domotic Environments. In *Lecture notes in computer science* (pp. 790–803). Springer Berlin Heidelberg. doi:10.1007/978-3-540-88564-1_51

Brickley, D. & Guha, R. V. (2014). RDF Schema 1.1. *W3C recommendation*. Accessed November 2018. Retrieved from http://www.w3.org/TR/2014/REC-rdf-schema-20140225/

Brickley, D., Guha, R. V., & McBride, B. (2004). RDF Vocabulary Description Language 1.0: RDF Schema. *W3C recommendation*. Accessed November 2018. Retrieved from https://www.w3.org/TR/2004/REC-rdf-schema-20040210/

Brown, G. (1990, January). The BRIS simulation program for thermal design of buildings and their services. *Energy and Buildings*, *14*(4), 385–400. doi:10.1016/0378-7788(90)90100-w

BSI. (2013). PAS 1192-2: Specification for Information Management for the Capital/Delivery Phase of Construction Projects Using Building Information Modelling. British Standards Institute Standards Limited London.

BSI Standards Limited. (2013). Specification for information management for the capital/delivery phase of construction projects using building information modelling. doi:10.3403/30259522u

buildingSMART. (2018). buidlingSMART History. Accessed November 2018. Retrieved from https://www.buildingsmart.org/about/about-buildingsmart/history/

Bus, N., Roxin, A., Picinbono, G., & Fahad, M. (2018). Towards French Smart Building Code: Compliance Checking Based on Semantic Rules. In M. Poveda-Villalón, P. Pauwels, & A. Roxin (Eds.), *Proceedings of the 6th Linked Data in Architecture and Construction Workshop*, June 19–21, 2018 (Vol. 2159, pp. 6–15). CEUR Workshop Proceedings. Accessed November 2018. UCL, London, UK: CEUR-WS.org. Retrieved from http://ceur-ws.org/Vol-2159/01paper.pdf

Carroll, J. J., Bizer, C., Hayes, P., & Stickler, P. (2005). Named graphs, provenance and trust. In A. Ellis & T. Hagino (Eds.), *Proceedings of the 14th international conference on World Wide Web*, May 10–14, 2005. Chiba, Japan: ACM Press. doi:10.1145/1060745.1060835

Casellas, N. (2011). *Legal ontology engineering*. Springer Netherlands. doi:10.1007/978-94-007-1497-7

Charpenay, V., Kabisch, S., Anicic, D., & Kosch, H. (2015). An ontology design pattern for IoT device tagging systems. In *Proceedings of the 5th International Conference on the Internet of Things (IoT 2015)*, October 26–28, 2015 (pp. 138–145). IEEE. Seoul, South Korea. doi:10.1109/iot.2015.7356558

Chawuthai, R. & Takeda, H. (2015). RDF Graph Visualization by Interpreting Linked Data as Knowledge. In *Joint international semantic technology conference* (pp. 23–39). Springer.

Chong, H.-Y., Wong, J. S., & Wang, X. (2014, August). An explanatory case study on cloud computing applications in the built environment. *Automation in Construction*, *44*, 152–162. doi:10.1016/j.autcon.2014.04.010

Collinge, B. & Connaughton, J. (2017, July). Mobilizing BIM in a Collaborative Project Environment. In *Proceedings of the 25th Annual Conference of the International Group for Lean Construction*, July 9–12, 2017. Heraklion, Crete, Greece: International Group for Lean Construction. doi:10.24928/2017/0008

Cyganiak, R., Wood, D., & Lanthaler, M. (2014). RDF 1.1 concepts and abstract syntax. *W3C recommendation*. Accessed November 2018. Retrieved from http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/

Dainty, A., Leiringer, R., Fernie, S., & Harty, C. (2017, March). BIM and the small construction firm: a critical perspective. *Building Research & Information*, *45*(6), 696–709. doi:10.1080/09613218.2017.1293940

Daniele, L., den Hartog, F., & Roes, J. (2015). Study on semantic assets for smart appliances interoperability: d-s4: final report. *European Commission, Brussels*.

Davis, K. A. & Songer, A. D. (2009). Resistance to IT change in the AEC industry: Are the stereotypes true? *Journal of Construction Engineering and Management*, *135*(12), 1324–1333. doi:10.1109/iemc.2002.1038444

DBIA. (2018, June). Design-build utilization: combined market study. Acccessed December 2018. Retrieved from https://dbia.org/wp-content/uploads/2018/06/Design-Build-Market-Research-FMI-2018.pdf

Deshpande, A., Azhar, S., & Amireddy, S. (2014). A Framework for a BIM-based Knowledge Management System. *Procedia Engineering*, *85*, 113–122. doi:10.1016/j.proeng.2014.10.535

Directive, C. (2014, April). 24/EU of the European Parliament and of the Council of 26 February 2014 on public procurement and repealing Directive 2004/18/EC. *Official Journal of the European Union, L*, *94*, 65. Accessed September 2018. Retrieved from https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=celex%5C%3A32014L0024

DS418. (2011). *DS 418–Calculation of Buildings' Heat Loss*. Danish Standard. Nordhavn,Denmark.

Eastman, C. [Charles]. (1975, March). The use of computers instead of drawings in building design. *AIA Journal*, *63*(3), 46–50.

Eastman, C. [Chuck], Teicholz, P., Sacks, R., & Liston, K. (2008, February). *BIM handbook*. John Wiley & Sons, Inc. doi:10.1002/9780470261309

Engelbart, D. C. (1962, October). *Augmenting Human Intellect: a Conceptual Framework*. Air Force Office of Scientific Research. Washington 25, DC: Defense Technical Information Center. doi:10.21236/ad0289565

Eppinger, S. D., Whitney, D. E., Smith, R. P., & Gebala, D. A. (1989). Organizing the tasks in complex design projects. In *Lecture notes in computer science* (pp. 229–252). Springer-Verlag. doi:10.1007/bfb0014281

Facebook. (2018). Graphql. Accessed December 2018. Retrieved from https://facebook.github.io/graphql/June2018/

Fielding, R. T. (2000). *Architectural styles and the design of network-based software architectures* (Doctoral dissertation).

Fuchs, S. & Scherer, R. J. (2017, March). Multimodels — instant nD-modeling using original data. *Automation in Construction*, *75*, 22–32. doi:10.1016/j.autcon.2016.11.013

Gallaher, M. P., O'Connor, A. C., John L. Dettbarn, J., & Gilday, L. T. (2004, August). *Cost Analysis of Inadequate Interoperability in the U.S. Capital Facilities Industry*. National Institute of Standards and Technology. National Institute of Standards and Technology. doi:10.6028/nist.gcr.04-867

Gandon, F. & Schreiber, G. (2015). RDF 1.1 XML Syntax. *W3C recommendation*. Accessed November 2018. Retrieved from http://www.w3.org/TR/2014/REC-rdf-syntax-grammar-20140225/

Gielingh, W. (1988). General AEC reference model (GARM). *ISO TC184/SC4*. Accessed December 2018. Retrieved from http://itc.scix.net/data/works/att/w78-1988-165.content.pdf

Girard, J. & Girard, J. (2015). Defining knowledge management: toward an applied compendium. *Online Journal of Applied Knowledge Management*, *3*(1), 1–20. Accessed December 2018. Retrieved from http://www.iiakm.org/ojakm/articles/2015/volume3_1/OJAKM_Volume3_1pp1-20.pdf

Harris, S. & Seaborne, A. (2013). SPARQL 1.1 Query Language. *W3C recommendation*. Accessed November 2018. Retrieved from http://www.w3.org/TR/2013/REC-sparql11-query-20130321/

Hawke, S., Beckett, D., & Broekstra, J. (2013). SPARQL Query Results XML Format (Second Edition). *W3C recommendation*. Accessed November 2018. Retrieved from http://www.w3.org/TR/2013/REC-rdf-sparql-XMLres-20130321/

Heath, T. & Bizer, C. (2011, February). Linked data: evolving the web into a global data space. *Synthesis Lectures on the Semantic Web: Theory and Technology*, *1*(1), 1–136. doi:10.2200/s00334ed1v01y201102wbe001

Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P. F., & Rudolph, S. (2012). OWL 2 Web Ontology Language Primer (Second Edition). *W3C recommendation*. Accessed November 2018. Retrieved from http://www.w3.org/TR/2012/REC-owl2-primer-20121211/

Hoekstra, R. (2009). Ontology Representation–Design Patterns and Ontologies That Make Sense. In *Proceedings of the 2009 Conference on Ontology Representation: Design Patterns and Ontologies That Make Sense* (pp. 1–236). Amsterdam, The Netherlands, The Netherlands: IOS Press.

Isikdag, U., Aouad, G., Underwood, J., & Wu, S. (2007). Building information models: a review on storage and exchange mechanisms. In *Proceedings*

of the 24th CIB W78 Conference, June 26–29, 2007. Accessed December 2018. Maribor, Slovenia. Retrieved from http://itc.scix.net/data/works/att/w78-2007-020-068b-Isikdag.pdf

ISO10303-11. (2004, November). *Industrial automation systems and integration – Product data representation and exchange – Part 11: Description methods: The EXPRESS language reference manual*. International Organization for Standardization. Geneva, CH.

ISO16739. (2013, April). *Industry Foundation Classes (IFC) for data sharing in the construction and facility management industries*. International Organization for Standardization. Geneva, CH.

ISO29481-2. (2012, December). *Building information models – Information delivery manual – Part 2: Interaction framework*. International Organization for Standardization. Geneva, CH.

ISO639-1. (2002, July). *Codes for the representation of names of languages – Part 1: Alpha-2 code*. International Organization for Standardization. Geneva, CH.

Jackson, P. (1998). *Introduction to expert systems* (3rd). Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

Kent, D. C. & Becerik-Gerber, B. (2010, August). Understanding construction industry experience and attitudes toward integrated project delivery. *Journal of Construction Engineering and Management*, *136*(8), 815–825. doi:10.1061/(asce)co.1943-7862.0000188

Kifer, M. & Boley, H. (2013). RIF Overview (Second Edition). *W3C Working Group Note*. Accessed November 2018. Retrieved from http://www.w3.org/TR/2013/NOTE-rif-overview-20130205/

Kiviniemi, A. (2005a). Integrated product models in life-cycle evaluation and management of the built environment. In *Proceeding of the international workshop on lifetime engineering of civil infrastructure*, November 9–11, 2005. Ube, Yamaguchi, Japan.

Kiviniemi, A. (2005b). *Requirements Management Interface to Building Product Models* (Doctoral dissertation, Stanford University, Stanford, CA, USA). AAI3162341.

Klyne, G. & Carroll, J. J. (2004). Resource Description Framework (RDF): Concepts and Abstract Syntax. Accessed November 2018. Retrieved from https://www.w3.org/TR/rdf-concepts/#section-anyone

Knublauch, H., Allemang, D., & Steyskal, S. (2017). SHACL Advanced Features. *W3C Working Group Note*. Accessed December 2018. Retrieved from https://www.w3.org/TR/2017/NOTE-shacl-af-20170608/

Knublauch, H. & Kontokostas, D. (2017). Shapes Constraint Language (SHACL). *W3C recommendation*. Accessed December 2018. Retrieved from https://www.w3.org/TR/2017/REC-shacl-20170720/

Konchar, M. & Sanvido, V. (1998). Comparison of U.S. Project Delivery Systems. *Journal of Construction Engineering and Management*, *124*(6), 435–444. doi:10.1061/(ASCE)0733-9364(1998)124:6(435)

Lahdenperä, P. (2012, January). Making sense of the multi-party contractual arrangements of project partnering, project alliancing and integrated project delivery. *Construction Management and Economics*, *30*(1), 57–79. doi:10.1080/01446193.2011.648947

Lassila, O. & Swick, R. R. (1999). Resource Description Framework (RDF) Model and Syntax Specification. *W3C recommendation*. Accessed November 2018. Retrieved from https://www.w3.org/TR/1999/REC-rdf-syntax-19990222/

Lefrançois, M. & Zimmermann, A. (2016). Supporting arbitrary custom datatypes in RDF and SPARQL. In *The semantic web. latest advances and new domains* (pp. 371–386). Springer International Publishing. doi:10.1007/978-3-319-34129-3_23

Lefrançois, M., Zimmermann, A., Siira, E., Ghariani, T., Girod-Genet, M., Santos, G., ... Järvinen, H. (2017). SEAS Ontology. Retrieved from https://ci.mines-stetienne.fr/seas/index.html

Leiner, B. M., Cerf, V. G., Clark, D. D., Kahn, R. E., Kleinrock, L., Lynch, D. C., ... Wolff, S. (2009, October). A brief history of the internet. *Computer Communication Review*, *39*(5), 22–31. doi:10.1145/1629607.1629613

Licklider, J. C. R. (1965). *Libraries of the future*. MIT Press.

Licklider, J. C. R. & Clark, W. E. (1962). On-line man-computer communication. In G. A. Barnard (Ed.), *Proceedings of the Joint Computer Conference*, May 1–3, 1962 (pp. 113–128). San Francisco, California, USA: ACM.

Liebich, T. & Wix, J. (1999). Highlights of the development process of industry foundation classes. In M. A. Lacasse & D. J. Vanier (Eds.), *Proceedings of the 1999 CIB W78 Conference*, May 31–June 3, 1999 (Vol. 18, pp. 2758–2775). Accessed December 2018. Vancouver, Canada. Retrieved from http://itc.scix.net/data/works/att/w78-1999-2758.content.pdf

Liu, H., Liu, Y.-S., Pauwels, P., Guo, H., & Gu, M. (2017, December). Enhanced explicit semantic analysis for product model retrieval in construction industry. *IEEE Transactions on Industrial Informatics*, *13*(6), 3361–3369. doi:10.1109/tii.2017.2708727

Liu, Y., van Nederveen, S., & Hertogh, M. (2017, May). Understanding effects of BIM on collaborative design and construction: an empirical study in china. *International Journal of Project Management*, *35*(4), 686–698. doi:10.1016/j.ijproman.2016.06.007

Lóscio, B. F., Burle, C., Calegari, N. et al. (2012). Data on the Web Best Practices. *W3C recommendation*. Accessed November 2018. Retrieved from https://www.w3.org/TR/2017/REC-dwbp-20170131/

Luiten, B., Böhms, M., Alsem, D., & O'Keeffe, A. (2018, October), In *Proceedings of the 6th International Symposium on Life-Cycle Civil Engineering*, October 28–31, 2018. Accessed January 2019. Ghent, Belgium. Retrieved from https://www.roadotl.eu/static/media/IALCCE_2018_paper_INTERLINK_final_def.pdf

Ma, L. & Sacks, R. (2016, July). A Cloud-Based BIM Platform for Information Collaboration. In *Proceedings of the 33rd International Symposium on Automation and Robotics in Construction (ISARC)*, July 18–21, 2016. Auburn, Alabama, USA: International Association for Automation and Robotics in Construction (IAARC). doi:10.22260/isarc2016/0070

Maile, T., O'Donnell, J., Bazjanac, V., & Rose, C. (2013). BIM–Geometry modelling guidelines for building energy performance simulation. In *Proceedings of the 13th International Building Performance Simulation Association*, August 25–28, 2013 (Vol. 13, pp. 3242–3249). Accessed December 2018. Chambéry, France. Retrieved from http://www.ibpsa.org/proceedings/BS2013/p_1510.pdf

Mazairac, W. & Beetz, J. (2013, October). BIMQL – an open query language for building information models. *Advanced Engineering Informatics*, *27*(4), 444–456. doi:10.1016/j.aei.2013.06.001

McGuinness, D. L., Van Harmelen, F. et al. (2004). OWL web ontology language overview. *W3C recommendation*. Accessed November 2018. Retrieved from http://www.w3.org/TR/2004/REC-owl-features-20040210/

Mcguinness, D., Fikes, R., Hendler, J., & Stein, L. (2002, September). DAML+OIL: an ontology language for the semantic web. *IEEE Intelligent Systems*, *17*(5), 72–80. doi:10.1109/mis.2002.1039835

McKinsey, G. I. (2017). Reinventing construction: a route to higher productivity. Accessed November 2018. McKinsey & Company. Retrieved from https://www.mckinsey.com/industries/capital-projects-and-infrastructure/our-insights/reinventing-construction-through-a-productivity-revolution

Mell, P. M. & Grance, T. (2011). *The NIST definition of cloud computing*. National Institute of Standards and Technology. doi:10.6028/nist.sp.800-145

Mendes de Farias, T., Roxin, A., & Nicolle, C. (2015). IfcWoD, semantically adapting IFC model relations into OWL properties. In *Proceedings of the 32nd CIB W78 Conference on Information Technology in Construction*, October 26–29, 2015. Accessed March 2018. Eindhoven, The Netherlands. Retrieved from https://arxiv.org/abs/1511.03897

Métral, C., Billen, R., Cutting-Decelle, A.-F., & Van Ruymbeke, M. (2010). Ontology-based approaches for improving the interoperability between 3D urban models. *Journal of Information Technology in Construction*, *15*(14), 169–184. Accessed December 2018. Retrieved from http://www.itcon.org/2010/14

Modoni, G. E., Sacco, M., & Terkaj, W. (2014, June). A survey of RDF store solutions. In S. Terzi, B. Katzy, & S. Cunninghman (Eds.), *2014 international conference on engineering, technology and innovation (ICE)*, June 23–25, 2014. Bergamo,Italy: IEEE. doi:10.1109/ice.2014.6871541

Niknam, M. & Karshenas, S. (2017, August). A shared ontology approach to semantic representation of BIM data. *Automation in Construction*, *80*, 22–36. doi:10.1016/j.autcon.2017.03.013

Noy, N. F., McGuinness, D. L. et al. (2001). Ontology development 101: a guide to creating your first ontology. Stanford knowledge systems laboratory technical report KSL-01-05.

O'Brien, W. J. (2000). Implementation Issues in Project Web Sites: A Practioner's Viewpoint. *Journal of Management in Engineering*, *16*(3), 34–39. doi:10.1061/(ASCE)0742-597X(2000)16:3(34)

Ogbuji, C. (2013). SPARQL 1.1 Protocol. *W3C recommendation*. Accessed November 2018. Retrieved from http://www.w3.org/TR/2013/REC-sparql11-protocol-20130321/

O'Leary, D. (1998, March). Enterprise knowledge management. *Computer*, *31*(3), 54–61. doi:10.1109/2.660190

Pauwels, P., McGlinn, K., Törmä, S., & Beetz, J. (2018a). Linked data. In A. Borrmann, M. König, C. Koch, & J. Beetz (Eds.), *Building Information Modeling* (pp. 181–197). Springer. doi:10.1007/978-3-319-92862-3

Pauwels, P., Poveda-Villalón, M., Sicilia, Á., & Euzenat, J. (2018b, September). Semantic technologies and interoperability in the built environment. *Semantic Web*, *9*(6), 731–734. doi:10.3233/sw-180321

Pauwels, P. & Roxin, A. (2016). SimpleBIM : From full ifcOWL graphs to simplified building graphs. In S. E. Christodoulou & R. Scherer (Eds.), *eWork and eBusiness in Architecture, Engineering and Construction (ECPPM 2016)*, September 7–9, 2016 (pp. 11–18). Limassol, Cyprus: CRC Press. doi:10.1201/9781315386904

Pauwels, P. & Terkaj, W. (2016). EXPRESS to OWL for construction industry: Towards a recommendable and usable ifcOWL ontology. *Automation in Construction*, *63*, 100–133. doi:10.1016/j.autcon.2015.12.003

Pauwels, P., Zhang, S., & Lee, Y.-C. (2017, January). Semantic web technologies in AEC industry: a literature overview. *Automation in Construction*, *73*, 145–165. doi:10.1016/j.autcon.2016.10.003

Prud'hommeaux, E., Carothers, G., Beckett, D., & Berners-Lee, T. (2014). RDF 1.1 Turtle - Terse RDF Triple Language. *W3C recommendation*. Accessed November 2018. Retrieved from http://www.w3.org/TR/2014/REC-turtle-20140225/

Qiu, H., Schneider, G. F., Kauppinen, T., Rudolph, S., & Steiger, S. (2018, July). Reasoning on human experiences of indoor environments using semantic web technologies. In *Proceedings of the 35th International Symposium on Automation and Robotics in Construction (ISARC)*, July 20–25, 2018. Berlin,Germany: International Association for Automation and Robotics in Construction (IAARC). doi:10.22260/isarc2018/0013

Rasmussen, M. H., Bonduel, M., Hviid, C. A., & Karlshøj, J. (2018a). Managing Space Requirements of New Buildings Using Linked Building Data Technologies. In *eWork and eBusiness in Architecture, Engineering and Construction (ECPPM 2018)*, September 12–15, 2018 (pp. 399–406). Copenhagen, Denmark: CRC Press.

Rasmussen, M. H., Frausing, C. A., Hviid, C. A., & Karlshøj, J. (2018b). Demo: Integrating Building Information Modeling and Sensor Observations using Semantic Web. In M. Lefrançois, R. García-Castro, A. Gyrard, & K. Taylor (Eds.), *Proceedings of the 9th International Semantic Sensor Networks Workshop, International Semantic Web Conference*, October 9, 2018 (Vol. 2213, pp. 48–55). CEUR Workshop Proceedings. Accessed September 2018. Monterey, CA, United States. Retrieved from http://ceur-ws.org/Vol-2213/paper4.pdf

Rasmussen, M. H., Hviid, C. A., & Karlshøj, J. (2017a). Web-based topology queries on a BIM model. In *5th Linked Data in Architecture and Construction Workshop*, November 13–15, 2017. Univeristy of Burgundy, Dijon, France. doi:10.13140/RG.2.2.22298.95685

Rasmussen, M. H., Lefrançois, M., Bonduel, M., Hviid, C. A., & Karlshøj, J. (2018c). OPM: an ontology for describing properties that evolve over time. In M. Poveda-Villalón, P. Pauwels, & A. Roxin (Eds.), *Proceedings of the 6th Linked Data in Architecture and Construction Workshop*, June 19–21, 2018 (Vol. 2159, pp. 24–33). CEUR Workshop Proceedings. Accessed September 2018. UCL, London, UK: CEUR-WS.org. Retrieved from http://ceur-ws.org/Vol-2159/03paper.pdf

Rasmussen, M. H., Pauwels, P., Hviid, C. A., & Karlshøj, J. (2017b). Proposing a central AEC ontology that allows for domain specific extensions. In F. Bosché, I. Brilakis, & R. Sacks (Eds.), *Proceedings of the Joint Conference on Computing in Construction*, July 4–12, 2017 (Vol. 1). Heraklion, Crete, Greece: Heriot-Watt University. doi:10.24928/jc3-2017/0153

Rasmussen, M. H., Pauwels, P., Lefrançois, M., Schneider, G. F., Hviid, C., & Karlshøj, J. (2017c). Recent changes in the Building Topology Ontology. In *5th Linked Data in Architecture and Construction Workshop*, November 13–15, 2017. Univeristy of Burgundy, Dijon, France. doi:10.13140/RG.2.2.32365.28647

Rasmussen, M., Lefrançois, M., Pauwels, P., Hviid, C., & Karlshøj, J. (2019a). Managing interrelated project information in AEC Knowledge Graphs. *Automation in construction*. Under review.

Rasmussen, M., Pauwels, P., Hviid, C., & Karlshøj, J. (2019b). The BOT ontology: standards within a decentralised web-based AEC industry. *Automation in construction*. Under review.

Rezgui, Y., Boddy, S., Wetherill, M., & Cooper, G. (2011, May). Past, present and future of information and knowledge sharing in the construction industry: towards semantic service-based e-construction? *Computer-Aided Design*, *43*(5), 502–515. doi:10.1016/j.cad.2009.06.005

Robertson, M., Sørensen, C., & Swan, J. (2001, December). Survival of the leanest: intensive knowledge work and groupware adaptation. *Information Technology & People*, *14*(4), 334–352. doi:10.1108/09593840110411149

Sacks, R., Kaner, I., Eastman, C. M., & Jeong, Y.-S. (2010, July). The Rosewood experiment — Building Information Modeling and interoperability

for architectural precast facades. *Automation in Construction*, *19*(4), 419–432. doi:10.1016/j.autcon.2009.11.012

Santos, R., Costa, A. A., & Grilo, A. (2017, August). Bibliometric analysis and review of Building Information Modelling literature published between 2005 and 2015. *Automation in Construction*, *80*, 118–136. doi:10.1016/j.autcon.2017.03.005

Schneider, G. F. (2017). Towards aligning domain ontologies with the Building Topology Ontology. In *5th Linked Data in Architecture and Construction Workshop*, November 13–15, 2017. Univeristy of Burgundy, Dijon, France. doi:10.13140/RG.2.2.21802.52169

Schneider, G. F., Rasmussen, M. H., Bonsma, P., Oraskari, J., & Pauwels, P. (2018). Linked Building Data for Modular Building Information Modelling of a Smart Home. In *eWork and eBusiness in Architecture, Engineering and Construction (ECPPM 2018)*, September 12–15, 2018 (pp. 407–414). Copenhagen, Denmark: CRC Press.

Schröder, M., Hees, J., Bernardi, A., Ewert, D., Klotz, P., & Stadtmüller, S. (2018). Simplified SPARQL REST API. In A. Gangemi (Ed.), *Proceedings of the 15th European Semantic Web Conference* (pp. 40–45). Heraklion, Crete, Greece: Springer International Publishing. doi:10.1007/978-3-319-98192-5_8

Seaborne, A. (2013). SPARQL 1.1 Query Results JSON Format. *W3C recommendation*. Accessed November 2018. Retrieved from http://www.w3.org/TR/2013/REC-sparql11-results-json-20130321/

Smith, P. (2014). BIM implementation – global strategies. *Procedia Engineering*, *85*, 482–492. doi:10.1016/j.proeng.2014.10.575

Sporny, M., Kellogg, G., Lanthaler, M., Longley, D., & Lindström, N. (2014). JSON-LD 1.0 - A JSON-based Serialization for Linked Data. *W3C recommendation*. Accessed November 2018. Retrieved from http://www.w3.org/TR/2014/REC-json-ld-20140116/

Studer, R., Benjamins, V., & Fensel, D. (1998, March). Knowledge engineering: principles and methods. *Data & Knowledge Engineering*, *25*(1-2), 161–197. doi:10.1016/s0169-023x(97)00056-6

Succar, B. (2009, May). Building information modelling framework: a research and delivery foundation for industry stakeholders. *Automation in Construction*, *18*(3), 357–375. doi:10.1016/j.autcon.2008.10.003

Sutherland, I. E. (1964). Sketch pad a man-machine graphical communication system. In *Proceedings of the SHARE design automation workshop (DAC '64)*, May 6–7, 1965. Cambridge, Massachusetts, USA: ACM Press. doi:10.1145/800265.810742

Taelman, R., Vander Sande, M., & Verborgh, R. (2018). Graphql-ld: linked data querying with graphql. In M. van Erp, M. Atre, V. Lopez, K. Srinivas, & C. Fortuna (Eds.), *Proceedings of the ISWC 2018 Posters & Demonstrations, Industry and Blue Sky Ideas Tracks*, October 8–12, 2018 (Vol. 2180). CEUR Workshop Proceedings. Accessed December 2018. Monterey,CA,USA:

CEUR-WS.org. Retrieved from http://ceur-ws.org/Vol-2180/paper-65.pdf

Terkaj, W. & Pauwels, P. (2017). A method to generate a modular ifcOWL ontology. In E. M. Sanfilippo, L. Daniele, & G. Colombo (Eds.), *Proceedings of the 8th International Workshop on Formal Ontologies meet Industry*, September 21, 2017 (Vol. 2050). CEUR Workshop Proceedings. Accessed December 2018. Bozen-Bolzano, Italy: CEUR-WS.org. Retrieved from http://ceur-ws.org/Vol-2050/FOMI_paper_3.pdf

Terkaj, W., Schneider, G. F., & Pauwels, P. (2017). Reusing domain ontologies in linked building data: the case of building automation and control. In E. M. Sanfilippo, L. Daniele, & G. Colombo (Eds.), *Proceedings of the 8th International Workshop on Formal Ontologies meet Industry*, September 21, 2017 (Vol. 2050). CEUR Workshop Proceedings. Accessed November 2018. Bozen-Bolzano, Italy: CEUR-WS.org. Retrieved from http://ceur-ws.org/Vol-2050/FOMI_paper_4.pdf

Törmä, S. (2013, September). Semantic Linking of Building Information Models. In *2013 IEEE seventh international conference on semantic computing*, September 16–18, 2013 (pp. 412–419). Irvine, CA, USA: IEEE. doi:10.1109/icsc.2013.80

Treldal, N., Parsianfar, H., & Karlshøj, J. (2016). Using BCF as a mediator for task management in building design. In *Proceedings of the international RILEM conference materials, systems and structures in civil engineering*, August 21–24, 2016 (pp. 48–59). Rilem publications. Kgs. Lyngby, Denmark.

Tserng, H. P. & Lin, Y.-C. (2004, November). Developing an activity-based knowledge management system for contractors. *Automation in Construction*, *13*(6), 781–802. doi:10.1016/j.autcon.2004.05.003

Turner, J. et al. (1990). AEC building systems model. *ISO TC*, *1184*.

Turner, M. J., Clough, R. W., Martin, H. C., & Topp, L. J. (1956, September). Stiffness and deflection analysis of complex structures. *Journal of the Aeronautical Sciences*, *23*(9), 805–823. doi:10.2514/8.3664

Uschold, M. & Gruninger, M. (1996, June). Ontologies: principles, methods and applications. *The Knowledge Engineering Review*, *11*(02), 93. doi:10.1017/s0269888900007797

van Berlo, L. & Krijnen, T. (2014). Using the BIM collaboration format in a Server Based Workflow. *Procedia Environmental Sciences*, *22*, 325–332. doi:10.1016/j.proenv.2014.11.031

van Nederveen, G. & Tolman, F. (1992, December). Modelling multiple views on buildings. *Automation in Construction*, *1*(3), 215–224. doi:10.1016/0926-5805(92)90014-b

Venkatesh, V. (2000, December). Determinants of Perceived Ease of Use: Integrating Control, Intrinsic Motivation, and Emotion into the Technology Acceptance Model. *Information Systems Research*, *11*(4), 342–365. doi:10.1287/isre.11.4.342.11872

Venugopal, M., Eastman, C. M., Sacks, R., & Teizer, J. (2012, April). Semantics of model views for information exchanges using the industry foundation class schema. *Advanced Engineering Informatics*, *26*(2), 411–428. doi:10.1016/j.aei.2012.01.005

Verborgh, R. (2018, December). Designing a Linked Data developer experience. Accessed January 2019. Retrieved from https://ruben.verborgh.org/blog/2018/12/28/designing-a-linked-data-developer-experience/

Verborgh, R., Sande, M. V., Hartig, O., Herwegen, J. V., Vocht, L. D., Meester, B. D., ... Colpaert, P. (2016). Triple Pattern Fragments: A Low-Cost Knowledge Graph Interface for the Web. *SSRN Electronic Journal*. doi:10.2139/ssrn.3199223

Vouk, M. A. (2008). Cloud Computing - Issues, Research and Implementations. *Journal of Computing and Information Technology*, *16*(4), 235. doi:10.2498/cit.1001391

W3C OWL Working Group. (2012). OWL 2 Web Ontology Language Document Overview (Second Edition). *W3C recommendation*. Accessed November 2018. Retrieved from http://www.w3.org/TR/2012/REC-owl2-overview-20121211/

Wong, A., Wong, F. K., & Nadeem, A. (2009). Comparative roles of major stakeholders for the implementation of BIM in various countries. In *Proceedings of the International Conference on Changing Roles: New Roles, New Challenges, Noordwijk Aan Zee, The Netherlands* (pp. 5–9). TU Delft, Delft, The Netherlands.

Zamanian, M. & Pittman, J. H. (1999, February). A software industry perspective on AEC information models for distributed collaboration. *Automation in Construction*, *8*(3), 237–248. doi:10.1016/s0926-5805(98)00074-0

Zhang, C., Beetz, J., & de Vries, B. (2018, September). BimSPARQL: domain-specific functional SPARQL extensions for querying RDF building data. *Semantic Web*, *9*(6), 829–855. doi:10.3233/sw-180297

The AEC industry is fragmented as several actors consume, process and further develop a common project material. This, in combination with continuous design iterations and a highly document-centric working manner, challenges the design process.

This thesis investigates the possibilities of creating a web-based infrastructure that supports the need for dynamic, interconnected information exchanges. Semantic web technologies are used to capture knowledge about a building, its spatial and physical elements and their internal relationships in an extendable, distributed data model. Thereby, the building information is described in an unambiguous, machine-readable manner where the processing of design information can to some extent be automated.
The main contributions of this work are two ontologies, BOT and OPM that respectively describe the main topological principles of a building and terminology for handling complex design properties with interdependencies and varying reliability. By connecting to legacy BIM authoring tools and IFC it is further demonstrated how to interact with the data model in ways that align with the workflows that lie in designing a building.

DTU Civil Engineering

Brovej, Building 118
2800 Kongens Lyngby
Tlf. 45251700

www.byg.dtu.dk

87-7877-518-3