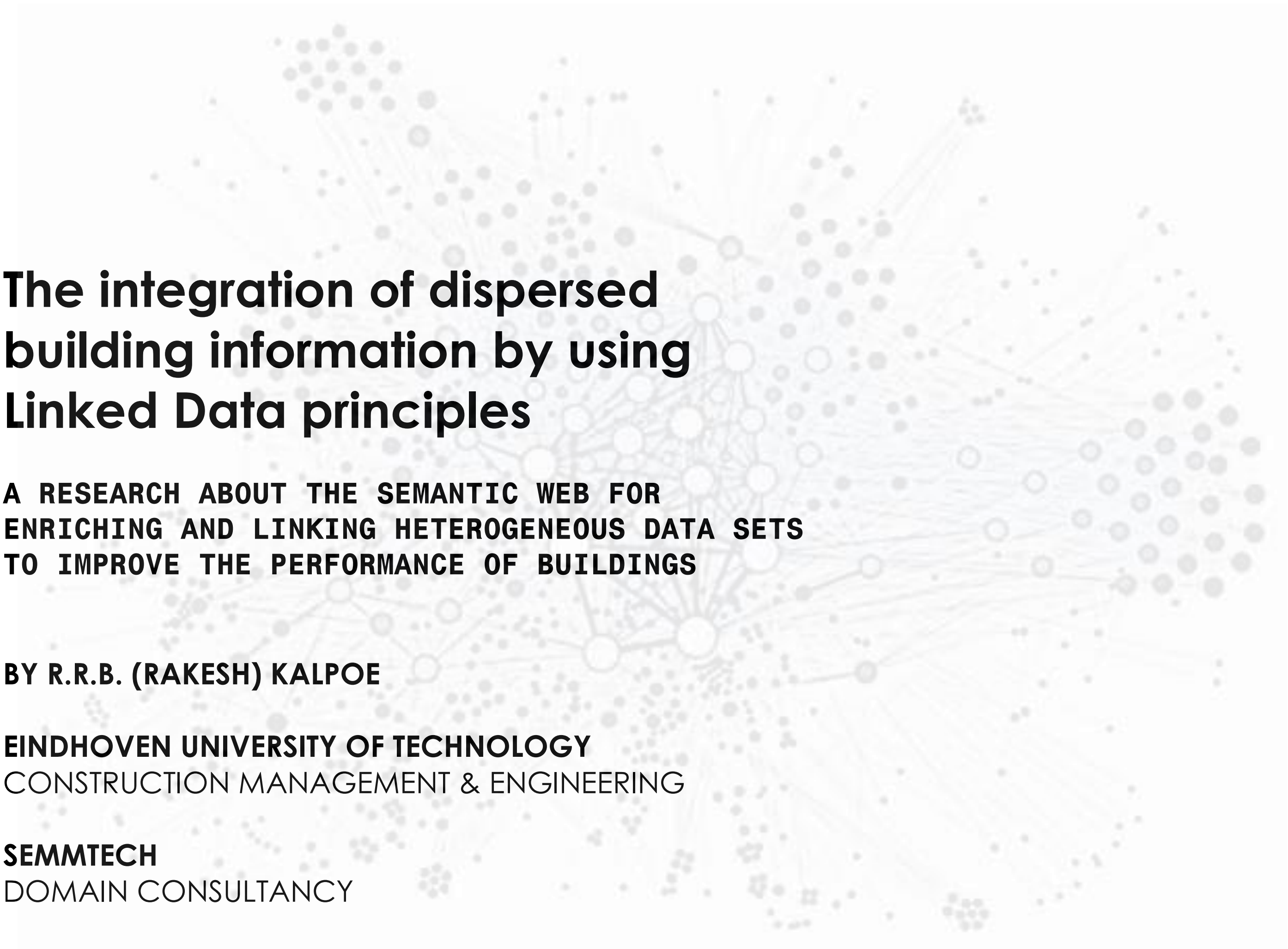
MASTER

The integration of dispersed building information by using Linked Data principles

a research about the semantic web for enriching and linking heterogeneous data sets to improve the performance of buildings

Kalpoe, R.R.B.

Award date:
2016



The integration of dispersed building information by using Linked Data principles

A RESEARCH ABOUT THE SEMANTIC WEB FOR ENRICHING AND LINKING HETEROGENEOUS DATA SETS TO IMPROVE THE PERFORMANCE OF BUILDINGS

BY R.R.B. (RAKESH) KALPOE

EINDHOVEN UNIVERSITY OF TECHNOLOGY
CONSTRUCTION MANAGEMENT & ENGINEERING

SEMMTECH
DOMAIN CONSULTANCY

July 8, 2016
: Final version

The integration of dispersed building information by using Linked Data principles

A RESEARCH ABOUT THE SEMANTIC WEB FOR ENRICHING AND LINKING HETEROGENEOUS DATA SETS TO IMPROVE THE PERFORMANCE OF BUILDINGS

Author	: R.R.B. (Rakesh) Kalpoe
Student number	: 0874207
E-mail	: r.r.b.kalpoe@student.tue.nl
Date of final presentation	: July 8, 2016
Place	: Eindhoven
UNIVERSITY	
University	: Eindhoven University of Technology (TU/e)
Faculty	: Faculty of the Built Environment
Master track	: Construction Management & Engineering (CME)
IN COLLABORATION WITH	
Company	: Semmtech
Division	: Domain Consultancy
GRADUATION COMMITTEE	
prof.dr.ir. B. (Bauke) de Vries	: TUE University supervisor (chairman)
dr.dipl.-ing J. (Jacob) Beetz	: TUE University supervisor
PhD- Candidate C. (Chi) Zhang	: TUE University supervisor
ir. J. (Jerrel) Yzer	: Semmtech External Supervisor

Preface

This master thesis is the final project of my master Construction Management & Engineering (CME) at the Eindhoven University of Technology. I'm glad to present my thesis to you as a reader which which will explore the 2 phenomena: Building Information Modeling and the Semantic Web. The last 6 months were a great opportunity for me to obtain new insights and set directions for formulating new goals in my future career.

This thesis has been conducted under the supervision of Semmtech which I would like to thank for allowing me to implement my research within a real world use case. Thereby, I would like to thank all colleagues for having a great time and especially my mentor Jerrel Yzer for his constructive feedback, support and insights. I would like to highlight the support by Daan Oostinga, Mike Henrichs, Nic Roest, Sander Stolk and Mohamed Moresey as well for helping me in their own way in regard to my graduation work and/or to me as a person.

I would like to express my gratitude for the feedback and support by TUE- mentor Jakob Beetz, as well for the discussions about the construction domain in general. Also I would like to thank PhD candidates Chi Zhang for his discussions about the Semantic Web and Thomas Krijnen for his advice regarding Building Information Modeling.

Also I would like to thank several companies for their input via conducted interviews. Thereby I would like to point out the help by Devlin Talman of Smits Bouwbedrijf for supporting me with a proper Revit – IFC mapping. Last but not least, I would like to thank my family and friends for their overall support.

I hope you all will enjoy reading this thesis as much as I carried out the research project.

Rakesh Ravi Bidjaipersad Kalpoe

Summary

Within the Dutch AEC domain government policies demand increasingly from construction projects, that they have to be implemented by performance based contracts. Because currently the number of such projects of the Dutch government in the Operation and Maintenance (O&M) phase increases, efficient contract management by the Dutch government is becoming more crucial in checking if the actual building performance complies to the agreed output specifications (OS). At the same time more heterogeneous non- geometric data is being produced in the O&M stage than ever before. Examples are: energy usage, indoor climate data, utility information, occupancy patterns, weather data, scheduling software, financial control etc. From literature study and conducted interviews with several companies, it has been found that often there is little interaction between these sensor data silos. However, cross- domain sensor data is seen as essential to understand the performance and optimize the operation of a building (by Facility Management) to ensure it is meeting the requirements of the organization/occupants. In short, building owners require more from their assets while the need to synchronize heterogeneous data sources become thereby more prevalent.

Even though the BIM approach has shown various improvements it has been criticized by academia as well as by construction businesses. Namely, it appears that the amount of non-geometric information within the AEC domain - especially data from the operational phase - cannot be stored in a central Building Information Model while it also provides limited interoperability in regard to sharing and integration of dispersed data sets.

The fundamental concept of Linked Data (which is based upon the Semantic Web) is that data is especially created with the mindset that it will be integrated and reused by others while being expressed by various vocabularies. Representing data by using the principles of the Linked Data model, will allow it to be combined with Linked Data from other relevant cross domain silos. In doing so, organizations can generate and extract additional value from current stand- alone repositories, across multiple disciplines.

So, because of the possibilities Linked Data provides, this thesis focused upon the following research question: "*In which way could linked sensor data be integrated into the BIM model to check the performance of its associated building based upon the agreed specifications during the operations & maintenance phase?"*

In order to provide an answer to this question, a use case was carried out. From the National Military Museum in Soest, 3 heterogeneous data sets were obtained, namely: A temperature dataset of 8 exposition rooms, a building model and 1 requirement. This requirement stated that a temperature sensor value of a certain room should always be between a specified lower boundary and a specified upper boundary.

From a brief analysis of these data sets, it was found that each data set could be linked together based upon each room. The building model describes namely the rooms (i.e. in a geometrical way) which serve at the same time as an installed location for sensors while it is being specified by the output specification. So it was sought to engineer a combined ontology which allows enriching and defining the requirements and temperature sensor data in an explicit way and could enable the visualization of which rooms in the associated building model do not comply to the OS.

The building information model was generated into a RDF data model (via a proven IFC-RDF converter) according to the IfcOWL ontology. The sensor data set was semantically expressed according to the following ontologies: SSN, time, QUDT, XSD and DUL. In regard to the requirement data set, it occurred that there were no tested (let alone widely accepted) ontologies. For this reason, an own

Systems Engineering (SE) ontology was engineered. Thereby an ontology pattern was used (multiple information models based upon the SE ISO 15288 standard). In order to convert the latter two data sets in actual RDF data models the generic CSVW procedure was used. The URI- pattern to identify resources in those 2 graphs was constructed as:
`http://{domain}/{type}/{concept}/{reference}`.

In total, 10 RDF graphs (1 IFC- RDF graph, a temperature RDF data set of each of the 8 exposition rooms and the OS data set in RDF) were imported into the same repository of a triplestore, after being successfully validated.

Thereafter, the sensor- and requirement data were linked to the IFC- RDF graph. This happened by linking room instances which represented the same resource together. For example, the rooms 191 (in the IFC- RDF graph), Hoofdthema1NederlandEnDeWereld (in the OS graph) and KrijgsmachtbredeThemaruimte1 (in the sensor data graph) all represent the same entity in the real world. The mapping procedure was conducted by using the `owl:SameAs` predicate.

Furthermore, the IF-ELSE logic of a SELECT SPARQL sub query to a remote triplestore was used for the retrieval of the GUIDS of building rooms of which the observed sensor values exceeded the specified upper and lower boundaries. This query was formulated in such a way it could make use of the REST architecture of the Web. The retrieved results were then used to provide virtually a RGB (Red-Green-Blue) color to the `IfcSpace` elements in the BIM model of the National Military Museum. The color red was used to indicate rooms that did not comply to the requirement, while green colored rooms did comply to the OS.

Finally, it can be said that the outcome of this research provides insights in how to express, access, integrate, retrieve and reuse data sets in a meaningful way by the means of open standards.
In this way, the client would (and/or the occupants would) be able to use a building that is able to satisfy their needs while the contractor is able to operate the building more effectively. Ultimately, the developed proof of concept in this research has shown that by using open Semantic Web technologies it is possible to improve interoperability between different building disciplines and thereby enhancing the building performance of construction objects within the AEC industry.

Samenvatting

Het overheidsbeleid eist in toenemende mate van de de Nederlandse bouwindustrie dat haar bouwprojecten worden uitgevoerd middels DBFMO- prestatiecontracten. Omdat op dit moment tevens het aantal DBFMO- projecten in de beheer- fase van de overheid alsmaar toeneemt, wordt efficiënt contract management steeds essentiëler bij het controleren of de feitelijke prestaties van gebouwen voldoen aan de overeengekomen outputspecificaties (OS). Tegelijkertijd wordt er in de beheerfase meer heterogene niet- geometrische sensor data geproduceerd als ooit tevoren. Voorbeelden hiervan zijn: energieverbruik, data over het binnenklimaat, aanwezigheidspatronen, weergegevens, financiele data etc. Op basis van literatuurstudie en interviews met verschillende bedrijven, is geconstateerd dat er vaak weinig interactie is tussen deze sensor data silo's. Echter, het samenbrengen van deze sensor data wordt gezien als een essentieel instrument om de gebouwprestaties te begrijpen en te optimaliseren (door het facilitair management) en daarmee ervoor te zorgen dat het gebouw voldoet aan de eisen van de eigenaar/bewoners. Kortom, eigenaren van gebouwen eisen meer van hun bouwobjecten, terwijl de noodzaak om heterogene gegevensbronnen te synchroniseren steeds belangrijker wordt.

Ondanks dat de BIM- ontwikkeling diverse verbeteringen heeft aangetoond binnen het bouwdomein, wordt het bekritiseerd vanuit zowel de academische wereld als het bedrijfsleven. Het blijkt namelijk dat de hoeveelheid niet- geometrische informatie binnen de gebouwde omgeving - en dan met name de data die wordt geproduceerd in de operationele fase - niet kan worden opgeslagen in een centraal BIM model. Daarnaast blijkt ook dat het model te beperkte compatibiliteit mogelijkheden kent ten aanzien van het delen en integreren van heterogene informatiestromen.

Het fundamentele concept van de Linked Data benadering (die tevens is gebaseerd is op het Semantisch Web) is dat gegevens zodanig worden gestructureerd, dat deze juist op eenvoudige wijze geïntegreerd als hergebruikt kunnen worden door anderen. Door data sets uit te drukken volgens de principes van het Linked Data model, is het mogelijk om de verrijkte informatie te linken met andere relevante (geisoleerde) data silo's. Op deze manier, is het mogelijk dat organisaties toegevoegde waarde te kunnen creëren middels individuele databases die in bezit zijn van verschillende disciplines.

Vanwege de mogelijkheden die Linked Data kent, richt deze scriptie richt zich op de volgende onderzoeksvraag: "Op welke wijze kunnen sensor gegevens worden gelinkt aan het BIM- model om zo de prestaties van het bijbehorende gebouw in de beheerfase te beoordelen op basis van de overeengekomen outputspecificaties?'

Om de onderzoeksvraag van een deskundig antwoord te voorzien, is er een casus uitgevoerd. Zo zijn er van het Nationaal Militair Museum in Soest de volgende 3 heterogene datasets verzameld: Een temperatuur dataset van 8 expositieruimten, een gebouw model en 1 outputspecificatie. Deze eis houdt in dat een gemeten temperatuursensor waarde van een bepaalde expositieruimte zich altijd tussen een bepaalde onder- en bovengrens dient te bevinden.

Op basis van een beknopte analyse van de datasets blijkt dat elke gegevensset met elkaar kan worden gekoppeld via de expositieruimten. Het gebouwmodel beschrijft namelijk de expositieruimte (o.s. op een geometrische manier), die tegelijkertijd als geïnstalleerde locatie dient voor de sensoren terwijl diezelfde ruimte wordt gespecificeerd door de outputspecificatie. Daarom is er in dit onderzoek gezocht naar een manier om een gecombineerde ontologie te ontwikkelen die het mogelijk maakt om de temperatuur- en outputspecificatie datasets te verrijken en te structureren op een expliciete manier en daarmee de ruimtes die niet voldoen aan de OS te visualiseren in het bijbehorende bouwwerkinformatiemodel.

Het IFC- bouwinformatiemodel werd omgezet in een RDF- informatiemodel (via een bewezen IFC- RDF converter) dat was uitgedrukt in de IfcOWL ontologie. De sensor data set werd semantisch uitgedrukt middels de volgende ontologieën: SSN, tijd, QUDT, XSD en DUL. Bij de outputspecificatie dataset, kwam naar voren dat dat er geen geteste (laat staan algemeen aanvaarde) ontologie bestond. Omwille van deze reden werd een eigen Systems Engineering (SE) ontologie ontwikkeld. Hiervoor werd er een template gebruikt in de vorm van enkele informatie-modellen die waren gebaseerd op de SE ISO 15288 norm. Bij het uitdrukken van ieder van de twee laatstgenoemde gegevensverzamelingen in een RDF informatiemodel is de generieke CSVW procedure toegepast. De URI- patroon om de entiteiten in die datasets uit te drukken werd geconstrueerd als:
`http://{domain}/{type}/{concept}/{reference}`.

In toaal waren er 10 RDF- informatiemodellen ontwikkeld (1 IFC- RDF graaf, een temperatuur sensor RDF data set voor elk van de 8 expositieruimten en een OS informatiemodel in RDF), gevalideerd en vervolgens geïmporteerd in dezelfde triplestore.

Daarna werden de sensor- en eisen data sets gelinkt met de IFC- RDF graaf. Dit gebeurde door het koppelen van de expositieruimten in elke dataset die dezelfde entiteit vertegenwoordigde. Bijvoorbeeld, de expositieruimten 191 (in de grafiek IFC- RDF), Hoofdthema1NederlandEnDeWereld (in de grafiek OS) en KrijgsmachtbredeThemaruimte1 (in de sensorgegevens grafiek) drukten allen dezelfde entiteit uit. Het linken van de datsets werd uitgevoerd met behulp van de `owl:sameAs` predikaat.

De IF-ELSE logica van een SELECT SPARQL was toegepast om de online triplestore te bevragen en zo de GUIDs op te halen van de expositieruimtes, waarvan de waargenomen temperatuur sensorwaarden de gespecificeerde boven- en ondergrenzen hadden overschreden. Deze vraag was zodanig geformuleerd dat het in staat was om gebruik te maken van de REST architectuur van het web. De opgehaalde resultaten werden vervolgens gebruikt om een virtuele RGB (rood-groen-blauw) toe te wijzen aan de de `IfcSpace` elementen in het BIM- model van het Nationaal Militair Museum. De kleur rood werd gebruikt voor kamers die niet voldeden aan de eis, terwijl de expositieruimtes die wel voldeden aan de OS groen waren gekleurd.

Tenslotte kan worden gesteld dat de uitkomst van dit onderzoek inzicht biedt in het uitdrukken, het integreren, het ophalen en opnieuw gebruiken van informatiestromen op een zinvolle manier door middel van open standaarden. Op deze manier is de cliënt (en/ of de bewoners kunnen) in staat om een gebouw aan te wenden die voldoet aan hun behoeften terwijl de aannemer in staat is om effectiever het gebouw te beheren. Uiteindelijk heeft de ontwikkelde prototype in dit onderzoek aangetoond dat door het gebruik van Semantisch Web-technologieën het mogelijk is, om de interoperabiliteit tussen de verschillende disciplines in de bouw te verbeteren en daarmee en daarmee de prestaties te vergroten van objecten binnen de bouwindustrie.

Table of contents

1. Introduction

1.1 Background

The amount and diversity of information is one of the most essential characteristics of a building project in the Architecture, Engineering and Construction (AEC) domain. Namely, while various domain experts work on the same project each of them have their own understanding of the project and deliver their own contribution by using its own software. Since the used information models are all part of one and the same project, a lot of information flows occur between the various involved parties of different disciplines. See figure 1A.

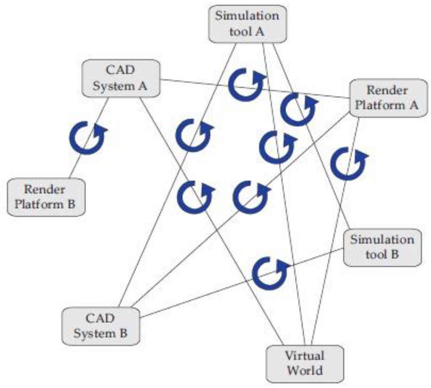


Fig. 1A The traditional approach of information exchange within the AEC domain (Pauwels, 2014).

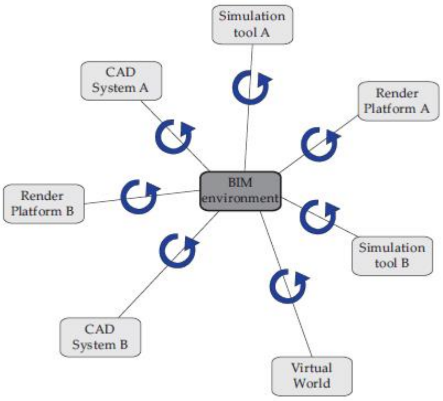


Fig. 1B The BIM approach of information exchange within the AEC domain (Pauwels, 2014).

In order to resolve this issue the Building Information Modeling (BIM) approach is increasingly becoming a standard within the AEC domain. This approach states that one central 3D building model is used as a centralized information structure. See figure 1B. Then, all information is stored in this central BIM model which can be accessed by diverse other applications in the AEC domain. Changes made to the design are applied to and stored into the BIM model and allow them to be directly available to other users.

Even though the BIM approach has shown various improvements it has been criticized within academic as well as in corporate domains. Namely, it appears that the amount of non-geometric information within the AEC domain –especially data from the operational phase - cannot be stored in a central Building Information Model. These data sources (including the BIM) are usually stored locally and are seldom connected with each other (Dankers, van Geel, & Segers, 2014). Furthermore, the open data model where upon BIM relies (which is called the Industry Foundation Classes (IFC)) is by itself not sufficient to enable interoperability with systems outside of the AEC domain (Curry et al., 2013). In this context, interoperability should be seen as the ability of information systems to integrate their information structures (or models) and "work together" effectively by means of information flows. This is noticed by Pauwels as well who stated that it is currently not possible to rely on the central information structure (IFC) for describing all building information (Pauwels, 2014).

As a result, the Linked Data strategy is increasingly getting attention within the AEC domain as one of the most promising approaches to tackle the interoperability challenge. It does this by separating the actual data from its authoring tools and relying on an data model in a linked open data structure (Pauwels, 2014). See figure 1C. Representing building data as a Linked Data model, will allow it to be combined easily with Linked Data from other relevant cross domain silos. In doing so, organizations can share, reuse and therefore improve interoperability between current stand-alone repositories, across multiple building domains. (Curry et al., 2013).

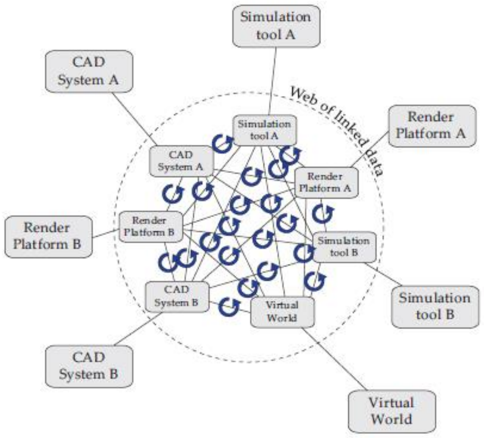


Fig. 1C The Linked Data approach of information exchange within the AEC domain (Pauwels, 2014).

1.2 Problem description

Within the Dutch AEC domain government policies demand increasingly that Public-Private Partnership projects, have to be implemented via integral contract types such as the Design, Build, Finance, Maintenance and Operate (DBFMO)- contract (Verweij, 2015). Because the number of DBFMO- projects of the Central Government Real Estate Agency (CGREA) in the operation & maintenance (O&M) phase increases, efficient contract management is becoming more crucial in checking if the actual building performance complies to the agreed output specifications (OS) (Algemene Rekenkamer, 2013).
At the same time more heterogeneous non- geometric data is being produced in the O&M stage than ever before. Examples are: indoor climate data, energy usage, utility information, occupancy patterns, weather data, scheduling software, financial control etc. Often there is little interaction between these sensor data silos. However, the reuse and integration of cross- domain performance sensor data is seen as essential to understand the performance and optimize the operation of the building to ensure it is meeting the requirements of the organization/occupants (Curry et al., 2013).

In short, building owners require more from their assets while the need to synchronize heterogeneous data sources become more prevalent. Yet literature states that Facility Management (FM) is still in its infancy in its adoption to advanced information models like BIM which hampers (automated) integration and reuse of data sets coming from other disciplines. For example, BIM- tools for the O&M phase have only recently become available on the market (C. e Eastman et al., 2011).
This has been stated as well by various parties (i.e. ISSO, Facilicom and Strukton) during conducted interviews. See appendices A, B and C. Facilicom stated that they use a so called Facility Management Information Model which they have to build up from scratch again. They also have to fill up the Building Performance System (BPS) manually. Also a data manager of Strukton mentioned that non- geometric information in Excel has to be merged manually (specifically: retyping or copy/pasting) because every data source is stored in its own rigid tabular structure. In another interview (Appendix D) a facility manager of The Ministry of Defense stated that data silos (created by the BPS) were not synchronized automatically while data sets were sometimes even distributed per mail. Therefore, he was not able to analyze relevant cross- domain data in order to gain a more elaborated overview of the building and improve his decision making to meet certain objectives. Finally, research shows that if

BIM is used anyway, the utilization in the operational phase is currently limited to the use as a static repository of information concerning building entities (Pauwels, 2014).

1.3 Research objective

So given (1) the need of an open information model which can enhance interoperability between various (non-) geometric data sources within the O&M phase and (2) the possibilities of Linked Data it seems beneficial to study the added value of Semantic Web technology within a DBFMO- project. Therefore, the objective of this research is to interlink non- geometric data sets produced by different disciplines to a BIM- model by using the Linked Data approach in order to perform performance assessments upon a building. The scope of this research will thereby be restricted to a provided use case by Semmtech. Therefore, in this report a focus will be put upon the building domains Systems Engineering (SE) within the design phase and Facility Management (FM) within the operations & maintenance stage.

1.4 Research questions

Based upon the mentioned objective and limitations in section 1.4 a main research question is formulated as follows:

> "In which way could linked sensor data be integrated into a BIM model to check the performance of its associated building based upon the agreed specifications during the operations & maintenance phase?"

The 6 sub questions that support answering the research question are formulated as:

1. What is Systems Engineering and Facility Management and to which extent do they relate to each other?
2. What is BIM and to what extent facilitates it information exchange between SE and FM at the moment?
3. In what way is it possible to improve information exchange by using Semantic Web technologies?
4. In which way could Linked Data sets be generated and aligned together?
5. How to develop a rule-based mechanism which is able to determine to which extent the output specifications are met?
6. In which way is it possible to visualize the results of the verification within in a building model?

1.5 Thesis outline

The structure of this report could be described in 10 steps. Firstly, a research method is explained that will be used throughout the project. Thereafter, the AEC domains Systems Engineering (SE) and Facility Management (FM) will be examined in how they relate to each other in part 3 and part 4. Thereby, a main focus is put in the structure of the data (or information) that is generally being produced during these buiding phases. Chapter 5 revolve around the BIM approach and how it facilitates information flows between these building domains at the moment while chapter 6 explores how Linked Data is able to improve the current state. Then, an actual use case will be obtained of which several data sets (produced by SE and FM) will be analyzed in chapter 7. The purpose of part 8 and part 9 is to structure and convert the analyzed data sets into Linked Data and connect (integrate) them together as one cohesive information model. Chapter 10 will then show how the rooms of a building model can be checked based upon the developed information model and how to visualize the results. Finally, this report will be ended by means of a conclusion and a discussion in chapter 11.

2. Methodology

The goal of this chapter is to set up an adequate research model. Therefore, literature study is conducted first which provides the means to design a structure which is able to elaborate upon the mentioned sub questions and thereby enable a comprehensive answer to the main question.

2.1 Methodological justification

Curry et al. proposed during their research a process to develop a semantic energy management application. This application could query data models that consisted of IFC data enriched by energy related data (Curry et al., 2013). This research model resembles greatly that of Nikman & Karshenas: In their study a BIM knowledge base was created from cross-domain data silos in order to develop an energy analysis application (Niknam & Karshenas, 2015). In conclusion they both identified the steps from obtaining unstructured energy related data and storing transformed Linked Data (of which its data format is called Resource Description Framework (RDF)) in adequate triple stores (which is a specific RDF database). They also noticed that a specific language was necessary to query this triple store. Because of the resemblance between his research and theirs, both methodologies are used as an overarching structure.

Radulovic et al. elaborated extensively the steps from obtaining raw data to a RDF transformation. The study stated that the generation of Linked Data in the AEC domain is still in its infancy. In order to stimulate a quicker adoption they developed (based upon the general rules to generate Linked Data) a set of specific guidelines to generate Linked Data related to energy consumption of buildings (Radulovic et al., 2015). Figure 2A shows the steps the team took to accomplish this.

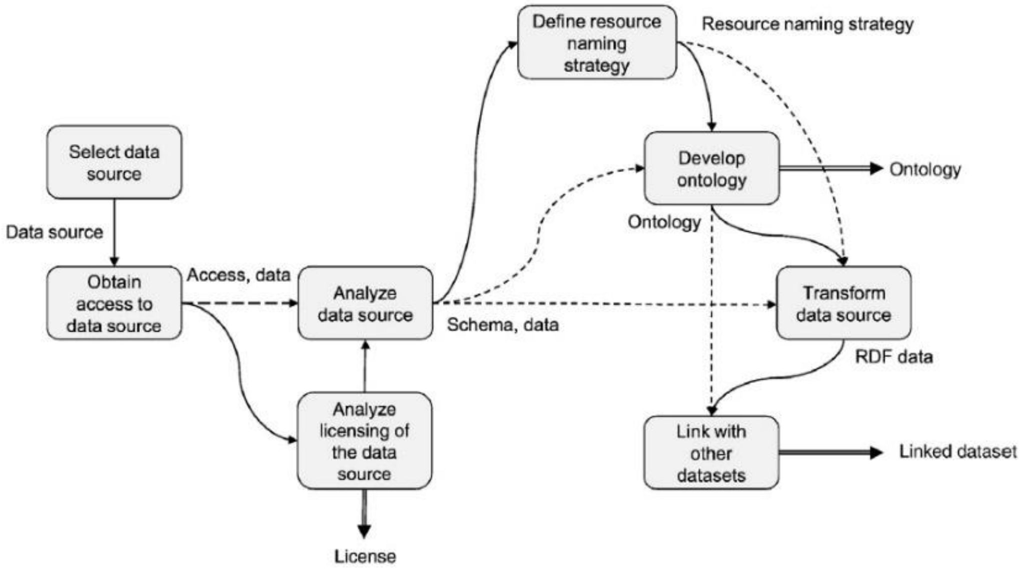


Fig. 2A A guide for a Linked Data generation process (Radulovic et al., 2015).

Figure 2A also shows that an ontology has to be developed first before the actual RDF translation can take place. However, it appears that this is an extensive process for which no single correct ontology engineering methodology exists. Namely, the "correctness" of an ontology depends on the usage by the application. It is therefore essential to start with a general template on which (when necessary) a more specific development vocabulary can be built upon. This general method is provided by Noy et al. who developed a ontology engineering guide called: Ontology Development 101. Their proposed methodology consist of

(1) determination of the domain and scope of the ontology (2) reuse of existing ontologies (3) list of important terms in the ontology (4) definition of classes and its hierarchy (5) definition of slots (6) definition of facets of slots (7) Creation of instances (Noy & McGuinness, 2001). Finally, in order to develop semantic tooling efficiently an adequate method is required. Davis provided a general and common approach in the software development domain to program an application. These steps can be enumerated as follows: (1) setup requirements (2) design (3) coding (4) testing like unit tests, acceptance test etc. (Davis, 1993). See figure 3.

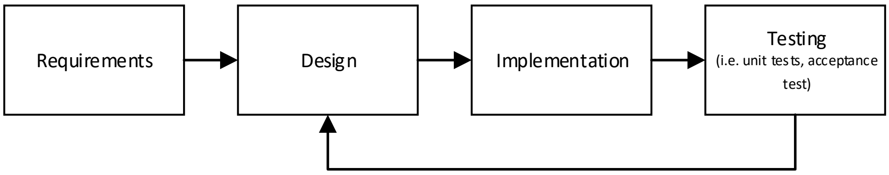


Fig. 2B A generic development approach within the Information Technology domain (Davis, 1993).

2.2 Research model

The research model is developed by combining the specific research questions with the general foundation as described in 3.1. Figure 2C shows the final model. Basically, the sub research questions determine the sequence of the phases, while the methodologies mentioned above provide efficient guidelines to answer them.

First, literature study will be conducted to gain insight in the aforementioned interoperability problem within the operations & maintenance phase. Hereby, concepts as Systems Engineering (SE), Facility Management (FM), Building Information Modeling (BIM) and Linked Data will be elaborated respectively. Thereafter, the verification process within a DBFMO- context is analyzed by using Business Process Modeling Notation (BPMN). In this way it is possible to capture the necessary requirements via the Unified Modeling Notation (UML) and define the scope of the research based upon the MoSCoW method which is followed by an ordinary Initial Data Analysis (IDA) of three data sets from a real world use case. Hereafer the actual Linked Data sets are generated by (where necessary) following best practices of i.e. Radulovic et al and Noy et al. When the Linked Data sets are mapped together and stored in a so called triplestore, a verification mechanism will be developed in order to determine the performance of the building. Finally a Linked Data tool is going to be built in order to visualize the results. Several validations will be performed to keep the future results aligned with the research goal during the process.

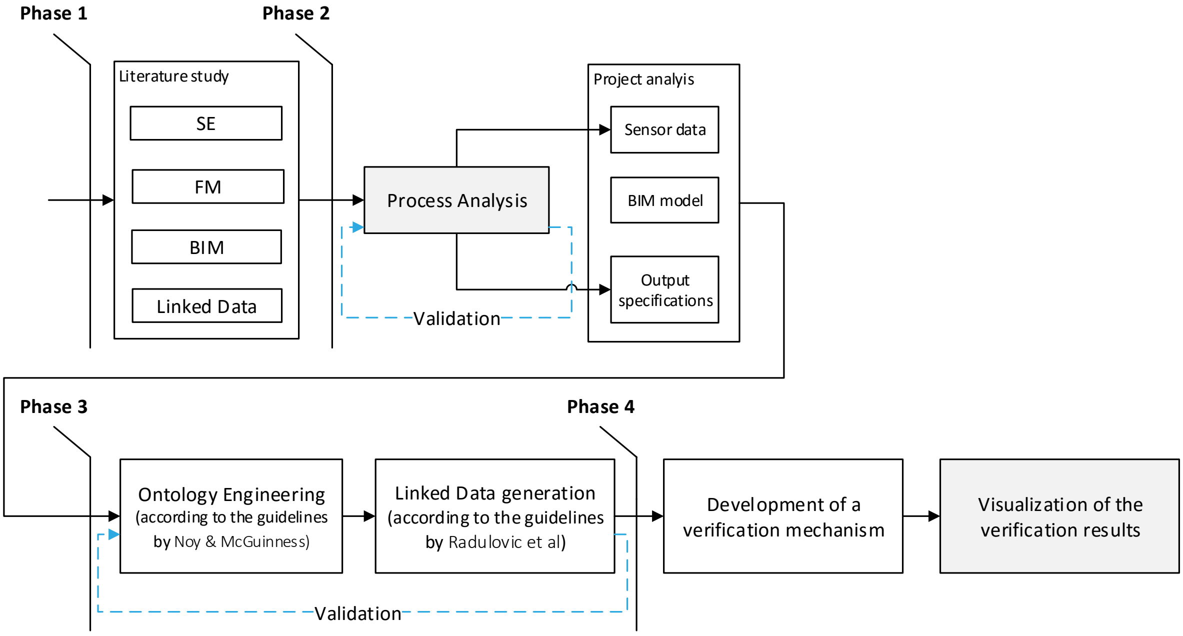

Phase 1
Phase 2
Literature study
SE
FM
BIM
Linked Data
Process Analysis
Validation
Project analyis
Sensor data
BIM model
Output specifications
Phase 3
Phase 4
Ontology Engineering (according to the guidelines by Noy & McGuinness)
Linked Data generation (according to the guidelines by Radulovic et al)
Development of a verification mechanism
Visualization of the verification results
Validation

Fig. 2C An high level view of the overall process of this research project.

3. Systems Engineering

The goal of this chapter is to examine the domain of Systems Engineering (SE). Firstly, the meaning of the term will be explored. Subsequently, the core activities, approaches and standardization efforts are elaborated. Thereby a description is provided about the data that is being produced during the processes. In regard to this research, this part provides an brief explanation about how this domain correlates with the FM domain during a DBFMO project.

3.1 Systems Engineering (SE)

It appears to be hard to find a single definition of SE because the existing literature provides multiple interpretations of the term. Though, the most widely accepted explanation is provided by the International Council on Systems Engineering (INCOSE): "Systems Engineering is an interdisciplinary approach and means to enable the realization of successful systems. SE considers both the business and the technical needs of all customers with the goal of providing a quality product that meets the user needs" (INCOSE, 2006).

Basically, a system can be seen as an integrated composite of people, products, and processes that provide a capability to satisfy a stated need or objective (Freeman, 2015). Systems can be grouped together as well in order to create more complex systems. See figure 3A for an example of such an overall system. Furthermore, the figure illustrates the decomposition of abstract systems into more concrete subsystems as well. In such situations these systems can be interpreted as system elements (also subsystems) in an overarching and hierarchical system. Examples are a so called System Of Systems or Federation of Systems. (BKCASE Editorial Board, 2014).

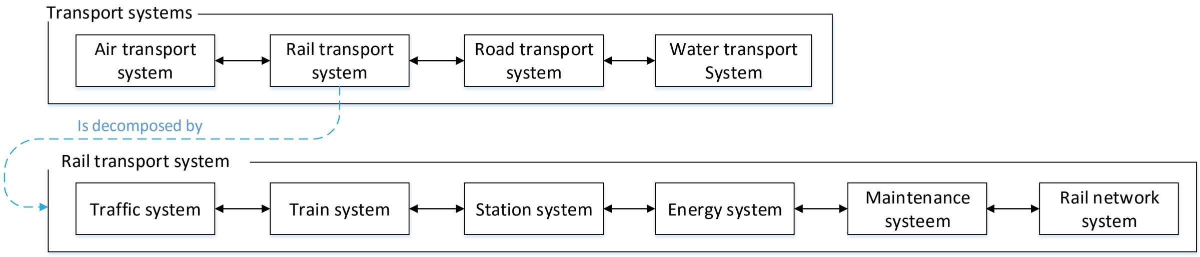


Fig. 3A An example of a System Of Systems (Based upon: Werkgroep Leidraad Systems Engineering et al., 2013).

3.2 The SE process

Key activities of SE processes are the decomposition of a system (as mentioned in part 3.1) and associated verification which provide jointly a holistic overview of a project (Douglass, 2016). A system can be decomposed in basically three breakdown structures (Werkgroep Leidraad Systems Engineering et al., 2013):

1. A Requirements Breakdown Structure (RBS) which form an hierarchy of requirements of a system;
2. A Functional Breakdown Structure (FBS) which specifies every function that must be addressed by a certain system;
3. The System Breakdown Structure (SBS) is a hierarchy of system elements, related life cycle processes and stakeholders.

These breakdown structures are related to each other as follows. Based upon the RBS, systems can be designed (according to the SBS), which has to fulfill certain functions which are specified within the

FBS. Throughout this iterative process, verification is imposed to check whether or not the functioning of a product, service, or system complies with a specification. Figure 3B illustrates this process.

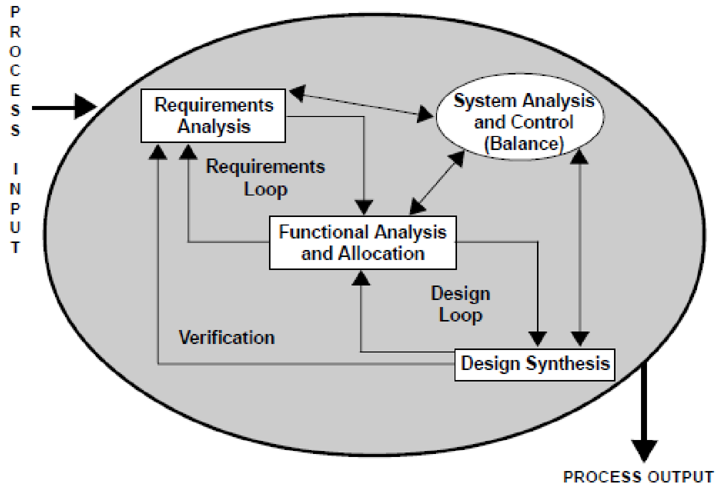


Fig. 3B A overview of the SE key activities (INCOSE, 2006).

Literature show that there are different approaches for executing these SE key tasks . The most common procedure (which is the facto standard as well) is the sequential V- approach. See figure 3C. The V- model highlights the need to define verification plans during requirements development, the need for continuous validation with the stakeholders, and the importance of continuous risk and opportunity assessment (Haskins, 2006).This is illustrated in the rigid distinction between the left and right side of the model. Hereby, the left side of the "V" illustrates the top- down decomposition process in subsystems (i.e. a building story). The right leg of the "V" represents the bottom- up process of implementation and verification of system components to the system level (INCOSE, 2006). A relevant example, is the RBS of which can be used to verify the intended operational use and ensure adequate maintenance during its life cycle (Ryen, 2008).

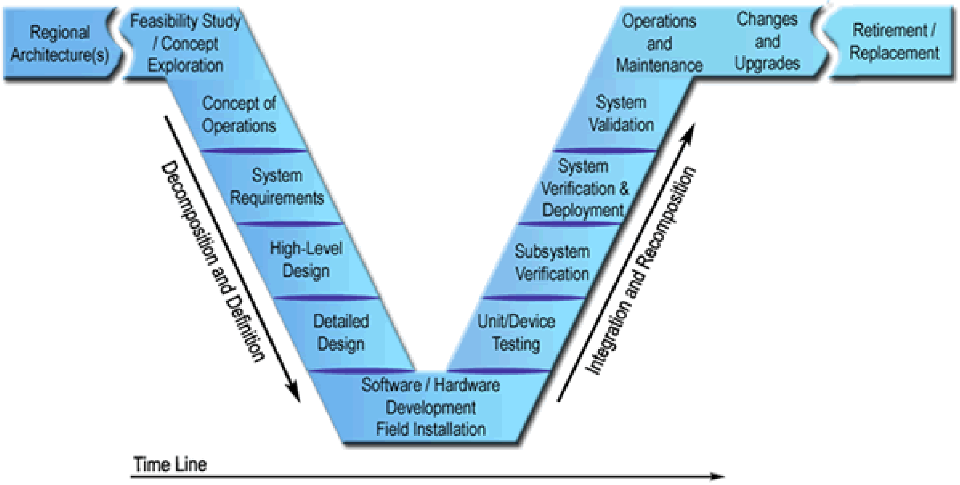


Fig. 3C The position of the operational building phase at the right wing of the V- model (Ryen, 2008)

The other extreme is the (agile) incremental approach. Within this approach SE and associated engineering disciplines deliver their products within short iterative intervals. This means that relatively small slices of the desired functionality of systems are being specified, implemented and verified before focusing on the other aspects of the performance of the functionality by the system. However, it appear that this method cannot be applied within the AEC domain because of the typical long lead times to create physical products (Douglass, 2016). Because each approach has its own (dis)advantages and each actual project is different in essence, several hybrid solutions has been developed during the last decade. (Freeman, 2015).

The execution of SE activities according to these approaches via a collection of terminology, tools and associated techniques have been standardized during the years. It appears that in 1969 the DOD was the first standard which was used to manage the military programs of the United States of America (USA). During the following years, a variety of SE standards have been developed from an increasingly commercial perspective. The three SE standards which are now commonly used are (Locatelli, Mancini, & Romano, 2014) :

1. ANSI/EIA-632 (2003) focuses on the early stages of a system's life cycle. It mainly describes SE “processes” and their relationships for the actual implementation.
2. IEEE (2005) focuses mainly on the “development stage” of a generic system. In general, IEEE (2005) provides also the most detailed SE processes.
3. ISO/IEC 15288:2008 (2008) provides a generic perspective of the entire life cycle of a system and describes SE processes via the highest level of abstraction.

3.3 Data from SE

The core products that serve as input for supporting the tasks by Facility Management are the system requirements themselves, an operations & maintenance plan and performance data (Ryen, 2008). Within conventional SE processes, all of these engineering data are represented either as textual specifications or occasionally as schematic drawings.

However, in the last 20 years, descriptive models have been introduced as a better way of creating, managing, and verifying engineering data than textual specifications (BKCASE Editorial Board, 2014). INCOSE defines this type of practice as Model Based Systems Engineering (MBSE) which holds the following definition: “The formalized application of modeling to support system requirements, design, analysis, verification, and validation activities” (INCOSE, 2007).

Thereby, the use of formal standards for creating models and defining data exchanges is seen as an important enabler for integrating and reusing data during SE processes. Identified examples of such modeling languages are the Unified Modeling Language (UML) and the Web Ontology Language (OWL). Standardized data exchanges are possible thanks to widely adopted data models like the Extensible Mark-Up Language (XML) and the Resource Description Framework (RDF) (BKCASE Editorial Board, 2014).

4. Facility Management

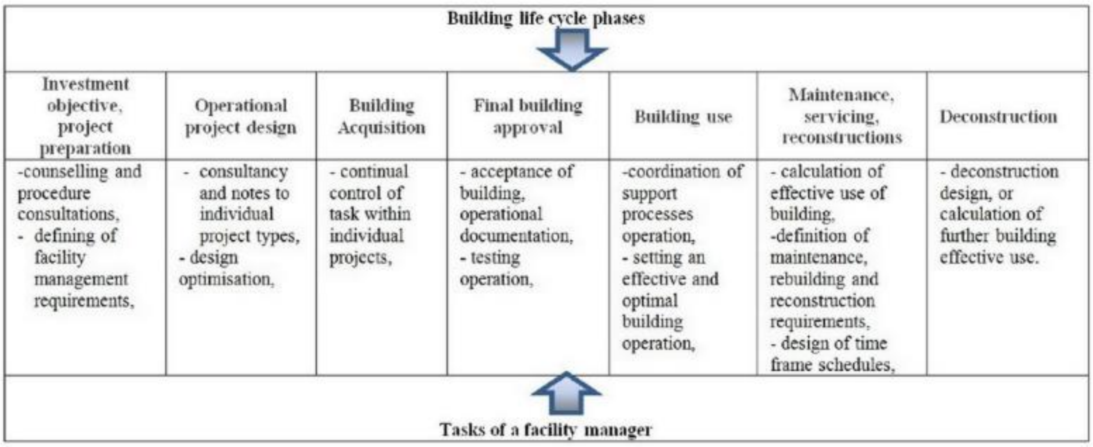
The purpose of this part is to explore the domain of Facility Management (FM). Firstly, a definition will be provided which will be used throughout this research. Furthermore, the core activities and approaches are mentioned. Thereby an extensive explanation is provided about the data that is being produced by the FM processes. For the sake of this research, the chapter explains how this domain is interlinked with the SE domain during the building life cycle (from the perspective of this research).

4.1 Facility Management (FM)

The domain of FM is defined in various ways as well. From the context of this research though, the explanation by the International Facility Management Association (IFMA) can be used: "A method whose task in organizations is to mutually harmonize employees, work activities and the work environment that includes principles of business administration, architecture and humanities and technical sciences" (Potkany, Vetrakova, & Babiakova, 2015). More concretely: Building's operations & maintenance (O&M) includes all services required to assure the built environment will perform according to the functions for which a building was designed and constructed (WBDG, 2015). In addition, the goal of FM has increasingly been commercialized like the (economic) maximization of building functions while (still) ensuring occupants wellbeing (Dawood, Vukovic, & Kassem, 2015).

4.2 The FM process

Though, FM plays a key role in the operation of an organization, it could be incorporated during other phases of a building life cycle as well (Mrackova, E., Hitka, M., Sedmak, R. 2014). In general, an ideal approach is if the role of FM is implemented already in activities during the initial phases of a building life cycle, such as Systems Engineering. Such an approach has multiple advantages, like reducing investment and operational costs (Miske, 2010). Figure 3F shows the diverse responsibilities of FM during these stages.



Building life cycle phases

Investment objective, project preparation	Operational project design	Building Acquisition	Final building approval	Building use	Maintenance, servicing, reconstructions	Deconstruction
-counselling and procedure consultations, - defining of facility management requirements,	- consultancy and notes to individual project types, - design optimisation,	- continual control of task within individual projects,	- acceptance of building, operational documentation, - testing operation,	-coordination of support processes operation, - setting an effective and optimal building operation,	- calculation of effective use of building, -definition of maintenance, rebuilding and reconstruction requirements, - design of time frame schedules,	- deconstruction design, or calculation of further building effective use.

Tasks of a facility manager

Fig. 4A An overview of the activities by FM (Potkany et al., 2015).

The operational activities by the FM discipline can be grouped in two major categories. The first category relates to making sure that the operation of the facility complies with certain specifications (according to the RBS) and regulations (i.e. NEN 2767). The second group of functions is mainly focused upon conditioning and maintenance of the building components. Because of the context of this research a focus has been put upon the first category.

4.3 Data from FM

In order to comply to the specified requirements by SE, extensive sensor data and information from various fields and disciplines is necessary (as already stated in 1. Introduction). In general, such building operational information consist of the following four data types: (Moon, Kim, & Choi, 2013):

> 1. Monitoring data which refers to the information measured that relates to building energy. This data includes information about building operation schedule, indoor climate, occupancy etc.
> 2. Forecasting data which contains the predicted information about weather and occupancy. This data is used to condition the building environment during future periods.
> 3. Control data which pertains to the information about the building control signals.
> 4. Simulation data which includes the results from simulation programs. It can be used to calibrate the simulation model and can be used for building control based upon simulations.

Traditionally, this FM sensor data and information are organized and maintained as data points in dispersed information systems such as Computerized Maintenance Management Systems (CMMS), Electronic Document Management Systems (EDMS), Building Automation Systems (BAS), etc.

Such data points seem to have different definitions across the literature. However, they all describe them as an addressable point of interaction between the control system and its domain object (i.e. indoor climate). Every data point has usually the following metadata associated with it (Domingues, Carreira, Vieira, & Kastner, 2016):

> **1. Access type:** Data points usually offer one of the three access types: read, write or both. Readable data points are read-only and usually relate to sensor devices. Writable data points are write-only and relate to updating the system's state.
>
> **2. Datatype:** In addition, the datatype tells applications how the information is structured when they read from a data point and how it must be structured when writing to that data point. Moreover, datatypes can have semantic information associated with them, usually represented by a unit. For instance, a data point's value can represent a room temperature in Celsius.
>
> **3. Installed location (and influence zone):** Knowing the installed location of a data point is essential, especially if that data point belongs to a sensor device. Besides, data points also have a zone of influence which may not be the same as the installed location. For example, a heating, ventilating, and air conditioning system (HVAC) usually occupies one room in the building and affects several other rooms.
>
> **4. Value update rate for reading and writing operations:** Data points that provide a read-access type should ensure a regular value update rate which is known as smallest sampling (time) interval. On the other hand, data points that support writing operations may provide a maximum rate at which writings can be performed.

5. Building Information Modeling

The purpose of this part is to explore the centralized Building Information Model(ing) (BIM) approach within the AEC domain. Currently, (the origin of) this phenomenon and its associated benefits has already been extensively investigated by various literature in multiple domains (C. Eastman et al., 2011; Bryde, Broquetas, & Volm, 2013). This chapter therefore starts directly with providing a technical definition of BIM based upon existing literature. Then, the essential BIM processes that enable interoperability will be described in 4 successive chapters. See figure 5A for a generic overview. This description is limited to the leading open data standard Industry Foundation Classes (IFC), even though the BIM approach is supported by various other open XML- based standards like gbXML, ifcXML, BCF and CityGML as well. Due to the context of this research, a focus is put on the exchange of non- geometric BIM information between stakeholders of the design phase and operations phase and associated limitations.

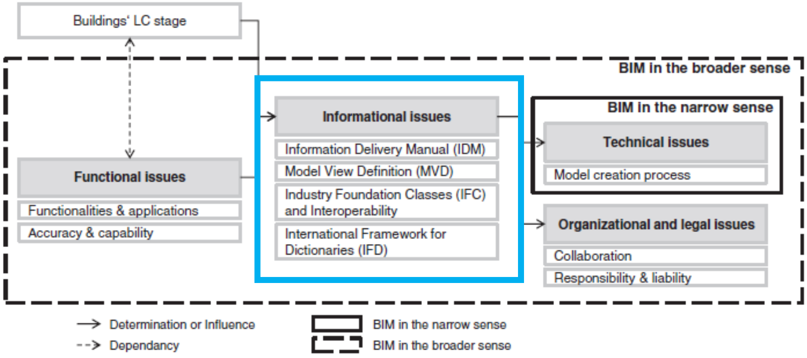

Buildings' LC stage
BIM in the broader sense
Informational issues
Information Delivery Manual (IDM)
Model View Definition (MVD)
Industry Foundation Classes (IFC) and Interoperability
International Framework for Dictionaries (IFD)
BIM in the narrow sense
Technical issues
Model creation process
Functional issues
Functionalities & applications
Accuracy & capability
Organizational and legal issues
Collaboration
Responsibility & liability
Determination or Influence
Dependancy
BIM in the narrow sense
BIM in the broader sense

Fig. 5A A holistic view of the essential BIM processes (Volk, Stengel, & Schultmann, 2014).

5.1 Building Information Modeling (BIM)

As already depicted in chapter 1, one of the latest approaches which is embraced within the global AEC domain is the BIM approach whereby one central 3D building model is used as a centralized information structure. Since all information is stored within a central BIM model, it can be accessed by various construction- related applications (i.e. Revit Architecture, Solibri, Relatics) of different stakeholders. In this way, all parties can use the available information during the building life cycle (Pauwels, 2014).

BuildingSMART, a neutral organization that plays a key role in the worldwide implementation of BIM in the AEC industry, defines BIM as a “shared digital representation of physical and functional characteristics of a facility founded upon open standards for interoperability.” This product model could then be employed for decision-making throughout the lifecycles of buildings. The ultimate goal is to enhance “collaboration by different stakeholders at different phases of the life cycle of a facility to insert, extract, update or modify information in the process to support and reflect the roles of that stakeholder” (International Alliance of Interoperability (IAI), 2007).

In this context, a product model can be seen as a formal information model that complies to agreed data structures. Therefore, it is possible to structure engineering information about construction elements in so called classes (Watson, 2011). Hereby a class can be defined as a (standardized) template or set of instructions to build a specific type of object (International Alliance of Interoperability (IAI), 2007).

Classes may have geometric or non-geometric attributes with functional, semantic or topologic information. For example, functional attributes can be installation durations or costs. Semantic attributes hold attribute such as connectivity, containment, aggregation or intersection while topologic attributes provide e.g. information about objects' locations, adjacency or perpendicularity. (C. e Eastman et al., 2011).

Such classes then allow the creation of any number of object instances, with forms that vary, depending on the determined parameters and relationships with other objects. This object- oriented approach enables users to develop their own information objects like a wall, slab, or roof and even develop object libraries for specific purposes (Volk et al., 2014).

5.2 Industry Foundation Classes (IFC)

ISO 10303 is a comprehensive ISO standard for the computer interpretable representation and exchange of (the previously mentioned) product models. The standard is often referred to as the STEP (Standard for the Exchange of Product model data) Standard. The STEP standard is divided into different parts, namely: Description Methods , Information Models, Application Protocols, Implementation Methods, and Conformance Tools. See figure 5B.

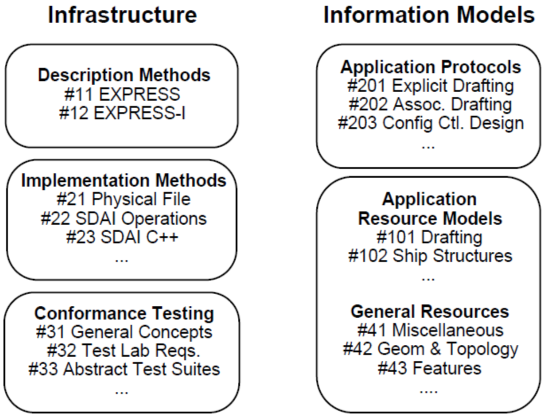


Fig. 5B An high level overview of the ISO 10303 standard (Loffredo, 1999).

Hereby the EXPRESS language (ISO 10303-11) is the main Description Method of STEP and should be seen as a standard data modeling language for data. The EXPRESS language can be considered as technology independent and consists of language elements which allow an explicit data definition. In EXPRESS, a number of declarations can be made, specifically: TYPE, ENTITY, SCHEMA, CONSTANT, FUNCTION, PROCEDURE (WHERE) RULE. The open IFC schema is built upon various IFC- classes (e.g. IfcBuilding, IfcSpace) that are specified by this EXPRESS data definition language. At the moment there are several available IFC EXPRESS schemas, including the most well-known IFC2X3.exp.

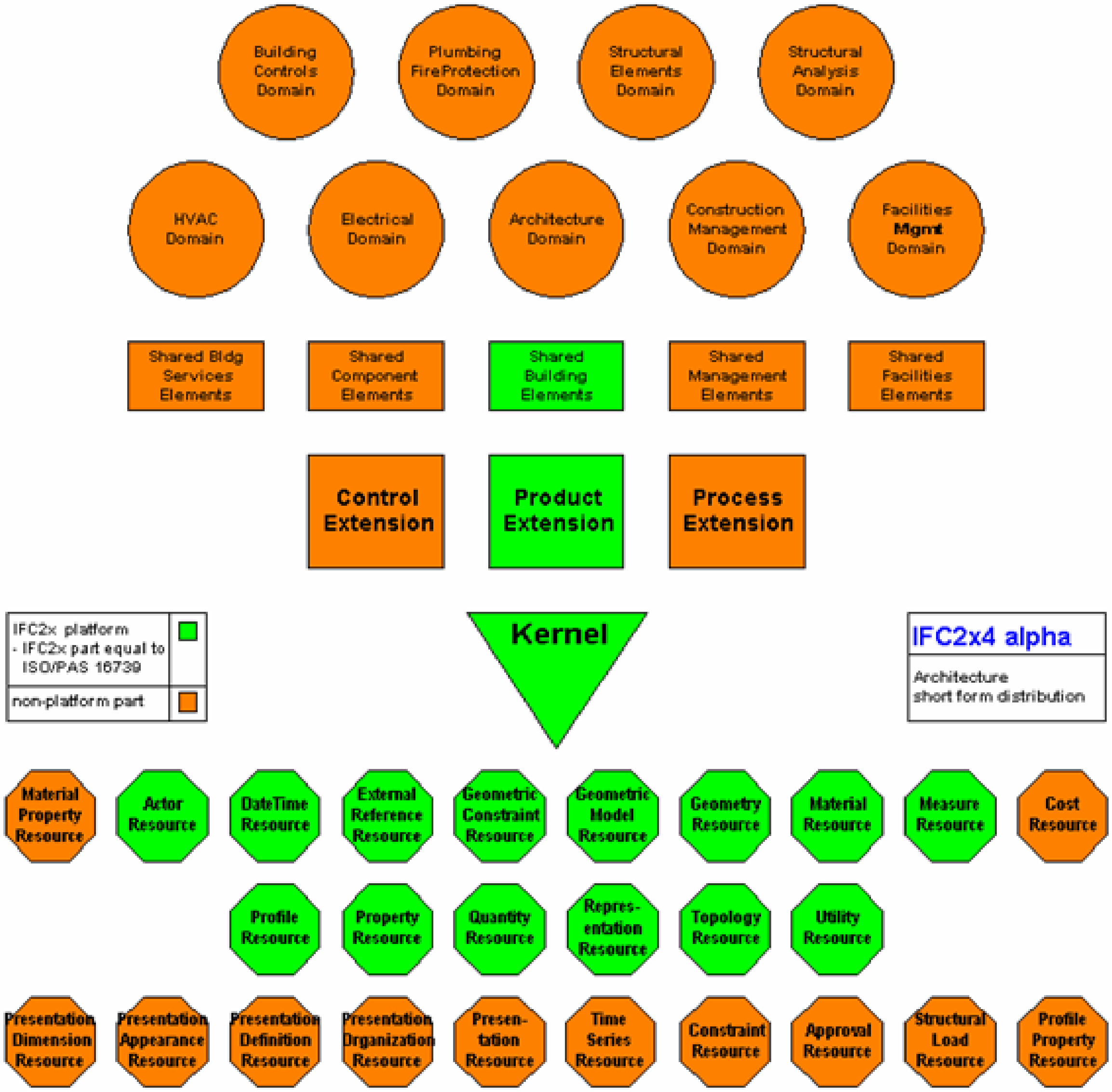

Building Controls Domain
Plumbing FireProtection Domain
Structural Elements Domain
Structural Analysis Domain
HVAC Domain
Electrical Domain
Architecture Domain
Construction Management Domain
Facilities Mgmt Domain
Shared Bldg Services Elements
Shared Component Elements
Shared Building Elements
Shared Management Elements
Shared Facilities Elements
Control Extension
Product Extension
Process Extension
IFC2x platform - IFC2x part equal to ISO/PAS 16739 🟩
non-platform part 🟧
Kernel
IFC2x4 alpha
Architecture short form distribution
Material Property Resource
Actor Resource
DateTime Resource
External Reference Resource
Geometric Constraint Resource
Geometric Model Resource
Geometry Resource
Material Resource
Measure Resource
Cost Resource
Profile Resource
Property Resource
Quantity Resource
Repres-entation Resource
Topology Resource
Utility Resource
Presentation Dimension Resource
Presentation Appearance Resource
Presentation Definition Resource
Presentation Organization Resource
Presen-tation Resource
Time Series Resource
Constraint Resource
Approval Resource
Structural Load Resource
Profile Property Resource

Fig. 5C A high level visualization of the IFC architecture (BuildingSMART, n.d.).

IFC schemas are structured according to a so called IFC Object Model architecture which provides a modular structure for its IFC classes. In essence, this structure can be conceptualized in four hierarchical layers. See figure 5C. Within each conceptual layer a set of model schemas are defined. The base layer provides Resource elements. The second overlay (the kernel layer) groups the Kernel and several Core Extensions. The third conceptual layer (which is called the interoperability layer) provides a set of modules defining concepts or objects common across multiple application types within the AEC domain. Finally, the fourth and highest layer in the IFC Object Model is the Domain/Applications Layer: It provides a set of modules tailored for specific AEC industry domain or application types.

Each schema groups a (hierarchical) set of entities. For example, `IfcSpace` lies within the ProductExtension schema and is related to the breakdown structure of `IfcSpatialStructureElement`. Namely, this class structures `IfcSpace` along with the following entities: `IfcProject`, `IfcSite`, `IfcBulilding` and `IfcBuildingStorey`. Herefore the decomposition relationship `IfcRelAggreagates` is used to link the (instances of the) classes. See Appendix H. `IfcSpatialStructureElement` is considered to be the primary decomposition of a project model into manageable spatial subsets and is essential for data exchanges (Liebich, 2009).

The IFC architecture operates on a 'ladder principle'. At any layer, a class may reference (i.e. inherit attributes) from a class at the same or lower layer but may not reference a class from a higher layer (Borgo, Sanfilippo, Aleksandra, & Terkaj, 2015). This essential aspect is illustrated by elaborating the IFC data model in detail by using the `IfcSpace`- class. The EXPRESS definition of an Ifc2x3Space is:

```
ENTITY IfcSpace
      SUBTYPE OF (IfcSpatialStructureElement);
            InteriorOrExteriorSpace: IfcInternalOrExternalEnum;
            ElevationWithFlooring: OPTIONAL IfcLengthMeasure;
      INVERSE
            HasCoverings: SET OF IfcRelCoversSpaces FOR RelatedSpace;
            BoundedBy: SET OF IfcRelSpaceBoundary FOR RelatingSpace;
END_ENTITY;
```

From this specification it is possible to see that the `IfcSpace` entity references to a Enumeration Data Type called `IfcInternalOrExternalEnum` and could be referenced from a `IfcRelCoversSpaces` class.

The `IfcSpace`- class could also be visualized by using EXPRESS-G. EXPRESS-G should be seen as a graphical modeling notation developed within SO 10303 and is used for IFC definitions as well. See figure 5E for an Ifc2X3Space- class in EXPRESS-G. Using this language users can draw classes, attributes of classes and the relationships that exist between classes. For example, the solid line from `IfcSpace` to `IfcInternalOrExternal` means that the attribute has to be defined in order to create an IfcSpace object.

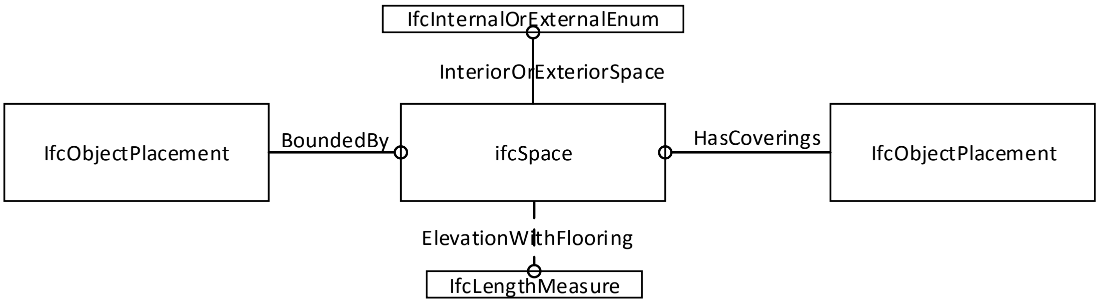


Fig. 5E An 2X3IfcSpace in EXPRESS-G (based upon: buildingSMART, n.d.).

An actual `IfcSpace`- object (the actual product model) can be exchanged via data files (.spf or .p21- files). These data files are clear text files following the so called STEP physical file format (ISO 10303-21). The Part 21 provides specifications to order EXPRESS-defined data so exchange between databases and CAD systems can take place. Examples are that a data file should have a “header”- section and a “data”- section or that each class instance should be represented in one line. IFC- SPF is a data model text format following this STEP- protocol while having the file extension ".ifc". Though other formats exist like IFCXML, this is the most widely used IFC exchange format. Below an example is provides of an STEP- specified Ifc2X3Space- object:

```
#191= IFCSPACE('0x2ZPKRKH3UgcA5UXfu_mq',#41,'1',$,$,#159,#187,
'EXAMPLE',.ELEMENT.,.INTERNAL.,$);
```

As is shown above, an `IfcSpace-` class (with local identifier #191) is constructed whereby the attributes between the brackets are its parameters. An “$” means that a parameter is not specified while e.g. #41 refers to a class on line 41 in the IFC- SPF file. The `IfcSpace`- entity consist of more parameters then is specified in its EXPRESS definition, because an IFC class is able to inherit attributes from classes in other layers according to the ladder-principle. For instance, the first parameter which represents a Global Unique Identifier (GUID) is inherited from the class `IfcRoot` that resides in the Kernel schema (which is one layer lower than `IfcProductExtension`). All classes or attributes which a class is able to reference to or is referenced from are defined in its associated Inheritance Graph[1].

The most recent version Ifc2x4 has about 800 entities (data objects), 358 property sets, and 121 data types (C. e Eastman et al., 2011).

5.3 Level of Detail (LOD)

During the life cycle of buildings various design, engineering, construction, maintenance and deconstruction functionalities (i.e. class detection or quantity takeoff) and potential applications (i.e. Solibri) require each a different capability of BIM (C. e Eastman et al., 2011).

These functionalities are usually inherent to either 3D, 4D or 5D BIM. 4D is achieved by linking the functional time attribute to building elements and space objects of the 3D-model. Then time parameters may describe e.g. the installation date and time of building elements. 5D BIM is accomplished by adding the cost dimension (specifically stated: adding cost attributes for particular times during the building lifecycle). This allow expert systems to use these underlying BIM data to support, extend, calculate or simulate specific cost analyses (International Alliance of Interoperability (IAI), 2007).

Furthermore, the degree of information needed by each functionality (and thus stakeholder) is different and require each a certain accuracy, type, information richness and timeliness of the underlying data to fulfill their purposes. This degree of BIM is called Level Of Detail (LOD). It defines geometric and non-geometric attribute information provided by a model component, often referenced to a point of time, building lifecycle stage or to a contractual responsibility (C. e Eastman et al., 2011).

Usually the required LOD by the functionalities are based upon process maps called Information Delivery Manuals (IDM). Such maps describe the logical flow of activities and the deriving information as well as the involved parties delivering specific functionalities (Volk et al., 2014).

5.4 Information Delivery Manual (IDM)

The Information Delivery Manual (IDM; ISO 2010a) should be considered as a business process modeling language. The main goal of IDM is to document information that needs to be exchanged to perform a task in a process (BuildingSMART, 2010).

As a product, the IDM extends Business Process Modeling Notation (BPMN). Thereby IDM focusses on in-depth descriptions of information elements (such as classes and attributes) and their exchange through object- oriented models. The IDM framework defines the functionality-related exchange of process information in BIM through process maps, interaction maps and the associated Exchange Requirement Model (ERM). The process maps describe the order of undertaken activities within a particular topic, the actors' roles and required, created and consumed information. The goal of interaction maps is to define roles and transactions for a specific purpose or functionality (P. C. M. Eastman, Tech, Ga, & Eastman, 2011). The ERM can be seen as a technical solution which defines a

[1] See: http://www.buildingsmart-tech.org/ifc/IFC2x3/TC1/html/ifcproductextension/lexical/ifcspace.htm.

“set of information that needs to be exchanged to support a particular business requirement” or functionality and correlates with the so called Model View Definition (MVD).

5.5 Model View Definition (MVD)

MVD definitions depends upon the required functionality (defined in the IDM) and the referred BIM information objects and associated attributes. Therefore, an IFC-based MVD is a model subset of an IFC schema- instance with respect to the requirements from end users of the desired functionality (Zhang, Beetz, & Vries, 2013). This means, that a MVD is able to structure relevant information to improve efficient information flow between stakeholders in building-related processes such as energy analysis or quantity takeoff.

For maintenance functionalities, Construction-Operations Building Information Exchange (COBie) is the predominant open standard MVD to exchange non- geometrical data such as contact and general facility information (i.e. attributes) about spaces, zones, floors, components etc. (Volk et al., 2014).

So COBie can be considered as a performance-based specification for facility asset information delivery. Two core types of assets are included in COBie: equipment and spaces. It aims to help the diverse project team (like SE) to organize its approved engineering data during design and construction and deliver an electronic O&M manual (including an as-built BIM) with little or no additional effort. For example, the standard is able to capture SE data for rooms in so called room data sheets: These sheets include non- geometrical information about the room including its name, acoustics, ventilation and environmental conditions. This COBie data may then be imported directly into information systems like Computerized Maintenance Management Systems (CMMS) as mentioned in section 4.3 (WBDG, n.d.).

COBie data is available in two main formats depending upon the user and building stage. Namely, information exchanges between machines during the design process are likely to use the ISO- SPF files. For human reading it is possible to translate the engineering data into a spreadsheet (i.e. Excel).

5.6 buildingSMART Data Dictionary (bsDD)

The buildingSMART Data Dictionary (bsDDISO 12006-3) is an open, shared object-oriented database where the terminology about the BIM objects is defined. It is considered as a library where terms and associated meanings are described (Bell el al. 2008). There are two types of information within this dictionary. Firstly, the naming of classes are defined in the different languages so that they are can be understood by people from different nationalities. Hereby each term is given a unique number: A Globally Unique Identifier (GUID). This number makes it possible for anyone to identify objects that are named in a foreign language. Secondly, characteristics are assigned to each concept. These characteristics can for example describe the length and width of the object or its function (BuildingSMART, 2009).

5.7 Limitations of non- geometrical information exchange

As already have been mentioned within chapter 1, a central BIM approach has been criticized within academic as well as in corporate domains. Even by using the MVD’s as described in section 5.4 to avoid an huge and complex information model, issues in respect to especially non- geometric data exchange occur. Namely, the BIM approach prescribes to share and reuse such data sets by applying the COBie methodology. However, the semi- structured formats in which the COBie data is exchanged (in STEP and common spreadsheet templates) raises issues related to semantic heterogeneity and interoperability (Kapourani et al., 2015).

In the first case, semantic heterogeneity occurs whenever there is more than one way to structure an information model. Thereby Beetz especially stresses the lack of formalism of the EXPRESS modeling languages. In the second case, interoperability issues exist partly because of the complex nature of the EXPRESS modeling language. In example, outside of the few engineering domains that use EXPRESS, the popularity among developers, the use of this particular family of modeling languages and the existence of (affordable or free) tools is very limited (Beetz, 2009). In regard to tabular data, spreadsheet were meant in the first place a way for read and manipulate data by humans easily and not a way to integrate data structures with (between machines). Furthermore, it appears that both data structures do not provide the means to provide meaningful answers to sophisticated queries (Kapourani et al., 2015).

A last promising approach in the AEC domain to tackle above mentioned problems is called Linked Data which is initiated by the W3C. The fundamental concept of Linked Data is that data is expressed according to an open information model in a formal way with the mindset that it will be shared and reused by other information systems (in different domains). It is based on so called Semantic Web technologies for representing, sharing, and querying structural data on the Web (Curry et al., 2013). In general, it uses the Web Ontology Language (OWL), Resource Description Framework (RDF), and Uniform Resource Identifiers which contrasts with respectively EXPRESS, STEP Physical File and GUIDs (Törmä, 2013). Therefore, in the next chapter an elaborated view of the Semantic Web and Linked Data will be provided.

6. Linked Data

As is found in chapter 5, it appears there is a need to integrate data in a semantic way which are fundamentally different produced and used. Therefore, the purpose of this part is to explore the field of the Semantic Web which was initiated by the W3C as a means of a solution. Since this technology is still in its infancy within the AEC domain, a brief examination of its origin will be provided. Secondly, the Semantic Web and its core concept called Linked Data will be defined concisely. Finally, an elaborated explanation will be given about how Linked Data could be generated, be linked and be used within the context of this research.

6.1 The World Wide Web (WWW)

Currently, the World Wide Web (the Internet) is made up of servers and clients. Clients have access to information that the servers provide via Hypertext Transfer Protocol (HTTP). In most cases the information is stored as web pages written in the HyperText Markup Language (HTML) language. The most important feature of HTML documents is that they contain links which form the basis of the complex structures of references. The actual meaning of the information it carries is normally provided by associating meta- data to it. Generally speaking, metadata can be seen as pieces of information about other data (Szeredi, Lukácsy, & Benko, 2014).

To support computer processing, meta-information is usually stored in the (Extensible Markup Language) XML language. An XML document is a text file designed to be capable of storing and exchange data in a structured form that conform to a specific syntax. However, when it comes to semantic interoperability, XML has disadvantages. Namely, the current Web revolves around the fact that anyone can create any type of content about any topic (this feature of the Web is also called the AAA- slogan) and finally publish it to the Web infrastructure. XML only aims at the syntactic structure of documents and does not enforce any common interpretation. Therefore, XML is not suitable for the long run for supporting information exchange between internet- related applications (i.e. a client and server) (Decker et al., 2000).

6.2 The Semantic Web and Linked Data

The vision of the Semantic Web is to extend the principles of the Web from documents to data allowing to create a web of open interlinked data sets which are created independently from each other. It can therefore be considered as an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation. In this way the Semantic Web provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries.

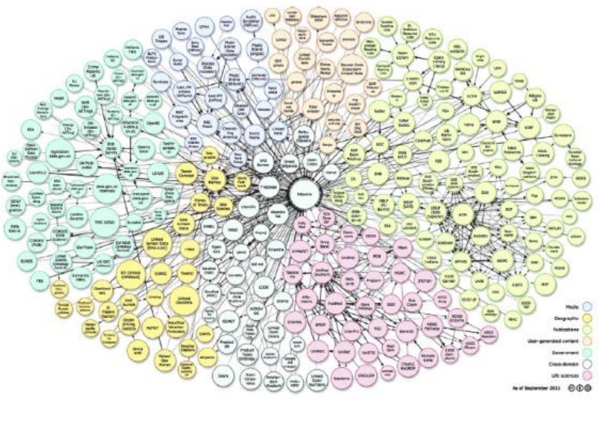


Fig. 6A A visualization of the Linked Open Data cloud on the Semantic Web (Curry et al., 2013)

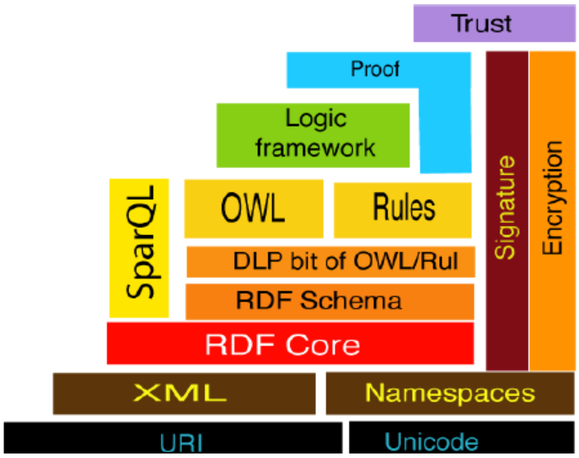


Fig. 6B The Semantic Web stack (Pauwels, 2011)

The basic idea of the Semantic Web is to enable semantics by associating the published web contents with meta-information in a standardized form, making it possible to link data sets together. See figure 6A. This metadata is based upon an abstract data model called Resource Description Framework (RDF) which can be extended by various other web languages. This mix of technologies forms the so called Semantic Web stack. See figure 6B. Standardized languages enable links to be set between terms in different data sources and therefore connect these sources. This is the essence of Linked Data. Basically, the Linked Data term refers to a set of best practices for publishing and interlinking structured data on the Web which are called Linked Data principles. These principles are the following (Heath & Bizer, 2011):

1. Use URIs as an identifier for things.
   A Uniform Resource Identifier (URI) is a string of characters used to identify a thing in the real world (also referred to as resource or concept). Therefore, such a URI has basically the same function as a GUID in an IFC- SPF model. Examples could be very concrete like a building or a wall but can represent also abstract concepts like a relation or a group of rooms. The first Linked Data principle advocates the use of URI references to uniquely identify resources in a data set.

2. Use URIs according the HTTP standard
   HTTP URIs enables the URI to be globally unique together with a simple, well-understood retrieval mechanism. In this way people over the whole world are able to use URIs to identify things which then can be dereferenced (to access the concept to which the URI points) over the HTTP protocol into a description of the identified object or concept. The way to construct HTTP URIs is described extensively by best practices called “Cool URIs” by the W3C.

3. Provide useful RDF metadata
   The third Linked Data principle advocates the use of the standardized languages from the on the Semantic Web Stack for publishing structured data (i.e. RDF).

4. Include links to other data sets
   The fourth and most essential Linked Data principle promotes the use of hyperlinks (outgoing links) to connect concepts in other data sets.

When publishing Linked Data on the Web, data is represented using the generic RDF data model. However, RDF does not provide any domain-specific terms for describing formal hierarchies of things in the world and how they relate to each other. This function is served by taxonomies, vocabularies and ontologies expressed in SKOS (Simple Knowledge Organization System) RDFS (the RDF Vocabulary Description Language, also known as RDF Schema) and OWL (the Web Ontology Language). In a Linked Data context, it is often sufficient to express vocabularies in RDFS and certain primitives from OWL. So, besides RDF only RDFS and OWL will be elaborated in the following paragraphs.

6.3. Semantic Web standards

6.3.1 RDF

Essentially, the purpose of RDF is to connect URI-identified real world entities with other resources or just with plain literals (attributes of resource) by using properties. This allows RDF statements to act as so called triples which consist of a subject, a predicate and an object. Hereby, the subject denotes the resource, and the predicate (also called as property) denotes attributes of the resource and represents a relationship between the subject and the object. In this way, multiple triples together form a graph. Thus, such a RDF data model generally resembles the function of the STEP Physical File in an IFC- SPF model.

Figure 6C illustrates a graph clearly via two triples. Considering, the prefixes the subject (a real world entity) is uniquely identified as http://example.com#SpaceX which overall height (http://example.com#hasHeight) is represented by the literal “2180” (note that the unit is not provided). Furthermore, the same subject has an opening element (http://example.com#isBoundedBy) that has the unique URI http://example.com#WallY.

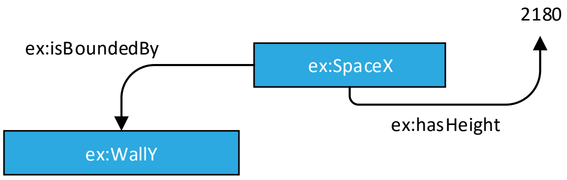


Fig 6C. A graph of two triples representing a window with a certain height and an opening element (Prefix: ex: http://example.com#).

Such a graph can be considered as a RDF data set. The meaning of such a description is that the statements it contains are true. They make it possible to formulate assertions unambiguously and to combine fragments of information coming from different sources. The RDF graph of fig. X can be represented using various syntaxes. The most commonly used syntaxes are RDF/XML (.RDF), Turtle (.TTL), Notation-3 (.N3) and N-Triples (.NT) (Heath & Bizer, 2011). The most often used syntax for machine- interpretable graphs is RDF/XML:

```
<rdf:Description rdf:about="http://example.com/SpaceX">
        <ex:hasHeight rdf:datatype="http://www.w3.org/2001/XMLSchema#double">2180</ex:hasHeight>
        <ex:isBoundedBy rdf:resource="http://example.com#WallY"
</rdf:Description>
```

However, throughout this report the Turtle- notation will be used, because it is considered to be the human- readable syntax for humans:

```
        ex:SpaceX   ex:isBoundedBy  ex:WallY.
        ex:SpaceX   ex:hasHeight    “2180.00”^^xsd:double .
```

As already stated, RDF graphs can be improved semantically by using RDF vocabularies or domain specific ontologies. The most basic elements to describe such an ontology are available in the RDF Schema (RDFS) vocabulary (Pauwels & Terkaj, 2016).

6.3.2 RDFS

RDFS is the schema language for RDF. The semantics of this language is expressed through the mechanism of inferencing: The meaning of any construct in RDFS is given trough a making logical conclusions by computers about the structured RDF statements. For example, it enables to define ambiguously the RDF- metadata (i.e. via `rdfs:Class`) and relates these to each other (i.e. via `rdfs:subClassOf`).

For instance, figure 6D shows how the class instances are linked to their definition via `rdf:type`. These classes are on their turn are asserted as an subclass of the class `rdfs:BuildingElement` via `rdfs:subClassOf`. From this illustration it can be found that both `ex:Wall` and `ex:SpaceX` reside on the same level in the class- hierarchy. According to the RDFS rules it possible to infer that the instances WallY and SpaceX belong to the concept `ex:BuildingElement`. So, the graphs hold actually 2 building elements.

In the same way, it is possible to structure the properties (via `rdfs:Property`) and the relationships between them (`rdfs:subPropertyOf`). Other key constructs to structure properties are `rdfs:domain` and `rdfs:range` which respectively declares the subject and object in a triple.

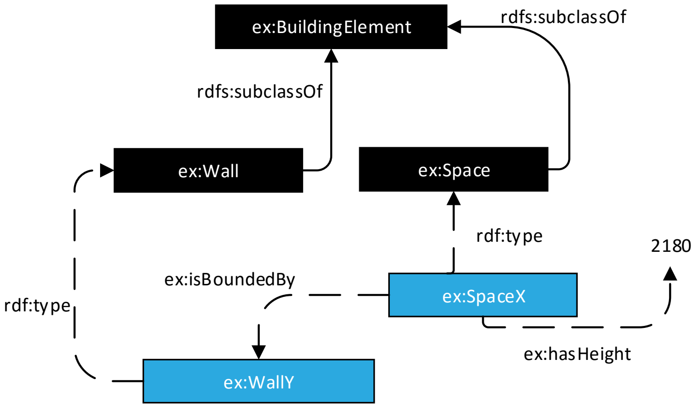


Fig. 6D A graph of 6 triples representing a window with a certain height and an opening element (prefix: ex: `http://example.com#`).

At first sight, these rules may seem simple, but by using combination of asserted and inferred statements it is possible to reproduce complex real world situations (Allemang & Hendler, 2011).

6.3.3 OWL

More expressive elements to describe data can be achieved by using OWL- ontologies. In short, OWL further enhances the RDFS concepts to allow making more complex RDF statements, such as cardinality restrictions, type restrictions, complex class expressions (Pauwels & Terkaj, 2016). An example is the `owl:sameAs` construct to merge data from multiple sources, which is used extensively in the context of Linked Data. By using `owl:sameAs` it is possible that to state that different resources actually represent the same real world entity. When resources are determined to be the same, information about them in different sets sources can be merged. Such a construct in turtle- syntax is shown below:

```
ex:SpaceX owl:sameAs ex:SpaceZ
```

An ontology is defined as a formal explicit description of formal naming and definition of the types, properties, and interrelationships of the entities that really or fundamentally exist for a particular domain of discourse. Ontologies are to improve data integration when ambiguities could exist on terms used in data sets or when extra knowledge can lead to the discovery of new relationships. Ontologies can capture the semantics of data, describing the knowledge for sharing in a specific domain and provide reasoning capabilities (Koukias et al., 2013b). For this reason, the purpose of an ontology is comparable to the EXPRESS functionalities within an IFC- SPF file.

There are basically 2 types of ontologies: A domain ontology that provides a unambiguous description of specific things from a certain point of view and an upper ontology that describes general things in a formal way. A concrete example of a domain ontology is the IfcOWL ontology while Dublin Core functions as an upper ontology.

The full vocabulary of OWL uses URIs in the RDF, RDFS, and OWL namespaces, and it also makes use of the XML Schema literal definitions. (Segaran, Evans, & Taylor, 2009).

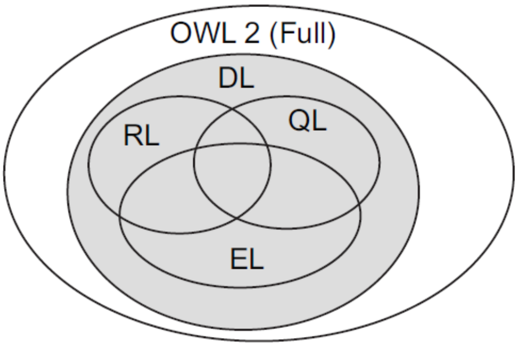


Fig. 6E The 4 OWL- profiles of OWL2.

However, it was observed that certain technologies could only process certain subsets of OWL conveniently. Therefore, OWL2 (the most recent version of OWL) is divided into four so-called profiles, namely OWL2 DL, OWL2 EL, OWL2 QL and OWL2 RL. As outlined in Motik et al. [19], an OWL2 profile “is a trimmed down version of OWL2 that trades some expressive power for the efficiency of reasoning”. In short, in each of the givenOWL2 profiles, a number of statements that can be used in OWL2 DL are not allowed. By not allowing these statements, and thus sacrificing some expressiveness, important improvements can be made in terms of performance. Fig. 2 displays the relationships between these three key profiles. Thereby it can been seen that DL is the largest subset of OWL2 and is the super set of the other three profiles.

6.3.4 SPARQL

RDF graphs are usually stored in so called triplestore (also known as a knowledge base) which can be seen as a database for the storage and retrieval of triples via semantic queries. The SPARQL Protocol And RDF Query Language (SPARQL) provides a means for performing such queries. The SPARQL query language relates closely with the RDF structure (subject, predicate, object) itself. Namely, the key element of a SPARQL query is the graph pattern. This pattern is a smaller graph including both resources and (unknown) variables that specifies what information needs to be retrieved from the RDF graph. SPARQL query patterns are produced as a variant of Turtle (DuCharme, 2013). SPARQL provides four forms of queries: SELECT, CONSTRUCT, ASK, and DESCRIBE. All of these attempt to find solutions to a graph pattern, and all share similar constructs. (Segaran et al., 2009). An example of the commonly used SELECT query (to the graph pattern shown in Fig. X) is given below:

```
SELECT ?Variable
      WHERE {
              ?Variable rdf:type ex:Space.
            }
```

Hereby `SELECT?Variable` represents the desired piece of data to be retrieved from the graph. The `WHERE` clause specifies the graph pattern which defines `?Variable` as a subject of a type `ex:Space`. Since there is exactly one subject within the graph that conforms to this pattern, only the instance `ex:SpaceX` wil be the result of this query. As already mentioned in section 5.6, EXPRESS technologies do no provide a query language by themselves (Törmä, 2013).

7. Project Analysis

The purpose of this part is manifold. Namely, during this chapter the context of Design Build Finance Maintain Operate (DBFMO) projects will be elaborated from the perspective of Facility Management (FM) in making sure that the operation of the facility complies with the specified SE requirements. Hereby the Business Process Modeling Notation (BPMN) is used to model the process and determine the main objective(s) of the Linked Data tool. Based upon this process model user requirements will be determined and prioritized in an immediate way to set up an agile workflow to develop the Linked Data tool. Thirdly, the architecture of the actual Linked Data tool is described concisely based upon these user requirements by using the Unified Modeling Language (UML). Finally, a real world use case is analyzed of which sensor data sets are obtained (or created), summarized and visualized. The end result will act as a backbone in a way that the future development of the Linked Data tool is able to actually improve the described monitoring process. During this stage the following technologies were used: Revit 2015 (together with the IFC for Revit V17.1.0 export extension) and Python 2.7 using the Pandas 0.18.1 library.

7.1 Process analysis

Currently, the Dutch Central Government Real Estate Agency (CGREA) provides a contractual template for DBFMO- projects that prescribes in general the operations during the operations & maintenance phase. Naturally, only the process of verifying compliance to the output specifications (OS) will be examined: Other related aspects (i.e. revision management when output specifications have to be changed) are left out. This process could then be described concisely via the following three sub-processes.

Firstly, the contractor has to provide the client a plan which describes which monitoring activities the contractor will perform. The contents of such a plan could be described as follows:

1. The contractor is obliged to develop a monitoring system. One of these functions is that the monitoring system should register warnings by the contractor. These warnings should at least include a description of the warning together with (the ID of the) registrar, the date and time, the room, (if possible) its cause, an ID registration number and (if applicable) any particularities. If the warning is dispatched, the system should able the contractor at least to register this together with the cause of the warning, the time of recovery, (if applicable) the obstacles that troubled the dispatching and (if applicable) any particularities.
2. The contractor must enable an impartial third party to perform measurements periodically. Both the client and the contractor are able to verify the measurements by this third party.
3. The client is able to measure incidentally into what extent the contractor complies to the agreed output specifications via a(nother) third party (Rijksvastgoeddienst, 2014).

In order to elaborate and validate whether or not this BPMN- model corresponded to the actual process four expert interviews were conducted. The outcomes can be found within the appendices A, B, D, E and F. Based upon the outcomes, it appears that the attained description by each interviewee corresponds closely with the template of the CGREA. They only differ in cases of the instantiation of parties (contractor and client). For example, the facility management as described in Appendix F is performed by the main contractor that built the project while in case D FM was conducted by an actual facility management firm. Therefore it can be concluded that the process analysis is valid. The final BPMN schema is depicted below as well in Appendix I.

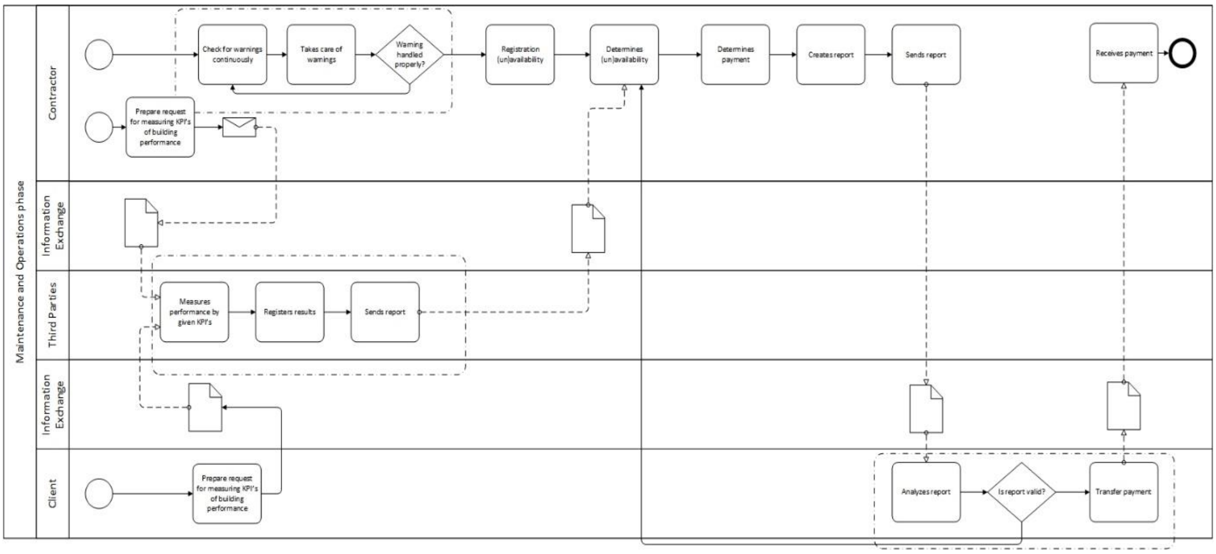

Maintenance and Operations phase
Contractor
Information Exchange
Third Parties
Information Exchange
Client
Check for warnings continuously
Takes care of warnings
Warning handled properly?
Registration (un)availability
Determines (un)availability
Determines payment
Creates report
Sends report
Receives payment
Prepare request for measuring KPI's of building performance
Measures performance by given KPI's
Registers results
Sends report
Prepare request for measuring KPI's of building performance
Analyzes report
Is report valid?
Transfer payment

Fig. 7A The designed business process (in BPMN notation) for monitoring a building.

The process analysis shows explicitly where the convergence of the produced data by a facility (during the operations & maintenance phase) takes place and how it is able to define the final outcome of a DBFMO- monitoring process. Another finding during this stage was that all interviewees stated (synchronous to the literature) that the data which indicate malfunctions is mainly captured by various sensors. See for all interviews the appendices A, B, E, F and G. Furthermore, all FM parties used building related data in some form to mark the location of the sensor(observations). Though, the convergence of this heterogeneous (raw) monitoring data to meaningful and integrated information seems to be essential, all of the interviewees state that Facility Management lack in this aspect. Therefore, the added value of a Linked Data application should lie in linking data sets (output specifications, sensor monitoring data and building data) that allow verifications of systems in order to improve FM activities and thereby securing (1) building performance according to the SE requirements to the owner (and other users) and (2) financial payments to the contractor.

7.2 Requirement analysis

Based upon these 2 identified objectives it is possible to indicate the main functional requirements for the desired tool functionality. These requirements have been prioritized directly by using the MoSCoW- method. This technique enables an agile approach in respect to the prototyping process and thereby keeping the focus throughout the project on delivering the identified business benefits. The so called "must haves" (M) stands for the obligatory requirements which must be incorporated in the end result while the "should haves" (S) are strongly desired. The end result will consist of the "could haves" (C) only if the process allows to do so. The "won't haves" (W) will explicitly not be incorporated in the Linked Data tool (DSDM Consortium, 2008). See figure 7B for the prioritized list of requirements.

	MUST
1	Link an output specifications data set and a sensor data set to an IFC- SPF data model
2	Make use of generic best practices (i.e. Cool URIs, Ontology Engineering 101)
3	Verify systems by checking whether a sensor value differs from a OS value
	SHOULD
4	Visualize the most recent (un)availability in an IFC according to specific time intervals

	COULD
6	Register (un)availability of systems in real time
	WON'T
7	Distinguish between different forms of availability
8	The ability to incorporate registered warnings and reparations by actors
9	Visualize results according to the standards of the contractual template for DBFMO-projects by the CGREA (like with associated ID number, a cause etc.)

Fig. 7B A prioritized list of requirements about what functions the Linked Data tool should comprise of.

7.3 Use Case

In order to capture the interaction of the tool and the user (the FM party) within the previously described process, a use case model was developed. Via this model it is possible to depict the core function of the desired architecture. See figure 7C.

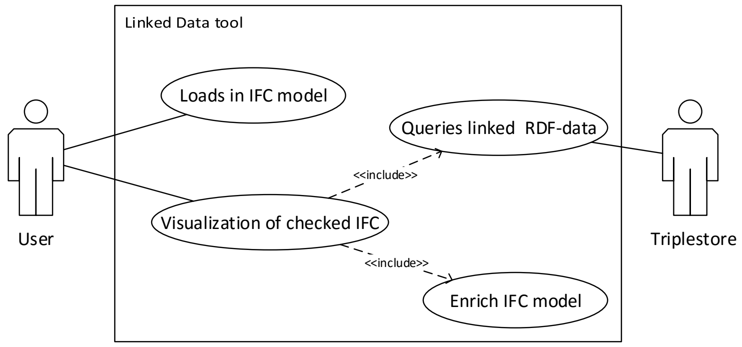


Fig. 7C A use case model of a semantic tool that is able
to a visualize a query result via an IFC- SPF model.

In accordance to the BPMN- schema the use case model consist of an authorized actor (i.e. the contractor) which queries the Linked Data sets in the triplestore. Thereby the main use case *Visualization of checked IFC* illustrates the task *Check for the warning continuously* which is depicted in the BPMN- model. However in order to let the system perform this essential task, three other use cases has to be processed first. Firstly, the correct IFC- model should be imported. Then when the user wants to verify the current availability of the checked building model, the tool must retrieve the (already checked) results and enrich the IFC- file.

7.4 Application architecture

Based upon the use case model a visualization of the logical static structure of the desired tool is made. In this case, a tool (or application) should be seen as a work that is able to process or display data using programming code (from a triplestore). From the prioritized requirements and the use case model it can be derived that the desired application consist of different levels (i.e. a triplestore, an interface) and therefore should have a so called multilayered architecture: Each individual layer consist of multiple tasks that perform a together a coherent process. In general, such systems could comprise the following layers:

1. Presentation: Deals with the interactions between a user and tool functionalities
2. Process: Provides the process logic of the tool

3. Business: Provides specific business logic for the application
4. Data: Deals with the interaction between the application and the data base
5. Utilities: Provides support for all other layers within the architecture
6. Business Component: Provides general business logic for the application

Within a class diagram these layers can be captured in packages while the tasks correspond to the actual classes (Hoogendoorn, 2004). For this research, only layers “1. Presentation”, “2. Process”, “4. Data” and “5. Utilities” have to be incorporated. This means the class diagram can be visualized as figure 7D.

Basically, this class diagram elaborates the use case model in such a way the actual tool can be programmed. Due to a lack of UML notations for semantic applications, it is chosen to model the Linked Data sets in classes according to relational database principles: In order to do so, appropriate stereotypes have been designed.

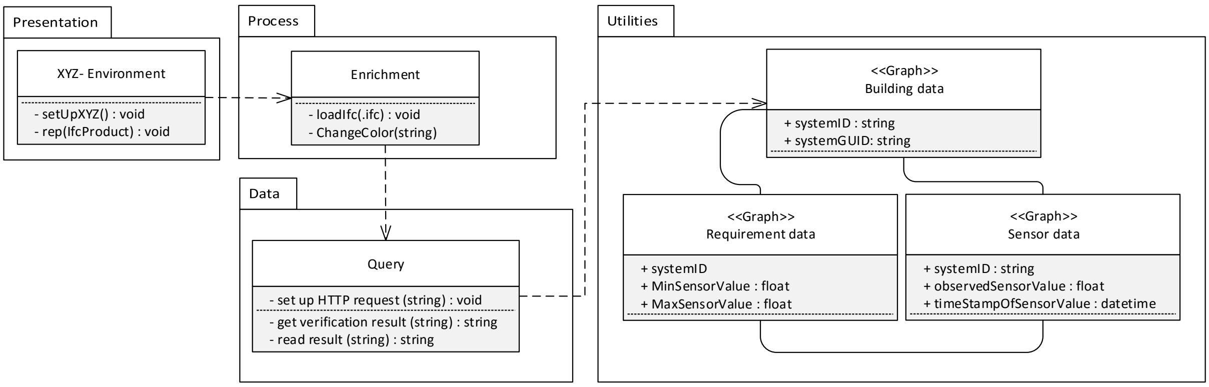


Fig. 7D The class diagram of the Linked Data tool that is able to a visualize a query result via an IFC- SPF model.

7.5 Descriptive data analysis

In order to script the tool functionalities a DBFMO- use case had to be obtained from which the actual data sets could be acquired. This use case is represented by the National Military Museum at Soest in The Netherlands and is currently maintained by the FM department of the contractor Heijmans.

After the data acquisition a brief data analysis is conducted that consist of two parts. Firstly, a general overview of the main findings is provided whereafter each data set is analyzed more thoroughly.

7.5.1 General data analysis

It seemed that each set was produced by a different actor in another stage of the building lifecycle with its own perspective. It also appeared that they were also produced by applications of different vendors and therefore consist of different information structures (CAD, PDF, Excel). These characteristics are briefly outlined in figure 7E.

	Output specifications (OS)	Building model information	Sensor data (2014-2016)
Actor	Central Government Real Estate Agency	Heijmans- construction division	Heijmans- facility division
Perspective	Develop OS	Create a model according to OS	Maintain building according to OS
Phase	Planning phase	Design & Construction	Operations & Maintenance
Tool	MS Word	Autocad	Facilicom
Structure	Natural language as plain text descriptions	2D – Geometry in CAD	Data points in tabular format
Format	.pdf	.dwg	.xlsx (in MS Acces)

Fig. 7E The class diagram of a semantic tool that is able to a visualize a query result via a IFC- SPF model.

Another important finding was that each data set dealt with building rooms (or spaces) though labeled these real world entities differently. Namely, a building model describes a room which serves as a installed location for the sensors (in the sensor data set) and is specified by the requirement in the OS data set. It must be noted, that the sensor data set did not contain room names at first and could only be added to Excel- file after an interview with Facility Management. Since it was necessary to determine which requirement and sensor values corresponded to which room of the building model it was necessary to map the room names between the sets. See figure 7F for the end result.

Output specificaties data set	Building data set	Sensor data set
Rooms	Rooms	Rooms
Intro-experience	Krijgsmachtbrede Themaruimte Introductie	Krijgsmachtbrede themaruimte introductie
Hoofdthema 1 nederland en de wereld	Nederland	Krijgsmachtbrede themaruimte 1
Hoofdthema 2 de wereld van de krijgsmacht	De Krijgsmacht	Krijgsmachtbrede themaruimte 2
Pronkzaal	Schatkamer	Krijgsmachtbrede themaruimte pronkzaal
Hoofdthema 3 militairen in de schijnwerpers	Militairen	Krijgsmachtbrede themaruimte 3
Hoofdthema 4a de wereld van de techniek	De Toekomst	Krijgsmachtbrede themaruimte 4
Hoofdthema 5 operaties	Oorlog	Krijgsmachtbrede themaruimte 5
Hoofdthema 6 samenleving en krijgsmacht	Samenleving en Krijgsmacht	Krijgsmachtbrede themaruimte 6

Fig. 7F The different room labels that point to the same exposition room entitiy. For example, “Intro-Experience”, “Krijgsmachtbrede Themaruimte Introductie” en “Krijgsmachtbrede themaruimte introductie” can be considered as 3 different ways to express the same exposition room in the real world.

7.5.2 Data set analysis

In regard to the OS data set, it was chosen to find a room specification that could be verified by temperature sensor data for demonstration purposes. Therefore, the requirement “Operatieve Temperatuur ” was selected. This requirement stated that a temperature sensor value of a certain room should always be between a specified lower boundary and a specified upper boundary. In order to enhance the focus of this research, only the 8 exposition rooms on the second level were taken into account.

It was essential that the building model could be converted to an IFC information model (and get an adequate COBie MVD). However, the provided model appeared to be set up in a DWG- format which meant that this was not possible. As a solution a BIM- model was created whereby only essential parts of the building (the second floor and facade) were modeled on a schematic level and finally exported as an IFC-SPF file (using an adequate mapping template).

An important requirement towards the 8 sensor data sets were the values and associated timestamps. Even though these values were present, the sensor measurement values were wrongly depicted due to the fact that the commas were missing: Therefore these values were corrected. The data was sliced between 2015 and 2016 and each summarized in statistical terms. This procedure and associated (visualization) results are shown in Descriptive analysis of sensor temperature data. From the results it appears that the indoor room temperature of each exposition room is stable throughout the whole year (with a mean of approximately 19,9°C) and does not exceed the imposed boundaries. Therefore, dummy variables were introduced to have a more varied dataset.

8. Ontology Engineering

The purpose of this part is to provide an thorough explanation of how the 3 data sets as described in section 7.5.2 could be expressed in a formal way by OWL domain ontologies and be interlinked with each other according to the generic Ontology Engineering 101 best practices. During this phase the following tool was used: Protegé 4.0.

The scope of such a combined ontology is to enrich and define the SE requirements and FM sensor data in an explicit way, which allows visualizing which rooms in the associated BIM model do not comply. With this aim in mind, each subsequent paragraph covers the development of an ontology of 1 data set type. Therefore, each chapter starts with a concise exploration to find ontologies for reuse. The selection of ontology (elements) was based upon the following criteria (Radulovic et al., 2015):

1. The semantics of the class or property in the ontology relates to the term;
2. If the term relates to a class, the class in the ontology should have as many properties that correlate to the term as possible;
3. The ontology that describes the class or property related to the search term is widely accepted and used.

If no ontology conforms to above criteria an own ontology is developed via the following steps: (1) definition of the classes and the hierarchy, (2) definition of slots (properties) and (3) definition of facets (Noy & McGuinness, 2001).

8.1 Sensor data

The most common way to express sensor related concepts formally is through the Semantic Sensor Network (SSN) ontology[2] conducted by a W3C incubator group. This ontology originates from 12 other ontologies like CSIRO Sensor Ontology, OnToSensor and SWAMO and several vocabularies (i.e. SensorML). Furthermore, this ontology is structured following an ontology design pattern called Stimulus-Sensor-Observation (SSO) while at the same time being aligned to the generic DOLCE Ultra-Light (DUL) upper ontology (M Compton, Henson, Neuhaus, Lefort, & A, 2009) [3]. Hereby an ontology design pattern can be seen as a flexible abstract (light-weight) template which refers to best practices for creating an actual (heavy-weight) ontology. Since this ontology will be used to structure the sensor data set, a concise explanation of SSO, SSN and DUL is provided below.

SSO consist of a minimal set of classes and relations centered around the notions of “Stimuli”, “Sensor”, and “Observation”. The term “Stimuli” Is hereby considered as the only link to the physical environment (Semantic Sensor Network Incubator Group, n.d.). However, the SSO ontology cannot be applied directly because it does not specify the actual meaning of its classes. For example, the SSO does not provide clarity to the question if the term “Observation” should be interpreted as database records or real observations.

So in order to enhance the interpretation of the abstract terms, the SSO pattern has been aligned to the DUL ontology. This upper ontology defines (widely accepted) general concepts (`dul:SocialObject`) that can be mapped to concepts of domain specific ontologies

[2] SSN URI : http://www.w3.org/2005/Incubator/ssn/ssnx/ssn.
SSN Namespace : http://purl.oclc.org/NET/ssnx/ssn#.

[3] DUL URI : http://www.ontologydesignpatterns.org/ont/dul/DUL.owl.
DUL Namespace : http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#.

(`sso:Observation`). By structuring additional classes along the aligned pattern the SSN domain ontology is able to be a framework to actually describe sensors, observations and related concepts (Janowicz & Compton, 2010) . This outcome of this process is illustrated by figure 8A.

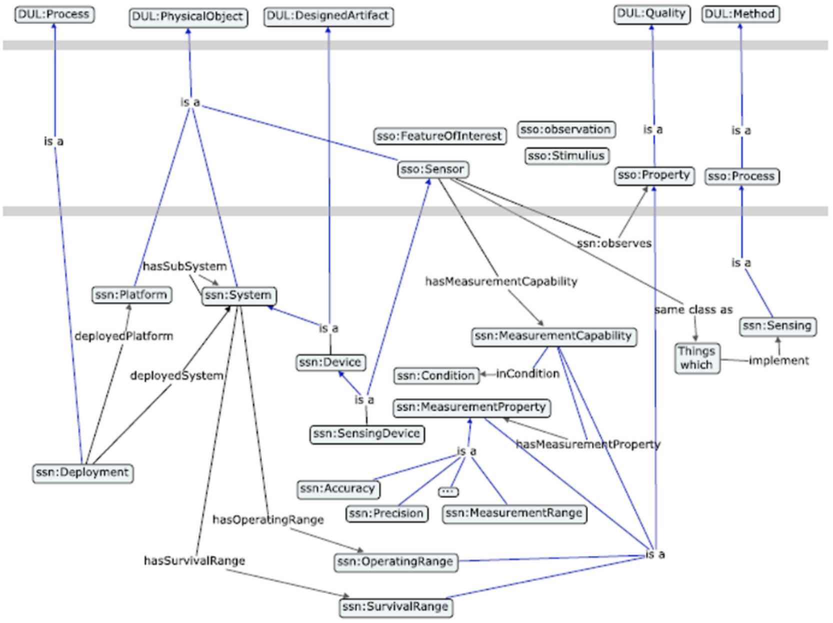


Fig. 8A A partial view on the integration of the DUL-aligned SSO ontology design pattern with the Semantic Sensor Network ontology (Janowicz & Compton, 2010)

The SSN ontology is organized, conceptually into ten modules. See figure 8B. The full ontology consists of 41 concepts and 39 object properties, directly inheriting from 11 DUL concepts and 14 DUL object properties. It deliberately does not describe concepts from other domains, such as time, measurement values and locations: These concepts have to be included from other domain ontologies (Semantic Sensor Network Incubator Group, n.d.).

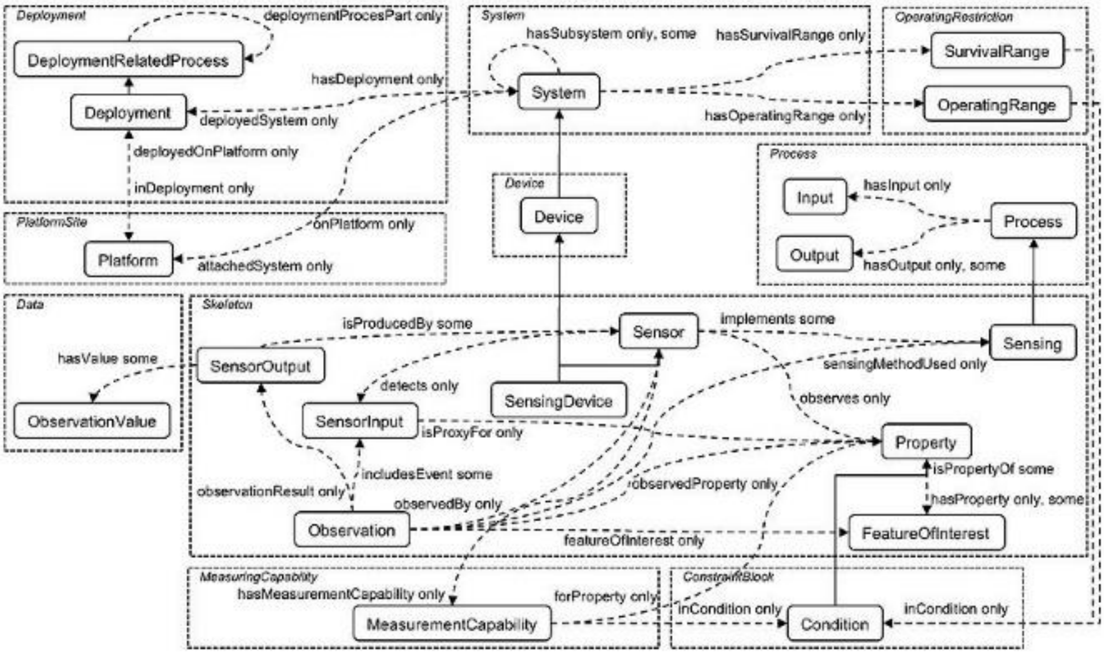


Fig. 8B A modular overview of the Semantic Sensor Network ontology classes and properties (Semantic Sensor Network Incubator Group, n.d.).

The SSN enables the description of sensor from 4 perspectives (Michael Compton et al., 2012):

1. A sensor perspective, with a focus on what (part of the sensor) senses, how it senses, and what is sensed;
2. An observation perspective, with a focus on observation data and related metadata;
3. A system perspective, with a focus on systems of sensors and deployments; and,
4. A feature and property perspective, focusing on what senses a particular property or what observations have been made about a property.

Considering the scope for this ontology, the SSN (data and skeleton) modules that represent the observation perspective have been used.

SSN allows to complement sensor's capabilities for time via the object properties `ssn:observationResultTime` (represents the time when the result became available) and `ssn:observationSamplingTime` (a time at which the sampling took place) (Michael Compton et al., 2012). Since the timestamp values of the sensor data set represent the actual sampling time, `ssn:observationSamplingTime` is selected to link the SSN ontology to a time related ontology that complies to the selection criteria. In this case, the widely accepted Time domain ontology[4] by the W3C appeared to be sufficient. It is now possible to add the following classes and properties to express the timestamp values of the sensor data in a formal way by incorporating the XSD vocabulary[5]:

```
ssn:Observation ssn:observationSamplingTime time:Instant .
time:Instant    time:inXSDTime              xsd:DateTime .
```

This addition to the SSN ontology is depicted by A in figure 8C.

The SSN ontology allows locations to be represented as either abstractions of real-world locations or as absolute or relative locations. The first case is possible by relating a sensor to a place (dul:PhysicalPlace) via the property dul:hasLocation. The other approach is possible by relating the sensor to observable aspects (i.e. the relative latititude and longitude) via ssn:hasProperty (Pfisterer et al., 2011). Only the first option allows interlinking with the IFC data set through:

```
ssn:Sensor dul:hasLocation dul:PhysicalPlace .
```

This addition to the SSN ontology is depicted by B in figure 8C.

SSN does not define how the temperature values should be expressed. Fortunately, literature explicitly shows interlinking with the Quantities, Units, Dimensions and Data Type (QUDT) ontology[6] is possible by inferring a ssn:ObservationValue instance as an type of qudt:Quantity via class subsumption (Bou-ghannam, 2013). However, since the data set is relatively small it is opted to simply assert this rdf:type- relationship (Kolchin et al., 2015). For example:

```
ssn:id/ObservationValue/72a3264f-1d9b-11e6-b5be-240a64020db4
      rdf:type ssn:ObservationValue ,
```

[4] Time URI : http://www.w3.org/2006/time.
Time Namespace : http://www.w3.org/2006/time#.

[5] XSD URI : http://www.w3.org/2001/XMLSchema.
XSD Namespace : http://www.w3.org/2001/XMLSchema#.

[6] QUDT URI : http://qudt.org/schema/qudt.
QUDT Namespace : http://qudt.org/schema/qudt#.

```
          rdf:type qudt:Quantity .
```

This allows SSN to express the data values of the room temperature (via the XSD vocabulary) as follows:

```
    ssn:ObervationValue  qudt:numericValue  xsd:Float
    ssn:ObservationValue qudt:unit          qudt:Unit
```

This addition to the SSN ontology is depicted by C in figure 8C.

In conclusion, the resulting graph consist of the following 5 ontologies that describe each another aspect: DUL, SSN, TIME, XSD and QUDT. See figure 8C for a visualization of the final result.

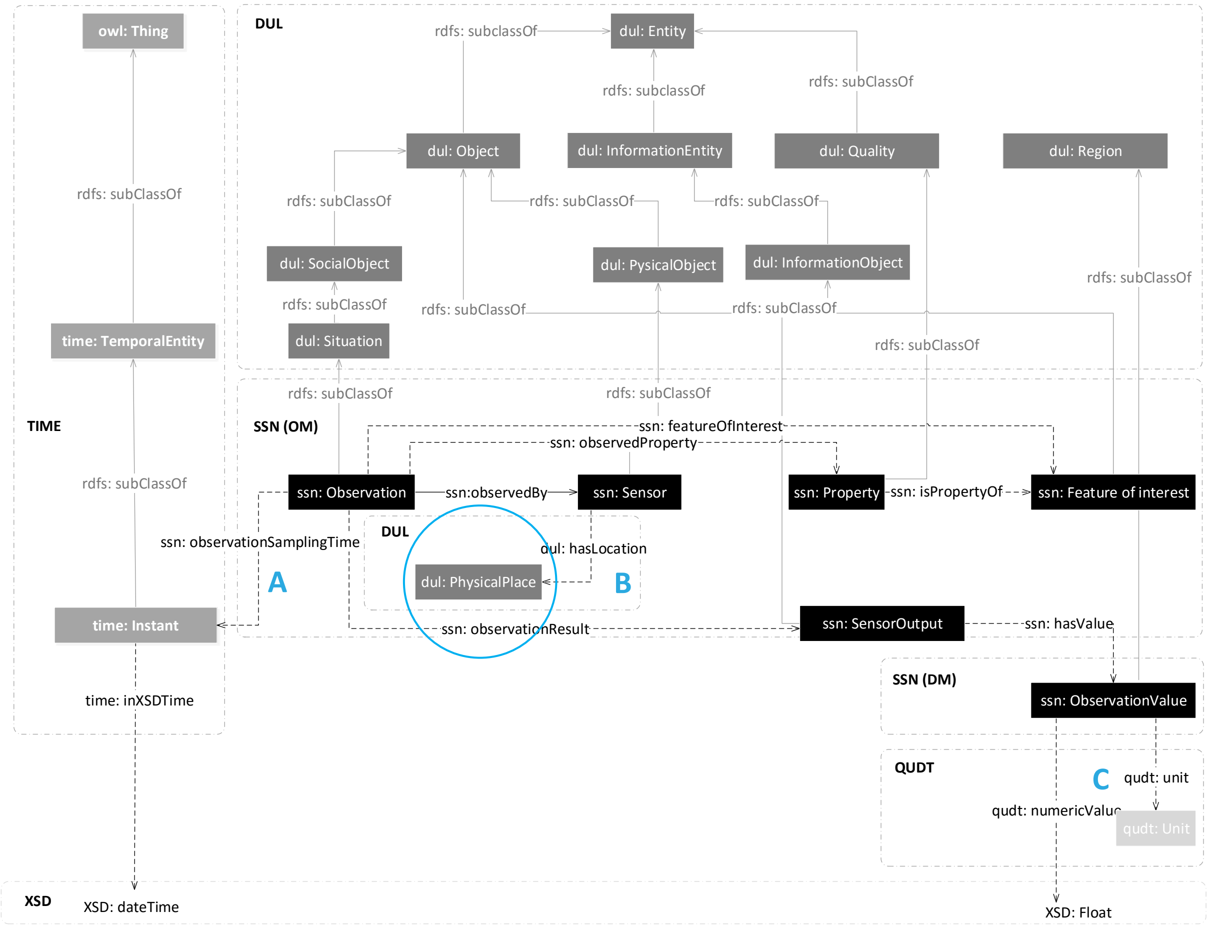


Fig. 8C An overview of the established ontology to express the sensor data set ambiguously. 4 Shades of grey are used to distinguish the different vocabularies. The colours distinguish the used ontologies whereby each is labeled (in bold) and demarcated by dotted lines. The blue circle points to a concept as a linking possibility to the requirement- and building data set.

8.2 IFC- SPF data

In respect to uplifting the IFC data into an ontological level a number of EXPRESS to OWL conversion procedures has been proposed so far. This so called IfcOWL provides a Web Ontology Language (OWL) representation of the Industry Foundation Classes (IFC) schema. The ifcOWL ontology has thereby the same status as the EXPRESS and XSD schemas of IFC. Since the current ifcOWL relies on findings of previous versions, the most relevant findings concerning the ifcOWL are enumerated below in chronical order.

One of the earlier methods was proposed by Schevers and Drogemuller by examing a unidirectional mapping procedure from EXPRESS to OWL for research purposes. The resulting prototype did not map all the IFC data to OWL though encouraged the exploration for a more adequate conversion (Schevers & Drogemuller, 2006). Consequently, Beetz et al. proposed a semi-automatic method for converting EXPRESS schemas to OWL ontologies. Thereby this research explained through use cases how this enhanced information model could tackle several problems in the AEC domain (Beetz, van Leeuwen, & de Vries, 2009). Another relevant research project is the OnToSTEP which aimed at providing a fully automatic conversion mechanism for any EXPRESS schema to an OWL ontology. This conversion is implemented as a plug-in within the Protegé software tool (Krima et al., 2009). The automated conversion procedure of IFC into Linked Data presented by Hoang was the first to take into account the existence of the OWL2 profiles EL, QL and RL and thereby making a case for a conversion procedure that results into a layered ifcOWL ontology. The most recent mapping procedure by Pauwels et al. takes the previously mentioned proposals into account and is currently being considered to be the standard for the ifcOWL[7]. For this EXPRESS to OWL conversion method 3 criteria were taken into account, namely (Pauwels & Terkaj, 2016):

1. The ifcOWL ontology must be in OWL2 DL
2. The ifcOWL ontology should match the original EXPRESS schema as closely as possible.
3. The ifcOWL ontology primarily aims at supporting the conversion of IFC instance files into equivalent RDF files.

This conversion allows the ifcOWL to be structured according to the original IFC Object Model architecture which is already described thoroughly in chapter 3. Building Information Modeling. For illustration purposes, Figure 8D shows how the ifcOWL represents the breakdown structure of `IfcSpatialStructureElement`, which resembles the breakdown structure of `IfcSpatialStructureElement` defined in EXPRESS (see Appendix H).

[7] IfcOWL URI: http://www.buildingsmart-tech.org/future/linked-data/ifcowl/20150917_latest/IFC2X3_TC1.owl/view.
IfcOWL Namespace: http://www.buildingsmart-tech.org/ifcOWL/IFC2X3_TC1#.

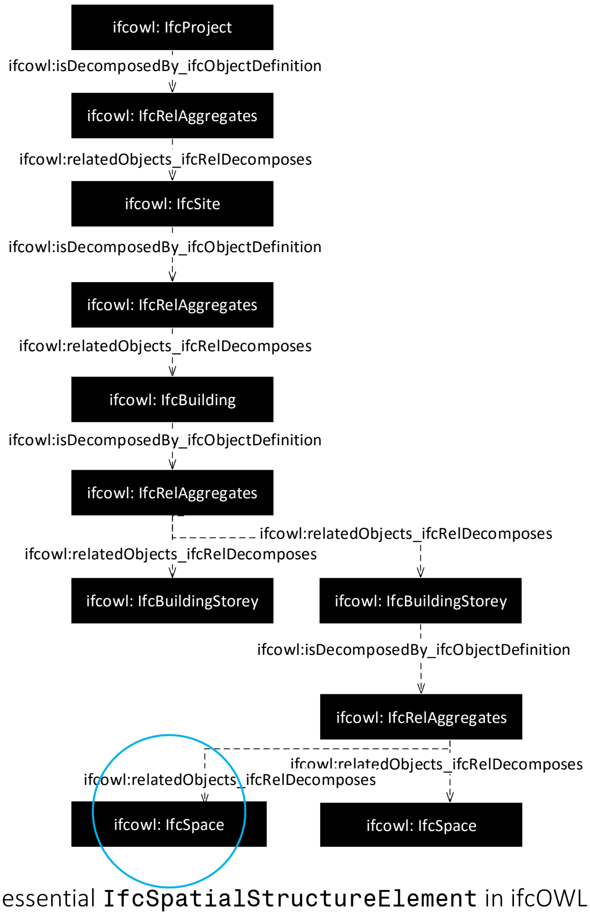

ifcowl: IfcProject
ifcowl:isDecomposedBy_ifcObjectDefinition
ifcowl: IfcRelAggregates
ifcowl:relatedObjects_ifcRelDecomposes
ifcowl: IfcSite
ifcowl:isDecomposedBy_ifcObjectDefinition
ifcowl: IfcRelAggregates
ifcowl:relatedObjects_ifcRelDecomposes
ifcowl: IfcBuilding
ifcowl:isDecomposedBy_ifcObjectDefinition
ifcowl: IfcRelAggregates
ifcowl:relatedObjects_ifcRelDecomposes
ifcowl:relatedObjects_ifcRelDecomposes
ifcowl: IfcBuildingStorey
ifcowl: IfcBuildingStorey
ifcowl:isDecomposedBy_ifcObjectDefinition
ifcowl: IfcRelAggregates
ifcowl:relatedObjects_ifcRelDecomposes
ifcowl:relatedObjects_ifcRelDecomposes
ifcowl: IfcSpace
ifcowl: IfcSpace

Fig. 8D An overview of the essential `IfcSpatialStructureElement` in ifcOWL. The blue circle points to a concept as a linking possibility to the sensor- and requirement data set.

The full ontology consists of 1230 concepts and 1578 object properties, while using classes of the imported EXPRESS and LIST ontologies. Also, the ifcOWL ontology uses OWL classes and properties that are defined the EXPRESS ontology which imports on its turn the LIST ontology. Currently, only the (`WHERE`) `RULE` and `FUNCTION` are not implemented within this ifcOWL ontology version (Pauwels & Terkaj, 2016).

Though it appears to be of secondary importance that an instance RDF file can be modelled from scratch using the ifcOWL ontology it seems to be that this ontology is/will be a formal BuildingSMART standard and therefore will be used as a means to express the IFC- SPF data into RDF. Hereby, it appeared that no other ontologies had to be interlinked.

8.3 SE data

As stated in chapter 2. Systems Engineering, Dutch SE requirement specification activities are usually conducted conform the ISO 15288. Even though every company interprets the abstract process descriptions of the ISO 15288 in a slightly different way (partly due to the usage of natural language) there is yet no agreement for a formal ontology that is able to describe the processes in an explicit and unambiguous way (Van Ruijven, 2013). This observation has been confirmed by several discussions with the Semantic Consultancy department of Semmtech. In conclusion, these findings impede the option for ontology reuse. Therefore, it is chosen to develop an OWL ontology which support the before mentioned scope.

It appeared that Ruijven established a certain taxonomy and associated collection of defined relationships within the context of the Dutch infrastructure projects and translated them to a set of 12 consistent and coherent information models which together form an integrated representation of the ISO 15288 standard. Hereof the requirement specification information model and a generic property and status information model will be used as a template for developing the actual OWL SE ontology. These models are described concisely in the following paragraph.

Within the requirement specification information model, requirements are usually classified in terms of severity, engineering discipline, type of requirement. See figure K1 in Appendix K. The requirement specification is allocated to a party that functions as 'client'. Basically, a requirement specification will be a design constraint, functional or a performance requirement. The actual requirement specification is defined as a piece of text that has a status in the context of the building life cycle and references to the original source. A requirement specification has a status as well.

Figure K2 in Appendix K shows the information model for properties of any element of a system. As long as an element of a system (such as a physical object, activity or event) exists on the class level a property is accompanied by a property specification, defining the range (upper and lower boundary) and the unit of measurement in which the property is expressed (Van Ruijven, 2013).

A top-down approach was used based upon the classes and properties of the before mentioned diagrams to develop the actual OWL ontology which is named as Systems Engineering and visualizaed in figure 8E.

Firstly, the class Requirement was selected together with the Party –class and defined respectively as `se:Requirement` and `se:Organization`. `Se:Requirement` could be considered to be a system element and therefore resemble a generic Possible Individual class as shown figure K2 in Appendix K. This assumption allows to specify its properties in a simplified fashion: The class Property Specification has a direct link with the `se:Requirement` class:

```
se:Requirement se:specifiesBoundaries se:QuantityValue .
```

The term of the class Property specification was considered as too generic, and therefore renamed as `se:QuantityValue` to explicitly state (and define) that it can be used for specifying numerical quantities. The original data set of the requirements is structured by its related room and categorized by a condition (i.e. temperature, acoustics). Therefore, the classes `se:Condition` and `se:Room` are introduced to express these real-world entities in an abstract way:

```
se:Requirement se:specifies se:Room .
se:Requirement se:categorizedBy se:Condition .
```

For example, this allows an actor to query for all requirements in regard to the room "De Pronkzaal" or all requirements related to "Acoustics". Furthermore, `se:Room` provides interlinking possibilities to the IFC data set. Finally, as a means of defining an hierarchical structure all concepts of the SE ontology are a subclass of `owl:Thing`.

The properties is lower bound for and is upper bound for were renamed and added to the graph to describe explicitly `se:QuantityValue` and eventually parameterize `se:Requirement`*:*

```
se:QuantityValue  se:lowerBoundary   xsd:float ,
                  se:upperBoundary   xsd:float .
```

Finally, slots were defined for each property. From the context of this research, the definition of only `rdfs:domain` and `rdfs:range` was sufficient. See figure 8E for the resulting graph. In conclusion, the resulting ontology consists of 6 concepts while having 7 properties. The ontology has been called Systems Engineering (with prefix as se).

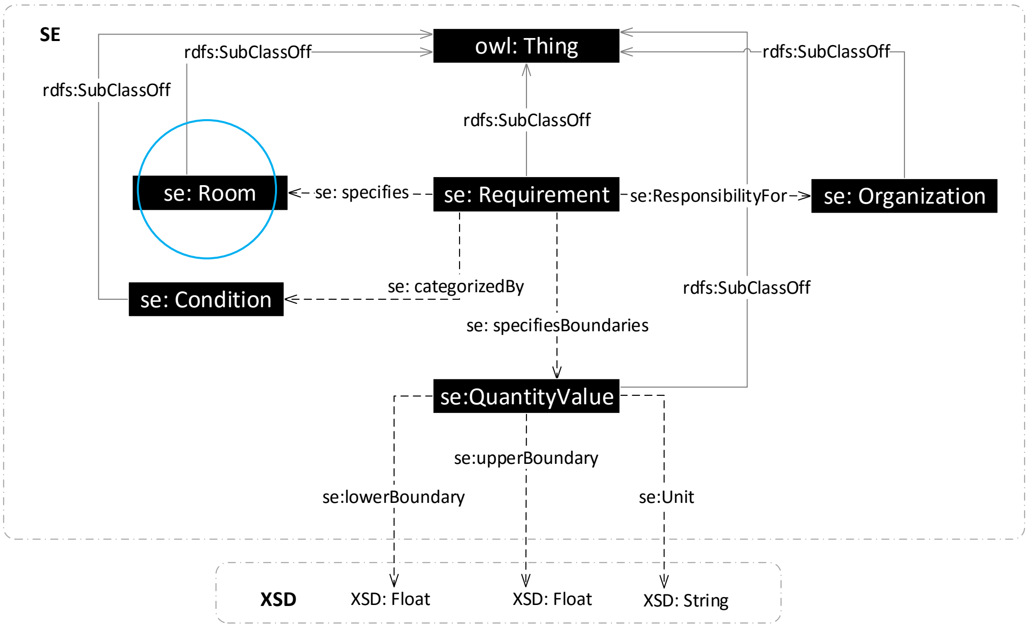


Fig. 8E An overview of the established SE ontology to express the requirement data set in a formal way. The blue circle points to a concept as a linking possibility with the sensor- and building data set.

9. Linked Data generation

The goal of this part is to provide an elaborated explanation about the way the Linked Data sets are actually created according to the selected domain ontologies. There are 4 main Linked Data principles which will be taken into account during the conversion process and the structuring of this chapter, namely: (1) The usage of URIs for identification of resources, (2) The use of HTTP, (3) Usage of open standards, like RDF and (4) Referral to resources in other graphs. Each of the graphs will be imported into the Allegrograph triplestore. During this phase the following technologies were used: AllegroGraph, Python 2.7 using the RDFLib 4.2.1 and Pandas 0.18.1 libraries and Gruff 5.1.7.

9.1 Naming things with HTTP URIs

The URIs for this research are carefully constructed according to the (303) URI- strategy for Linked Data which was initiated by Geonovum and the Dutch government. This strategy is on its turn, based upon the principles of Cool URIs (Overbeek & van den Brink, 2013) which allows the URIs to have the following composition:

http://{domain}/{type}/{concept}/{reference}

It is chosen to use http://example.com/ as a base URI because this domain is free to use without prior coordination or asking permission[8]. Thereafter, the key term id is minted which let users know that the URI concerns a data instance (the other option is def which is used to refer to a definition of a concept of an ontology). This term is followed by a concept that belongs to the real world entity that a practioner wants to express. Finally, a GUID as a means of a reference is added to ensure uniqueness of URIs. A concrete example of a URI construct of a data instance is (Overbeek & van den Brink, 2013):

http://example.com/id/Instant/338d0dae-1d9a-11e6-83f6-240a64020db4

9.2 Describing things with RDF

As already identified in chapter 3, the data sets containing the sensor observations and the system specifications have a tabular data structure (in Excel) while the building model is expressed in IFC- SPF. This enforces the paragraph to be split up into two subparts.

9.2.1 IFC- SPF data

It appears there is already a tool[9] which is able to convert the IFC file into RDF and conforms to the ifcOWL described in the previous paragraph. See Appendix M for a slice of the end result. The appendix shows thereby in blue which namespaces are used and how a `IfcSpace` is described in RDF- triples. A partial RDF- graph was validated successfully via the online W3C- validator[10], wherafter the full data set was successfully imported into the triplestore.

9.2.2 Tabular data

It appears that there are various ways to convert tabular data into RDF via tools like Linked Open Data (LOD)Refine. However, for flexibility reasons it was chosen to use the Python programming language

[8] See: http://www.example.com/.

[9] See: https://github.com/mmlab/IFC-to-RDF-converter.

[10] See : https://www.w3.org/RDF/Validator/.

to conduct the RDF- transformation by following a widely- accepted conversion method. From the existing methodologies available, the CSV on the Web (CSVW) procedure by the W3C[11] was selected due to its generic and stable conversion procedure.

The basic purpose of CSVW is to enable a description of a CSV via metadata by using a second data model: A JavaScript Object Notation for Linked Data (JSON-LD) formatted document. In general, JSON-LD facilitate conversion methods for encoding Linked Data by using the JSON- structure.

In this context, JSON-LD can be described as a table description. Such a file consist of a table schema object which defines an array of descriptions per CSV-column by using various name/values pairs (McGlinn, 2015). In order to let a program iterate over the annotated CSV data and use the annotations per column in the JSON-LD file to create computer interpretable triples, CSVW defines a mechanism to construct URI's (of CSV values). Namely, the names `aboutUrl` and `valueUrl` allows users to construct URI's for subjects and objects. Hereby the URI template: `#concept-{column}` is used as value. When defining `concept` and `column`, `column` will take the cell value in the CSV -column to construct a URI. The specified namespaces in the JSON- LD file are used for constructing predicates. If a datatype is allocated to a CSV- column the CSV-values will be literals which conform to the defined XSD- datatype. Using these principles it is possible to create virtual columns for injecting extra information (CSV on the Web Working Group, 2015).

Even though the logic of an JSON-LD data model is sufficient for a computer program to exhibit the RDF conversion, it was chosen to deviate slightly from the CSVW procedure. Both the SE and sensor Excel data set had to undergo cleansing and transformation operations such as adding and restructuring columns and file conversion (from Excel to CSV) before the conversion could take place. During these operations it appeared to be more efficient to create directly the URI's for the subjects and objects as well instead of creating them using `aboutUrl` and `ValueUrl` in the second procedure in another file. This approach let the processes data preparation on the one hand and data annotation on the other to be separated from each other and therefore reducing the complexity of the overall CSVW procedure.

Both the data cleansing and transformation procedures (via Python) are depicted simultaneously in figure N1 in Appendix N. The associated code is shown in Appendix O. First, the original sensor data table is adjusted by removing and renaming columns. Secondly, extra columns are added of which the headers represents the properties. Each new column consist of URIs to enable RDF transformation of things (concepts) identified by the SSN ontology.

This process is repeated 8 times (since each room has 1 sensor which generates 1 data set). For visualization purposes of the outcome, a truncated Excel- sheet of the room "Introduction" can be found in figure Q1 in Appendix Q. In regard to the SE data a new CSV data set is created, which means that only little transformation procedures were required. See Appendix P for the code and figure Q2 in Appendix Q for the intermediate results.

When having both data sets aligned with the SSN ontology and SE ontology, each one was annotated via the JSON-LD data model. Since the instance data contains already URIs for the subjects and objects only URIs for the properties had to b constructed. This has been done via the `propertyURL` statement. See Appendix R and Appendix S.

Basically, the task of the RDF parser is to read each row of the tabular data model and generate RDF triples. The used RDF parser is based upon the Python code by Walshe, Diarmuid Ryan and Markus Ackermann[12] and adjusted to the specific needs of the data sets. The activity diagram of the parser is shown in figure M2 in Appendix M while Appendix T shows the script for the actual sensor data conversion.

First an empty graph is created to which triples can be added to. Secondly, the code creates triples that describe the CSV-table (as a subject) by using the generic name/values as predicate and object. It then creates then for each CSV- line a `ssn:Observation` subject. This means that the 1320 lines of CSV- data represent 1320 `ssn:observation` instances. Based upon the cell location the code is

[11] See: http://w3c.github.io/csvw/csv2rdf/.

[12] See: https://github.com/CNGL-repo/MTeval/blob/master/rdf_from_csvw.py.

able to determine whether or not a cell value represents a real world entity and with which literal or resource it should form a triple. Since the design of the URI allows an instance to hold its concept, it is possible to link an instance to its concept by using `rdf:type`. In this way the graph is structured following the OWL ontology as described within the previous chapter. See Appendix V. The same procedures (the CSVW method and Python script) were used for converting the Systems Engineering data. The associated code can be found in Appendix U while the result is shown by Appendix W. Finally, of each graph a snippet was produced. After getting nine times a successful result of each RDF-snippet from the online- W3C validator their whole graphs were imported into the same repository as of the IFC-RDF graph within the triplestore.

9.3. Making links to other data sets

The third step is to link the each data set (the output specifications data, a building model and the temperature data set) the room instance which represent the same resource together and thereby interlink the 10 graphs. For example, the rooms `191`, `Hoofdthema1NederlandEnDeWereld` and `KrijgsmachtbredeThemaruimte1` all represent the same entity in the real world. The mapping procedure was conducted by using the `owl:SameAs` predicate. Such a mapping then allows to navigate easily from one data set to another while performing distributed joins.
The script that exhibits the semi-automatic mapping procedure is shown in Appendix X. Firstly, all rooms of each data set are queried and put into 3 separate lists in the same order. By creating a loop it is possible to get of each list the instance that points to the same resource. In the same loop the following triples are created for each defined room in the IFC data set:

```
Ifcowl:IfcSpace_191 owl:sameAs ex:id/Room/Hoofdthema1NederlandEnDeWereld
Ifcowl:IfcSpace_191 owl:sameAs ex:id/PhysicalPlace/KrijgsmachtbredeThemaruimte1
```

These triples are then added to a separate *Link* graph: See Appendix Y. Since `owl:sameAs` appears to be a symmetrical and a transitive property, it is only required to assert two triples per room (Allemang & Hendler, 2011). Namely, because of these qualities it is possible to (if necessary) infer the linkage between Hoofdthema1NederlandEnDeWereld and KrijgsmachtbredeThemaruimte1 by using an inference engine of the triplestore. Finally, this link graph was validated successfully and imported into the Allegrograph triple store as well. By adding this graph to the triplestore all 3 graphs are now linked to each other (via assertion or inference). See figure 9A for a visualization of the final result.

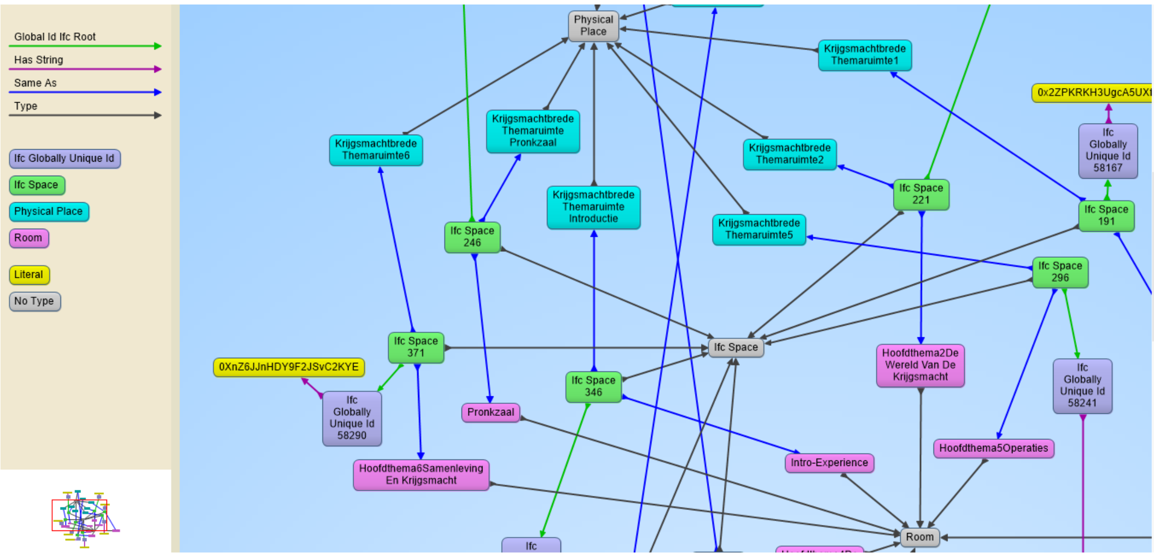

Global Id Ifc Root
Has String
Same As
Type
Ifc Globally Unique Id
Ifc Space
Physical Place
Room
Literal
No Type
Physical Place
Krijgsmachtbrede Themaruimte1
0x2ZPKRKH3UgcA5UX
Ifc Globally Unique Id 58167
Krijgsmachtbrede Themaruimte Pronkzaal
Krijgsmachtbrede Themaruimte6
Krijgsmachtbrede Themaruimte2
Ifc Space 221
Ifc Space 191
Krijgsmachtbrede Themaruimte Introductie
Ifc Space 246
Krijgsmachtbrede Themaruimte5
Ifc Space 296
Ifc Space 371
Ifc Space
0XnZ6JJnHDY9F2JSvC2KYE
Ifc Globally Unique Id 58290
Ifc Space 346
Hoofdthema2De Wereld Van De Krijgsmacht
Ifc Globally Unique Id 58241
Pronkzaal
Intro-Experience
Hoofdthema5Operaties
Hoofdthema6Samenleving En Krijgsmacht
Room
Ifc

Fig 9A A simplified visualization of the 3 linked RDF- graphs based upon the room- entity which was defined in all three graphs.

10. Rule - based verification

The aim of this chapter is to provide a description about the development of a rule mechanism that is able to check if current sensor values lie between the specified boundaries. This checking procedure allows to visualize automatically whether or not a room of the Building Information Model complies to a certain requirement. During this stage the following technologies were used: AllegroGraph and Python 2.7 using the RDFLib 4.2.1, Urllib and IfcOpenShell 0.4.0 libraries.

From the Semantic Web context of this research, rules can be defined as representations of knowledge. They are basically represented in the form of IF-THEN clauses containing logical functions and operations and can be expressed in rule languages or formats, such as Semantic Web Rule Language (SWRL) or SPARQL. Since the required room verification does not exceed the expressive power of power of SPARQL, dedicated rule languages (SWRL, N3, RIF and SPIN) are avoided.

The first part of this chapter explains the used SPARQL query by decompose the actual SPARQL code. Subsequently, a concise explanation will be provided in how the query is able to retrieve verification results from the online triplestore. Finally, the procedure to visualize the results will be described.

10.1 SPARQL Query

The logic of a SELECT SPARQL query specifies that if all the conditions are matched, the conclusions are operated. These conditions described as a graph pattern in the WHERE clause. For this case, the SPARQL query is extended by the keywords: FILTER and a subquery (also a child query). The child query allows the retrieval of sensor values of the current time, while the FILTER checks for each room if the statement about the retrieved sensor values holds true. If true, the room will be retrieved as a verification result.

The subquery is shown in figure 6A. This small query retrieves the most recent date time value and stores this as a variable `max_DateTime`.

```
SELECT (MAX(?DateTime) AS ?max_DateTime)
WHERE {?Instant time:inXSDDateTime ?DateTime}
```

Then this variable is used as a condition in the graph pattern which is used for the retrieval of sensor values. This goes as follows. First, the GUIDs of each rooms is retrieved by the pattern:

```
?IfcRoom a                          ifcowl:IfcSpace .
?IfcRoom ifcowl:globalId_IfcRoot    ?IfcGuid .
?IfcGuid express:hasString          ?IfcString .
```

Via the next statement it is possible to navigate to the other SSN sensor data set:

```
?IfcRoom owl:sameAs ?SsnRoom .
```

In this set the actual sensor values (`?Float`) at `max_DateTime` are selected. Hereafter the Systems Engineering set is entered which allows the selection of the lower boundary (`?LowerBoundary`) and upper boundary (`?UpperBoundary`) via the triple:

```
?IfcRoom owl:sameAs ?SeRoom .
```

Now that all current sensor values are selected together with the boundaries, it possible to retrieve the (GUIDS of the) rooms that do not comply by imposing the FILTER declaration:

```
FILTER ( (?Float < ?LowerBoundary || ?Float > ?UpperBoundary) )
```

By using the `ORDER BY desc` statement it is possible to order the results. The full SPARQL query is shown in Appendix Z.

10.2 REST Protocol

In order to visualize the malfunctioning rooms it is crucial to be able to fire the described query from a (Python) program and then store the retrieved result. Since the Allegrograph triplestore is online accessible though the HTTP protocol the Python script has to make use of Representational State transfer (REST) functionalities. REST can be considered to be the software architecture of the web. Its protocol allows systems to communicate over each other over the web using typical HTTP terms, like `GET`, `POST` and `DELETE`. This protocol allows the script to send the query to the remote triplestore, get the (XML) results and store the result in a variable. The full code is shows in Appendix Z.

10.3 IFC Visualization

The final step is to visualize the results within a BIM model. This can be established by using the retrieved GUID(s) in the XML result to look up the room(s) in the building model. When each room is found, a virtual RGB (Red-Green-Blue) color is given to its `IfcSpace` equivalent in the IFC file. This is done by using the IfcOpenShell library[13]. The result is shown in fig 10A.

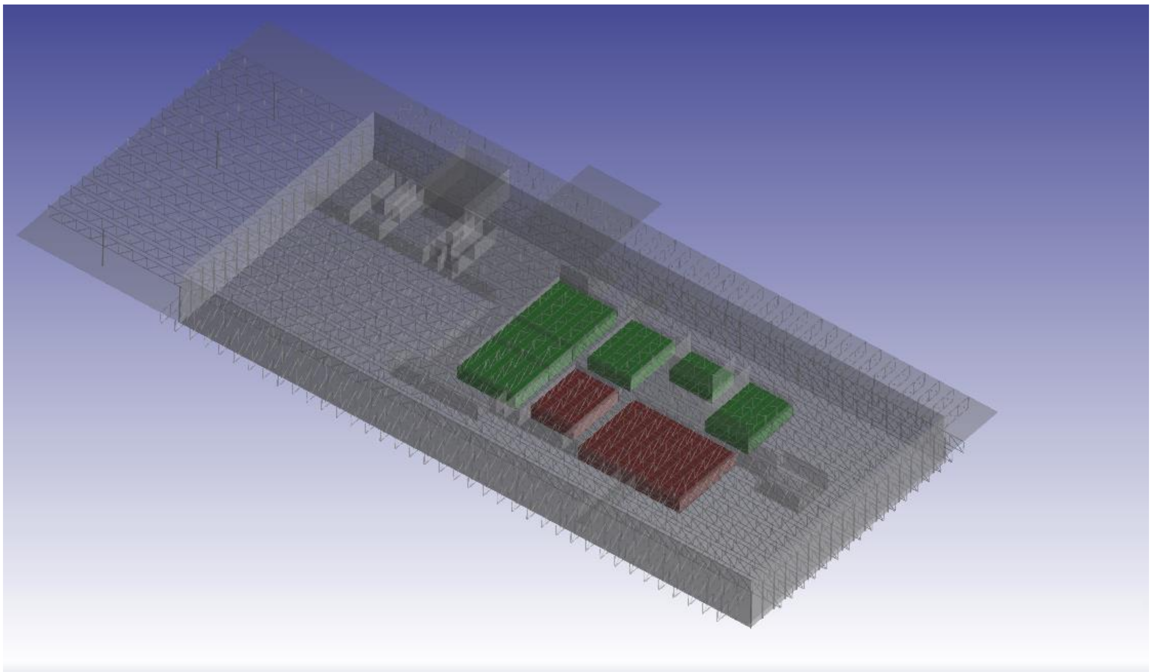


Fig. 10E A visualization of the verification results in an IFC- model of the National Military Museum.

[13] See: http://ifcopenshell.org/python.html.

11. Conclusion & Discussion

The aim of this chapter is to provide a satisfiable answer to the research question as described in section 1.3. This will be done by describing the developed “proof of concept” (or prototype) while stating its societal and scientific relevance therafter. Subsequently, several design decisions will be highlighted which could be improved if this study would be reproduced in the future. Finally, multiple proposals will be recommended for further studies that could dwell upon this research.

11.1 Conclusion

Nowadays, information management within the built environment and particularly information exchange between different project phases and associated systems is becoming increasingly important.

Building Information Modeling (BIM) approach is therefore becoming a global standard within the AEC domain. However, limitations have been found by academica as well as by practioners within the industry when integrating and sharing data sets among different AEC domains as defined by the BIM strategy. Therefore, this research explored a new phenomenon called Linked Data in order to tackle this interoperability issue in a important and relevant use case.

This use case of this research project revolves around the fact that owners of buildings (represented by Systems Engineering (SE)) require more and more from their assets while the need to synchronize heterogeneous sensor data sources for monitoring the building performance become thereby more essential. Yet literature explains that Facility Management (FM) is still in its infancy in its adoption to advanced information models and is unable to integrate or reuse data sets from other data sets automatically. Therefore, the research question during this project was:

> **In which way could linked sensor data be integrated into the BIM model to check the performance of its associated building based upon the agreed output specifications during the maintenance & operations phase?**

In order to solve the research question, a combined ontology was engineered to enrich and define the SE requirements and FM temperature sensor data in an explicit way, which allowed to visualize which rooms in the associated BIM model did not comply to the system specifications. Thereby, the building information model was generated into a RDF data model (via an IFC-RDF converter) according to the IfcOWL. The sensor data set was semantically expressed according to the following ontologies: SSN, time, QUDT and DUL. In regard to the requirement data set, it occurred that there were no tested (let alone widely accepted) ontologies. For this reason, a new Systems Engineering ontology was engineered. Hereby an ontology pattern (information models based upon the SE ISO 15288 standard) was used. In order to convert the latter two data sets in actual RDF data models, the generic CSVW procedure was used. The pattern for the used HTTP URI to identify resources in those 2 data sets was constructed as: `http://{domain}/{type}/{concept}/{reference}.`

Finally, the IF-ELSE logic of a SELECT SPARQL query to a remote triplestore was used for the retrieval of the GUIDS of rooms of which the observed sensor values exceeded the specified upper and lower boundaries. This query was formulated in such a way it could make use of the REST architecture of the Web. The retrieved results were used to provide virtually a RGB (Red-Green-Blue) color to the `IfcSpace` elements of the BIM model.

11.2 Discussion

11.2.1 Relevance of the research

The relevance of this research is twofold. From an industry- like perspective, this proof of concept explaines that by using open Semantic Web technologies it is possible to improve the building performance of construction objects.Namely, the developed proof of concept provides insight how to transform and integrate other data sets (like humidity, occupancy patterns and weather conditions) in order to verify numerous other output specifications as well. Then, the client would (and/or the occupants would) be able to use a building which complies to their needs while the contractor is able monitor the building more effectively (and thereby avoiding fines). Ultimately, this situation also could lead to a more solid relationship between client and contractor.

By means of incorporating a real world case scenario and showing thereby the added value of Linked Data this report aims to stimulate the adoption of open Semantic Web technologies within the built environment. From a scientific- like perspective therefore, this report contributes to the pioneering Linked Data movement within the AEC domain which aims to improve the long- lasting interoperability problem by using Semantic Web technologies. This is essential since such open technologies could support the AEC domain during the upcoming data revolution (i.e. the increasing importance of predictive analytics) by making data accessable, retrievable, reusable and integrated witch each other in a meaningful way without being forced into a vendor lock- in.

11.2.2 Future work

A way to improve this research is to develop a more sophisticated Linked Data tool. Namely, within this research the tool development has been subjected to several limitations and objectives and should therefore be considered for scientific use only. Furthermore, the research could be improved by applying the CSVW- conversion method strictly and thereby relying more upon a standardized logic (derived from the JSON- LD table) instead of a piece of code. An advantage would be that other users would be able to adjust the information model on their own terms just by consulting the extensive and open documentation on the Internet instead of making contact to the specific developer of a customized script.

As explained during the research, all graphs were imported as one within a triplestore to allow SPARQL- queries to retrieve information. However, in a real world situation all three graph data sets- (1) requirements data, (2) building data and (3) temperature sensor data- are obviously produced and stored by three different parties and therefore reside each in a separate triplestore. Then, in order to get the same results as shown within this report a *federated* SPARQL- query has to be formulated which retrieves the data from 3 SPARQL- endpoints instead of 1 SPARQL- endpoint.

Furthermore, it would be an improvement to put more emphasis upon the determination of correspondences between concepts of different ontologies. This aspect is called ontology aligment and was relatively underexposed during the research. For example, it could have been possible to interlink the RDF- graphs by using predicates coming from the Simple Knowledge Organization System (SKOS) vocabulary, like `skos:closeMatch` or `skos:exactMatch`.

Further research could focus upon extending the use of sensor data by Semantic Web technologies. For example, allowing sensor devices to communicate with each other in a meaningful way despite their different protocols and technologies. A means to conduct such an experiment is by using the

Smart Appliances reference ontology (SAREF) ontology. Ultimately, such an experiment could lead to automation of building services based upon the agreed output specifications.

Another research area is the exploration of the possibilities for and consequences of the development of a Linked Data tool that operates upon actual streaming (sensor) data. It would be interesting to explore to which extent the proposed research method in this research would hold in such a project or that a whole new Linked Data recipe would be necessary like the RDB to RDF Mapping Language (R2RML). A final result would be to actually visualize the checked rooms in real time and enable actual monitoring via the Internet.

The above highlighted aspects could improve significantly the developed Linked Data prototype. By using the conclusions and recommendations within this report it is possible to develop a simple Linked Data information system which can be used in practice. Ultimately, such information systems could support the Linked Data movement within the AEC domain with their goal of achieving improvements in cost, value and environmental performance in the creation and operation of civil infrastructure and buildings on a global scale.

12 References

Algemene Rekenkamer. (2013). Contractmanagement bij DBFMO-projecten. *Algemene Rekenkamer*, 50. Retrieved from http://www.rekenkamer.nl/Publicaties/Onderzoeksrapporten/Introducties/2013/06/Contractmanagement_bij_DBFMO_projecten

Allemang, D., & Hendler, J. (2011). Semantic Web for the Working Ontologist, (June), 1. http://doi.org/10.1016/B978-0-12-385965-5.10001-9

Beetz, J. (2009). *Facilitating distributed collaboration in the AEC / FM sector using Semantic Web Technologies*.

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*(01), 89. http://doi.org/10.1017/S0890060409000122

BKCASE Editorial Board. (2014). Guide to the Systems Engineering Body of Knowledge. *Guide to the Systems Engineering Body of Knowledge (SEBoK)*, 945. Retrieved from http://g2sebok.incose.org/app/mss/menu/index.cfm

Borgo, S., Sanfilippo, E. M., Aleksandra, S., & Terkaj, W. (2015). Ontological Analysis and Engineering Standards, 1–28. http://doi.org/10.1007/978-3-319-15326-1

Bou-ghannam, A. (2013). Foundational Ontologies for Smarter Industries, 1–28. Retrieved from http://www.redbooks.ibm.com/abstracts/redp5081.html?Open

Bryde, D., Broquetas, M., & Volm, J. M. (2013). The project benefits of building information modelling (BIM). *International Journal of Project Management*, *31*(7), 971–980. http://doi.org/10.1016/j.ijproman.2012.12.001

BuildingSMART. (n.d.). architecture diagram. Retrieved June 11, 2016, from http://www.buildingsmart-tech.org/ifc/IFC2x3/TC1/html/index.htm

buildingSMART. (n.d.). IfcSpace. Retrieved June 11, 2016, from http://www.buildingsmart-tech.org/ifc/IFC2x3/TC1/html/ifcproductextension/lexical/ifcspace.htm

BuildingSMART. (2009). IFD Library. *Framework*, (April), 1–9. Retrieved from https://www.uneto-vni.nl/cms/streambin.aspx?requestid=EA0F4F6F-560D-4D15-9BD6-0B651404B812

BuildingSMART. (2010). Information Delivery Manual Guide to Components and Development Methods. *buildingSMART*, 1–84.

Compton, M., Barnaghi, P., Bermudez, L., García-Castro, R., Corcho, O., Cox, S., … Taylor, K. (2012). The SSN ontology of the W3C semantic sensor network incubator group. *Journal of Web Semantics*, *17*, 25–32. http://doi.org/10.1016/j.websem.2012.05.003

Compton, M., Henson, C., Neuhaus, H., Lefort, L., & A, A. S. (2009). Survey of the Semantic Speci Cation of Sensors. *In 2nd International Semantic Sensor Networks Workshop*.

CSV on the Web Working Group. (2015). Generating RDF from Tabular Data on the Web. Retrieved from https://www.w3.org/TR/2015/REC-csv2rdf-20151217/

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*(2), 206–219. http://doi.org/10.1016/j.aei.2012.10.003

Dankers, M., van Geel, F., & Segers, N. M. (2014). A Web-platform for Linking IFC to External Information during the Entire Lifecycle of a Building. *12th International Conference on Design and Decision Support Systems in Architecture and Urban Planning, DDSS 2014*, *22*, 138–147. http://doi.org/http://dx.doi.org/10.1016/j.proenv.2014.11.014

Davis, A. (1993). *Software Requirements : objects, functions and states.* Prentice Hall.

Dawood, N., Vukovic, V., & Kassem, M. (2015). Bim for Facilities Management : Evaluating Bim Standards in Asset Register Creation and Service Life Planning. *Journal of Information Technology in Construction*, *20*(January), 313–331.

Decker, S., Harmelen, F. Van, Broekstra, J., Erdmann, M., Fensel, D., Horrocks, I., … Melnik, S. (2000). The Semantic Web - on the respective Roles of XML and RDF. *IEEE Internet Computing*, *4*(October), 19. http://doi.org/10.1109/4236.877487

Domingues, P., Carreira, P., Vieira, R., & Kastner, W. (2016). Building Automation Systems: Concepts and Technology Review. *Computer Standards & Interfaces*, *45*, 1–12. http://doi.org/10.1016/j.csi.2015.11.005

Douglass, B. P. (2016). *What Is Model-Based Systems Engineering? Agile Systems Engineering*. http://doi.org/10.1016/B978-0-12-802120-0.00001-1

DSDM Consortium. (2008). DSDM : Enabling Business Agility. *Framework*, 1–289.

DuCharme, B. (2013). *Learning SPARQL*.

Eastman, C. e, Teicholz, P., Sacks, R., & Liston, K. (2011). *BIM Handbook*. *PhD Proposal* (2nd ed., Vol. 1). New Jersey. http://doi.org/10.1017/CBO9781107415324.004

Eastman, P. C. M., Tech, G., Ga, A., & Eastman, C. (2011). 2011 Charles M . Eastman Top Phd Paper Award Information Delivery Manuals To Integrate Building.

Freeman, J. R. (2015). Guide to the System Engineering Body of Knowledge (SEBoK). *Caise15*, *44*(7), 151–154.

Heath, T., & Bizer, C. (2011). *Linked data: Evolving the Web into a global data space (1st edition)*. *Synthesis Lectures on the Semantic Web Theory and Technology 11* (Vol. 1). http://doi.org/10.2200/S00334ED1V01Y201102WBE001

Hoogendoorn, S. (2004). *Pragmatisch modelleren met UML 2.0*. Pearson Education Benelux.

INCOSE. (2006). Systems engineering handbook. *Incose*, 185. Retrieved from http://smslab.kaist.ac.kr/Course/CC532/2012/LectureNote/2012/INCOSE Systems Engineering Handbook v3.1 5-Sep-2007.pdf

INCOSE. (2007). Systems Engineering Vision 2020. *Systems Engineering*, *1*(September). Retrieved from http://www.incose.org/ProductsPubs/pdf/SEVision2020_20071003_v2_03.pdf

International Alliance of Interoperability (IAI). (2007). The buildingSMART Glossary of Terms, (January).

Janowicz, K., & Compton, M. (2010). The stimulus-sensor-observation ontology design pattern and its integration into the semantic sensor network ontology. *CEUR Workshop Proceedings*, *668*.

Kapourani, B., Fotopoulou, E., Papaspyros, D., Zafeiropoulos, A., Mouzakitis, S., & Koussouris, S. (2015). *On the Move to Meaningful Internet Systems: OTM 2015 Workshops*. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (Vol. 9416). http://doi.org/10.1007/978-3-319-26138-6

Kolchin, M., Klimov, N., Andreev, A., Shilin, I., Garayzuev, D., Mouromtsev, D., & Zakoldaev, D. (2015). Ontologies for web of things: A pragmatic review. *Communications in Computer and Information Science*, *518*, 102–116. http://doi.org/10.1007/978-3-319-24543-0_8

Krima, S., Barbau, R., Fiorentini, X., Rachuri, S., Foufou, S., & Sriram, R. D. (2009). OntoSTEP : OWL-DL ontology for STEP. *Technology*.

Liebich, T. (2009). IFC 2x Edition 3: Model Implementation Guide. *System*, 1–188.

Locatelli, G., Mancini, M., & Romano, E. (2014). Systems Engineering to improve the governance in complex project environments. *International Journal of Project Management*, *32*(8), 1395–1410. http://doi.org/10.1016/j.ijproman.2013.10.007

Loffredo, D. (1999). Fundamentals of STEP implementation. *STEP Tools, Inc*, 1–12. Retrieved from ftp://mail.im.tku.edu.tw/Prof_Shyur/PDM/Paper/David.pdf

McGlinn, K. (2015). CSVW Tutorial Files. Retrieved from http://phaedrus.scss.tcd.ie/buildviz/csvw/

Moon, H., Kim, B., & Choi, M. (2013). A Bim Based Data Model for an Integrated Building Energy Information Management in the Design and Operational Stages. *13th Conference of International Performance Simulation Association, Chambery, France, August 26-28*, 3217–3224. Retrieved from http://www.ibpsa.org/proceedings/BS2013/p_2517.pdf

Noy, N., & McGuinness, D. (2001). Ontology development 101: A guide to creating your first ontology. *Development*, *32*, 1–25. http://doi.org/10.1016/j.artmed.2004.01.014

Overbeek, H., & van den Brink, L. (2013). Towards a national URI-Strategy for Linked Data of the Dutch public sector, (2), 1–19. Retrieved from http://www.pilod.nl/w/images/a/aa/D1-2013-09-19_Towards_a_NL_URI_Strategy.pdf

Pauwels, P. (2011). Ontologies in architecture, engineering and construction (AEC) (pp. 19–21). Barcelona.

Pauwels, P. (2014). *Supporting Decision-Making in the Building Life-Cycle Using Linked Building Data*. *Buildings* (Vol. 4). http://doi.org/10.3390/buildings4030549

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

Pfisterer, D., Romer, K., Bimschas, D., Kleine, O., Mietz, R., Truong, C., … others. (2011). SPITFIRE: toward a semantic web of things. *Communications Magazine, IEEE*, *49*(11), 40–48. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6069708\nhttp://ieeexplore.ieee.org/Xplore/cookiedetectresponse.jsp?reload=true\nhttp://www.iti.uni-luebeck.de/fileadmin/user_upload/Paper/IEEEComMag.pdf

Potkany, M., Vetrakova, M., & Babiakova, M. (2015). Facility Management and Its Importance in the Analysis of Building Life Cycle. *Procedia Economics and Finance*, *26*(15), 202–208. http://doi.org/10.1016/S2212-5671(15)00814-X

Radulovic, F., Poveda-Villalón, M., Vila-Suero, D., Rodríguez-Doncel, V., García-Castro, R., & Gómez-Pérez, A. (2015). Guidelines for Linked Data generation and publication: An example in building energy consumption. *Automation in Construction*, *57*, 178–187. http://doi.org/10.1016/j.autcon.2015.04.002

Rijksvastgoeddienst. (2014). Rijksbrede modelovereenkomst DBFMO huisvesting, *Model-Vers*, 141.

Ryen, E. (2008). Overview of the System Engineering Process Prepared by, (March), 10–13, 15–16. Retrieved from https://www.dot.nd.gov/divisions/maintenance/docs/OverviewOfSEA.pdf

Schevers, H., & Drogemuller, R. (2006). Converting the industry foundation classes to the web ontology language. *Proceedings - First International Conference on Semantics, Knowledge and Grid, SKG 2005*, (Skg 2005), 2005–2007. http://doi.org/10.1109/SKG.2005.59

Segaran, T., Evans, C., & Taylor, J. (2009). *Programming the Semantic Web*. *Statewide Agricultural Land Use Baseline 2015* (Vol. 1). http://doi.org/10.1017/CBO9781107415324.004

Semantic Sensor Network Incubator Group. (n.d.). No Title. Retrieved June 11, 2016, from https://www.w3.org/2005/Incubator/ssn/wiki/Report_Work_on_the_SSN_ontology

Szeredi, P., Lukácsy, G., & Benko, T. (2014). *The Semantic Web Explained*. *Governing Sustainability*. http://doi.org/10.1017/CBO9780511807756.003

Törmä, S. (2013). Semantic linking of building information models. *Proceedings - 2013 IEEE 7th International Conference on Semantic Computing, ICSC 2013*, 412–419. http://doi.org/10.1109/ICSC.2013.80

Van Ruijven, L. C. (2013). Ontology for systems engineering. *Procedia Computer Science*, *16*, 383–392. http://doi.org/10.1016/j.procs.2013.01.040

Verweij, S. (2015). Producing satisfactory outcomes in the implementation phase of PPP infrastructure projects : A fuzzy set qualitative comparative analysis of 27 road constructions in the Netherlands ☆ , ☆☆. *Jpma*, *33*(8), 1877–1887. http://doi.org/10.1016/j.ijproman.2015.08.006

Volk, R., Stengel, J., & Schultmann, F. (2014). Building Information Modeling (BIM) for existing buildings - Literature review and future needs. *Automation in Construction*, *38*, 109–127. http://doi.org/10.1016/j.autcon.2013.10.023

Watson, A. (2011). Digital buildings - Challenges and opportunities. *Advanced Engineering Informatics*, *25*(4), 573–581. http://doi.org/10.1016/j.aei.2011.07.003

WBDG. (n.d.). No Title. Retrieved June 12, 2016, from https://www.wbdg.org/resources/cobie.php

Werkgroep Leidraad Systems Engineering, ProRail, Rijkswaterstaat, Vereniging van Waterbouwers, NLingenieurs, BouwendNederland, & UnetoVni. (2013). Leidraad voor Systems Engineering binnen de GWW-sector, 73.

Zhang, C., Beetz, J., & Vries, B. De. (2013). Towards Model View Definition on Semantic Level : A State of the Art Review, (July 2015), 1–10.

Appendix A Interview ISSO

04-03-2016 Interview with ISSO

Interviewee 1: Rob Van Bergen (Chief Executive Officer)

Motive(s): The party is a neutral and respectable knowledge institution within the installation industry which prescribes normative guidelines that are widely adapted by the Dutch construction industry.

Goals: This interview is held in order to determine (1) in which way is sensor data used for monitoring in the construction industry (2) if BIM used during the operations & maintenance phase (3) how sensor data is being stored and used (4) in which way sensor data is used for (preventive) maintenance

General questions concerning the construction industry:

1. In which way would you appreciate the level of monitoring for compliance to the agreed output specifications at your company? Have you ascertained difficulties on this topic?

> Interviewee cannot provide a reliable answer.

2. To which extent does Building Information Modeling (BIM) plays a role in monitoring construction projects?

> The use of Building Information Modeling is limited unfortunately: This is because the operations & maintenance sector is in it's infancy. This is because the main attention of the industry was to incorporate BIM in the design and realization phases rather than on the facility management. The BIM approach has been introduced only 2 or 3 years ago.

Technical questions concerning sensor data:

3. What are the typical characteristics of sensor data (besides being real time and changing continually)

> An important attribute of sensor data is that it is able to let you discover trends and that it helps you to use this new knowledge to improve workflows in the building.

4. To which extent do sensor data play a role in what way during monitoring of projects.

> I think every building has a building management system at the moment. However, it is not always used for monitoring the performance: This is only done when facility management parties are obliged by contract. Also, they only use the sets to optimize the building performance if they are told do so (which is rarely the case).

5. Do sensor data have other purposes besides only monitoring? Could you mention these?

> To my understanding I do not think that data is being used in many ways: I think it is used also used for keeping track of the energy consumption of buildings.

6. In which ways are sensor data getting stored (i.e. spreadsheets, relational databases). Is this enough to reuse this data together with other types of data sets?

Interviewee cannot provide a reliable answer.

7. Besides sensor related data, do other (external) datasets play a role within the operations & maintenance phase? If so, which kind of data? If not, why is this the case?

Most of the time they only use sensor data of the environment of the building in order to keep track / lower energy consumption.

8. To which extent does the construction sector perform preventive maintenance?

Currently, the facility management sector performs corrective measurements based upon the actual state of the building component.

9. Do you think that there will be need to monitor performance and perform (preventive) maintenance based upon combinations of different and continuous changing data sets?

I think this would be very interesting, because it would let the management improve work flows in buildings (as already stated).

Appendix B Interview Facilicom

16-03-2016 Interview with Facilicom

Interviewee 1: Gerard Wennekes (Facility coordinator of the Dutch Tax and Customs Administration (DTCA) in Doetinchem)

Motive(s): The party is a specialized firm in facility management services. They have experience with several DBFMO- projects like some of the Dutch National Monuments in The Hague.

Goals: This interview is held in order to determine (1) how the process goes for monitoring the building during this project (2) if BIM is used during the operations & maintenance phase and (3) in which way sensor data is used for (preventive) maintenance

General questions concerning the monitoring process of this particular DBFMO-project:

1. Which parties are involved during the maintenance and operations phase of DBFMO- projects? Which activities do the perform.

> There are two contractual parties: Central Government Real Estate Agency (CGREA) and the DTCA.

2. Which actions are performing each of these parties in what order?

> The CGREA is concerned with the state of the building (“the hardware”), while the DTCA deals with the services provided by us (“the software”).

Specific questions concerning the monitoring system (and sensor data)

3. To what extent is BIM used in the facility management industry?

> From my experience, I do not think BIM is widely used in this domain. We monitor critical rooms for abnormal values via the Facility Management Information Model (instead of a BIM model) based upon data collected by and stored in the building management system.
>
> However, this is not ideal. The knowledge that is built up during the previous design and build phases is nullified. In order to set up a FMIS we have then to set up the model from scratch (i.e. importing the drawings) and check if it is the same as the BIM model. Also I think this is a pity, due to the fact you could use BIM models to query efficiently for sophisticated information.

4. Other than sensor data, which data sets do you use at the moment for managing the building? If so, how are these stored and interchanged? If not, why not?

> We only user sensor information: We do not have permission for using all generated data due to the fact that parts of these are confidential.

Appendix C Interview Strukton (BIM department)

04-03-2016 Interview with Strukton

Interviewee 1: Pepijn van der Vooren (BIM – specialist at the design & realization department)

Motive(s): The company has proven itself for approximately 80 years as a solid construction & civil company and is therefore a good indicator for the level of knowledge about information management (in the operations & maintenance phases) within the whole industry in The Netherlands.

Goals: This interview is held in order to determine (1) in which way is sensor data used for monitoring in the construction industry (2) if BIM used during the operations & maintenance phase (3) if sensor data being stored and used for other practices (4) in which way sensor data is used for (preventive) maintenance

General questions concerning the construction industry:

1. In which way would you appreciate the level of monitoring for compliance to the agreed output specifications at your company? Have you ascertained difficulties on this topic?

> Interviewee cannot provide a reliable answer.

2. To which extent does Building Information Modeling (BIM) plays a role in monitoring construction projects?

> BIM plays an essential role within the design and realization phase of our projects. The design & realization department implements the output specifications within the building model: This allows to simulate the performance of future buildings and shows us if the as-planned building meets the requirements.
>
> At the moment the workflow between the design & realization department and the maintenance & operations department is separated due to the fact that they have their own data warehouses. Another issue is that these sources are not synchronized. The result is that the information (i.e. design and monitoring data respectively) of a single project is spread over multiple distinct data storages, which should not be the case.
>
> This is because BIM usage within the operations & maintenance phase is in its infancy: They started only a year ago with implementing the BIM approach within their workflow. At the moment, the facility management data is spread across many (local) databases and documents (i.e. Excel). Also, the data (CAD or Excel) from sub-contractors is integrated into these sources manually, due to the fact that both information sources is structured differently.

Technical questions concerning sensor data:

> 3. What are the typical characteristics of sensor data (besides being real time and changing continually)
>
> > Another typical attribute of sensor data is that (in comparison with other data) these are diverse: The reason is that we have many sensors which measure each different variables. For example, we have besides climate sensors also sensors which keep track of the occupancy of buildings.

4. To which extent do sensor data play a role in what way during monitoring of projects.

Interviewee cannot provide a reliable answer.

5. Do sensor data have other purposes besides only monitoring? Could you mention these?

Yes, in our company we use the sensor data to test whether the performance of the as-built buildings match with the one of the as-designed buildings.

6. In which ways are sensor data getting stored (i.e. spreadsheets, relational databases). Is this sufficient enough to reuse this data together with other types of data sets?

Currently, the sensor data is getting stored as XML (or gbXML) files into relational databases. We started with saving our data in this way since only last year. This allows us to read-in the sensor data in our simulation program. We do not mix data sets together. However, I believe this will be the future since the rise of Big Data.

7. Besides sensor related data, do other (external) datasets play a role within the operations & maintenance phase? If so, which kind of data? If not, why is this the case?

Other data sets that are used during the operations & maintenance phase are those of the subcontractor i.e. durability of building components. Another type is the outside environmental data (only temperature and the relative humidity level). However even though this kind of data is considered not as internal general data, this *is* sensor related data. We do not use other (external) data sets like i.e. the one of the KNMI.

8. To which extent does Strukton perform preventive maintenance?

We only conduct corrective maintenance based upon historical and current measurements: For example, we perform maintenance after outliers are observed during monitoring via the sensors. We do not act upon predictions: However, there are future ideas within the company about using this types of predictive data.

9. Do you think that there will be need to monitor performance and perform (preventive) maintenance based upon combinations of different and continuous changing data sets?

Yes, I think this could provide us new insights when performing maintenance. It is thereby essential that the storage of information or data is integrated as a whole. I believe that the BIM approach / model should play a central role herein.

Appendix D Interview Ministry of Defense

07-03-2016 Interview with Ministry of Defense

Interviewee 1: Richard van Asselt (Facility and Logistics division)

Motive(s): The party is a neutral institution of which a department now takes care of one of the most enormous DBFMO – projects in The Netherlands.

Goals: This interview is held in order to determine (1) how the process goes for monitoring the building during this project (2) if BIM is used during the operations & maintenance phase (3) how sensor data being stored and (re)used (4) in which way sensor data is used for (preventive) maintenance

General questions concerning the monitoring process of this particular DBFMO-project:

1. Can you tell me if this project is still the most enormous DBFMO- project within The Netherlands?

> I don't know for sure. However, concerning square meters I can tell you that this project is one of the hugest DBFMO- projects: For example, it is greater than your case study (The National Military Museum).

2. Which parties are involved during the maintenance and operations phase of DBFMO- projects?

> In our case, the Ministry of Defense (which I represent) and a sub-contractor called Comfort is involved during the maintenance and operations phase. Also we are able to involve third (and neutral) parties for performance checking. One of these checks is the client satisfaction. We both hire such party to measure how the performance of the building meets the requirements of the client.
>
> Furthermore, when revisions (approved by myself) have to be made in the contract the National Government will be involved. They will implement these changes within the DBFMO-contract (i.e. changing the associated output specifications) and perform cost-benefit calculations.

3. Which actions are performing each of these parties in what order?

> The contractor takes care of the maintenance. They use their own software to determine (un)availability which provides the basis for determining their payout. The sent report will then be analyzed by the National Government for approval. When approved, the National Government will conduct the required payment transactions.

Specific questions concerning the monitoring system (and sensor data):

4. How is the building management system used? In what way is the information stored?

> At the moment they use two systems: one system is for logging (i.e. registration of the number of visitors) and is probably registering sensor values. the other one is more a planning tool called Planon for making certain reservations (i.e. for meeting rooms). I don't know how the data is stored at the moment.

5. Other than sensor data, which data sets do you use at the moment for managing the building? If so, how are these stored and interchanged? If not, why not?

> Yes, we use other data sets as well. They are very diverse (i.e. the number of prints of a printer during a day) and are stored in several ways across different (local) databases (I don't know which type however). Sometimes, I even get the information per e-mail. Because the information is not synchronized, I am concerned that I do not have control over the building. For example, I cannot perform analysis for performance enhance of the building. The think I can do at the moment, is observing "trends" by eye.

General questions concerning the current state of the facility management

6. To which extent does the Ministry of Defense perform preventive maintenance?

> Currently, we only perform maintenance based upon the actual state and standardized preventive maintenance (i.e. we have a 3-monthly schedule for cleaning the windows). Furthermore, we can check if some things has to be replaced based upon known life durations.

7. In what ways would you improve the monitoring process? Could you name concrete things that should be researched?

> At the moment we do not have concrete KPI's. This means it is necessary to check which KPI's could be used to monitor the building. Furthermore, the client (and thereby I refer to myself), does not have a sufficient tool to check the building performance by themselves. At the moment, I can to this based upon the supplied reports by the contractor.

8. Do you have any ideas how the facility domain will evolve in the near future, particularly concerning data and information (re)use?

> I think that "the more you measure, the more you gain in knowledge". Thereby it is important to synchronize the information in order to perform analysis. As already said, I am not able to do this.
>
> Furthermore, I rely solely on the information by the sub-contractor. If I want to check upon them I have to measure by hand (i.e. measuring the room temperature via a thermostat). I think it would be great if I could check upon them in a transparent way. I also think that using synchronized data, it is possible for the sub-contractor to conduct their tasks more cost-efficiently.

Appendix E Interview Semmtech

18-02-2016 Interview with the Domain Consultancy department of Semmtech

Interviewee 1: Coen Dorge (consultant Domain Consultancy)

Motive(s): The party is a specialized firm in information management and Semantic Web services.

Goals: This interview is held in order to get familiar about how a business process is analyzed in practice.

General question concerning the monitoring process of a DBFMO-project:

How could a monitoring process be described within a DBFMO- context?

At the moment a verification tool as proposed by this research does not not exist yet for buildings though it is already implemented in civil projects. However, in respect to the latter case such a verification tool does not make use of Semantic Web functionalities.

Normally, a verification tool is able to record sensor data automatically as well as data that is asserted by hand. In regard to the first case, measurements take every time duration which allow the system to check periodically whether or not the situations complies to the requirements. The associated database will get notified by the tool, if the situation does not satisfies the specifications. Based upon these notifications the actual payments are calculated and sent as an invoice to the contractor.

Appendix F Interview Heijmans

03-03-2016 Interview with Heijmans

Interviewee 1: Laurens Timmerman (Facility Management)
Interviewee 2: Peter Muller (Measurement & Control technology)

Motive(s): Both actors are heavily involved during the maintenance and operations phase of the provided use case (The National Military Museum)

General questions concerning the National Military Museum:

1. For which objective(s) is (are) sensor data used?

    The main purpose is to use the sensor data to check whether the contractor complies to the agreed output specifications which concern solely the part: 4. Climate conditions.

2. What are the most critical rooms or areas within the National Military Museum?

    These spaces which are marked with "B*" (in the overview of the List of Requirements) and are primarily rooms wherein expositions are installed.

3. To which extent is it possible to configure the HVAC- systems?

    The configuration of the HVAC systems is very flexible so the apparatus can be adjusted locally: It is even possible to just dim only one light spot in a room, while the other ones remain unimpaired.

Technical questions concerning the Building Management System (BMS):

4. In which way are the inside operations influenced by changes in the (outside) environmental climate?

    The undertaken activities are mainly corrective: So, the outside conditions affect the inside operations only after changes are occurred. However, the exact relationship has not been found (yet).

5. To which extent is it possible to conduct preventive maintenance within the museum?

    We are not able to perform preventive maintenance (except lighting within the glass cabinets, because we know the life duration of the lightbulbs). In general, we only perform corrective maintenance.

    However, it would be great if we could predict the values of certain set conditions in order to perform preventive maintenance and/or decrease energy usage. Namely, this would decrease the necessity to use extra cooling / heating / ventilation in order to comply to the requirements.

    At the moment, the (change of) the set point for temperature is based upon historic information (namely, the average of the outside temperature of the last four days) and not upon future circumstances.

6. Which variables are measured as well, besides the already mentioned sensor data?

   The only variable that is measured and is not climate related are the number of people that visit the museum. We keep track of the amount of visitors via the gates near the entrance. However, we do not incorporate these data into the BMS.

7. Does the monitoring system make use of any other external data set / information source?

   No, unfortunately not.

8. What is the relationship between the variables: “temperature”, “comfort (min. and max. operational temperature) and the set- point temperature?

   The required temperature is stated by the operational temperature. We monitor in such a way that this value keeps within between this range. The set-point temperature is the overall set temperature within the specific rooms: Fluctuations are compensated by extra heating or cooling.

9. The values of the sensor data in the database do not contain a separator, like a decimal point or comma. In which way should the values be interpreted? For example, the data set contains a temperature value of “209” Celsius degrees.

   The values should be interpreted with a separator (comma, point) and therefore be seen as a decimal- value.

Appendix G Interview Strukton (data management department)

17-03-2016 Interview with Strukton

Interviewee 1: Barry Tuip (Specialist at the monitoring & data management division)

Motive(s): The company has proven itself for approximately 80 years as a solid construction & civil company and are ahead of other construction companies in managing sensor information during the operations & maintenance phase.

Goals: This interview is held in order to determine (1) in which way is sensor data used for monitoring in the construction industry (2) if BIM used during the operations & maintenance phase (3) if sensor data is being collected and stored in which ways and is used for other practices (4) in which way sensor data is used for (preventive) maintenance

General questions concerning the construction industry:

1. Could you tell me what the state of art is concerning monitoring within the operations and maintenance phase? What is already accomplished and which future is waiting for this domain?

> At the moment there are several trends (i.e. Big Data) of which the facility management could benefit from. According to numerous literature, almost 70 percent of the utility construction sector are not functioning correctly: The amount of data is increasing and it is thereby important to make use of it in an efficient and reliable way. For example, via simulation, information analytics and building information modeling it is possible to compare as- designed buildings with as- designed buildings. Furthermore, the performance of the building objects that do function properly can still be improved with approximately 5-10%.
>
> However, at the moment sensor data is stored in a poor way: Actually only the values are stored and in order to derive knowledge from it u should annotate the data (with meta data). As a result, each construction company is annotating (and thereby storing and reusing) the data in different proprietary ways. At the moment there is no single main standard in how to store building sensor data (in comparison to the IFC- data model in the design and build phases).

2. To which extent does Building Information Modeling (BIM) plays a role in monitoring construction projects?

> Currently, the use of BIM is very limited in comparison with the design and build parties/departments in the construction sector. The reason is that those parties are not putting more information than is required to meet their own objectives. So, because of the fact that the facility management sector does not get involved most of the times during the design and build phase of projects, their needs are not taken into account. BIM- objects are used only by the facility management firm when the client demands such a model for the maintenance and operations phase. However, due to the rise of performance based contracts these parties will probably will get more involved in future projects and will become eventually a standard as well.

Technical questions concerning sensor data:

3. What are the typical characteristics of sensor data (besides being real time and changing continually)

> Other characteristics are the accuracy of the measurements. For example, the building management system that we use (this system called Priva is used in almost all building projects) cannot store decimal values. So a temperature of 19.5 ℃ is stored as 195. This means we have to take into account within in our analysis to divide such values by a factor of 10.

4. How important is sensor data during the monitoring of projects? I.e. Which variables / subjects are being monitored and is data being collected through sensors?

> This depends very much upon the budget of the parties (i.e. the client). For example, for a project we had to place only 6 sensors in a 6- storey building: Even though it measured different variables, we could not derive any knowledge from this data in order to enhance its performance. If parties choose to not use sensors, they can (and will) act based upon only complaints or questions by the building users.

5. Do sensor data have other purposes besides only monitoring? Could you mention these?

> Sensor information is used mainly for monitoring objects. However, It depends very much in what way you reuse the information that makes the data useful via combining sets or sophisticated analysis. In doing so, the data can then be reused for various objectives like energy management, user interaction, maintenance, facility services (i.e. cleaning) and comfort management.

6. Via which way is these sensor data collected?

> You should visualize the process as follows. The Building Management System collects the sensor data. It has a module which allows it to store the values. Furthermore, the energy data is collected by (tools of) energy companies while data of elevator is collected by their manufacturers. Thereby, each source has its ow proprietary protocols and formats.

7. In which ways are sensor data getting stored (i.e. spreadsheets, relational databases) and where (internet or local)? Who is in charge of this database?

> Unfortunately, I do not know the answer to this question. However I can tell that the client is always the owner.

8. Besides sensor related data, do other (external) datasets play a role within the operations & maintenance phase? If so, which kind of data? If not, why is this the case? How are these data stored?

> Yes, we do use other data sets for our analysis. Examples are the weather data by the KNMI or the data set of the number of visitors.

9. Are all these data (sensor and other) stored in a central way? If yes, why? If not, why not? How is the availability ("beschikbaarheid") of critical rooms checked?

> The sensor data is stored within the building management system and can be retrieved which allows you to perform simple analysis within this environment. Or the company retrieves the

sensor data (together with all of the data from various sources) into a so called online local-central data ware house. Strukton is doing the latter: Once per day. In that central database we have actually 9 local databases. For each storage the data get annotated (and analyzed) in a different way. For example, we have a “presentation” and a “analysis” database.

The checking of availability is done by data that is entered (in an automated way) into Facility Management Information Models which can perform verifications.

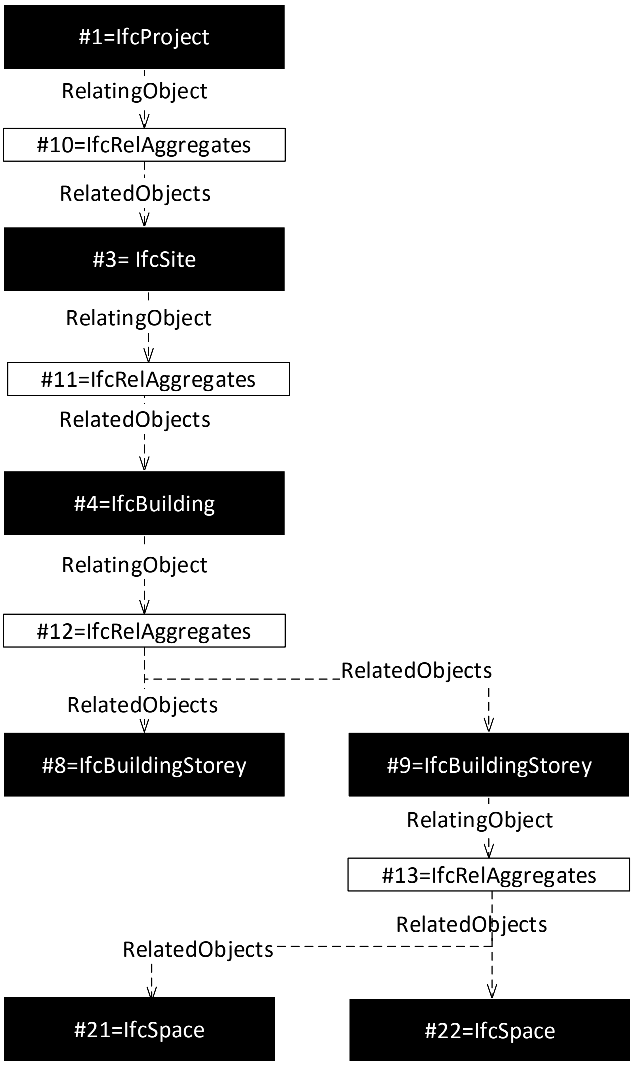
10. Does the business process require integration of data sets? If yes, is above mentioned way of storage methods sufficient to reuse/ integrate data together with other types of data sets?

Yes, according to our standards. We integrate data sets together during our analysis. However, I do not expect other companies are doing the same yet. By integrating data we can derive knowledge: After all, it are only data values what you actually are acquiring at first which you must give meaning in a later stadium.

11. To which extent do firms perform preventive / predictive maintenance?

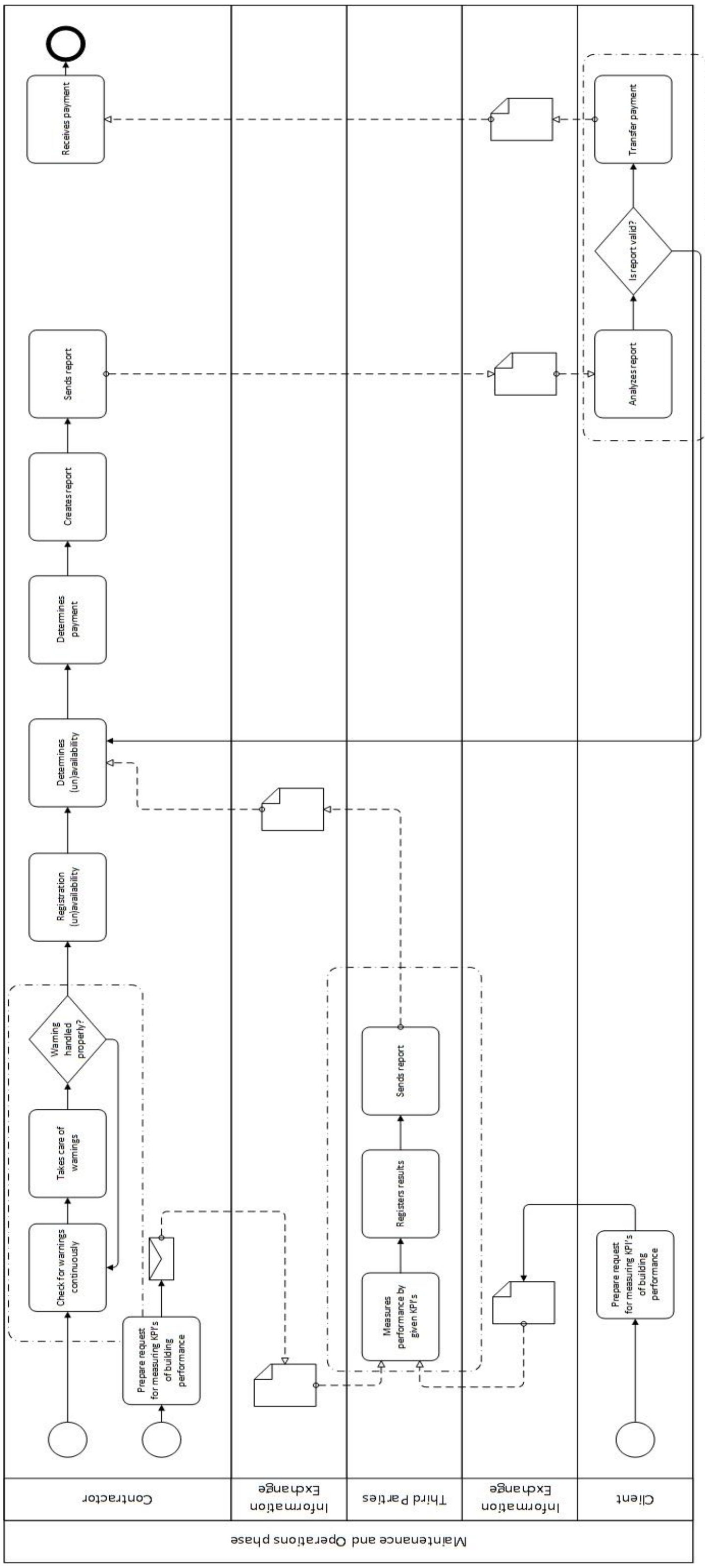
We perform predictive performance, based upon key figures and statistic models. Furthermore, we use some form of machine learning principles.

Appendix H An IfcSpatialStructureElement decomposition



A visualization of the breakdown structure of IfcSpatialStructureElement.

Appendix I The monitoring process within in a DBFMO- project


Maintenance and Operations phase
Contractor
Information Exchange
Third Parties
Information Exchange
Client
Prepare request for measuring KPI's of building performance
Check for warnings continuously
Takes care of warnings
Warning handled properly?
Registration (un)availability
Determines (un)availability
Determines payment
Creates report
Sends report
Receives payment
Measures performance by given KPI's
Registers results
Sends report
Prepare request for measuring KPI's of building performance
Analyzes report
Is report valid?
Transfer payment

A BPMN- visualization of the monitoring process within a DBFMO context.

Appendix J Descriptive analysis of sensor temperature data

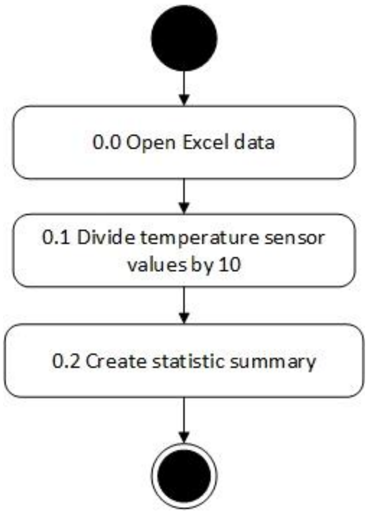

0.0 Open Excel data
0.1 Divide temperature sensor values by 10
0.2 Create statistic summary

Fig J1 A simplified flow chart which explains the undertaken steps to create a statistic summary of the temperature sensor values. Thereby, the number refers to the actual script.

	Count	Mean (°C)	Std (°C)	Min (°C)	25% (°C)	50% (°C)	75% (°C)	Max (°C)
Krijgsmachtbrede themaruimte introductie	8700	20.00	0.29	19.10	19.80	20.00	20.20	20.70
Krijgsmachtbrede themaruimte 1	8700	20.08	0.39	19.00	19.80	20.00	20.30	21.30
Krijgsmachtbrede themaruimte 2	8700	20.09	0.33	19.20	19.90	20.10	20.30	21.10
Krijgsmachtbrede themaruimte pronkzaal	8700	20.15	0.27	19.40	20.00	20.20	20.30	20.90
Krijgsmachtbrede themaruimte 3	8700	20.16	0.40	19.20	19.90	20.10	20.40	21.30
Krijgsmachtbrede themaruimte 4	8700	19.78	0.26	19.00	19.60	19.80	20.00	20.40
Krijgsmachtbrede themaruimte 5	8700	20.17	0.38	19.30	19.90	20.20	20.40	21.40
Krijgsmachtbrede themaruimte 6	8700	19.08	0.38	17.80	18.90	19.10	19.30	20.10
Average		**19.94**	**.34**	**19.**	**19.73**	**19.94**	**20.15**	**20.9**

Fig. J2 A descriptive analysis of the main characteristics of each room

Appendix K SE information models

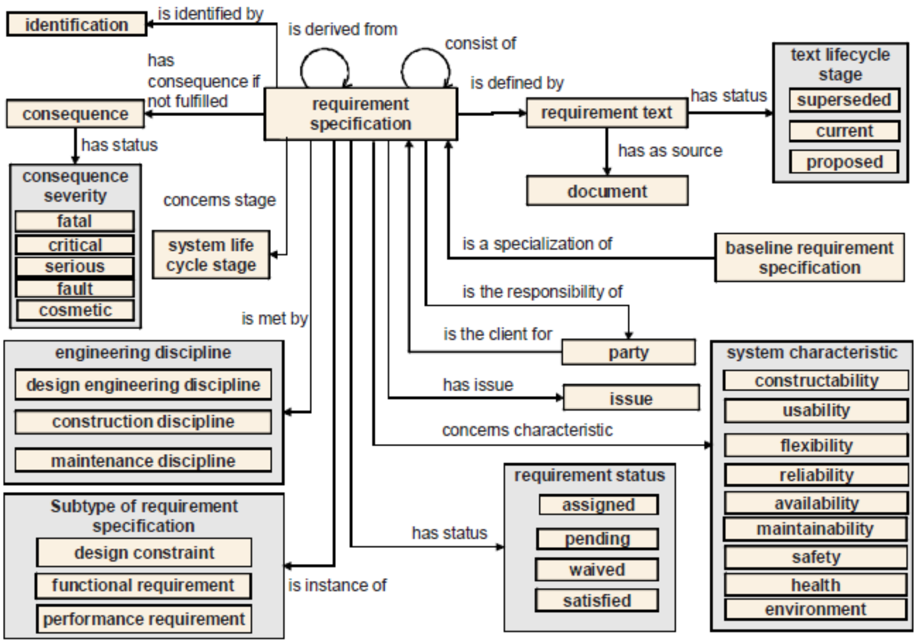


Fig. K1 A SE information model of a requirement specification

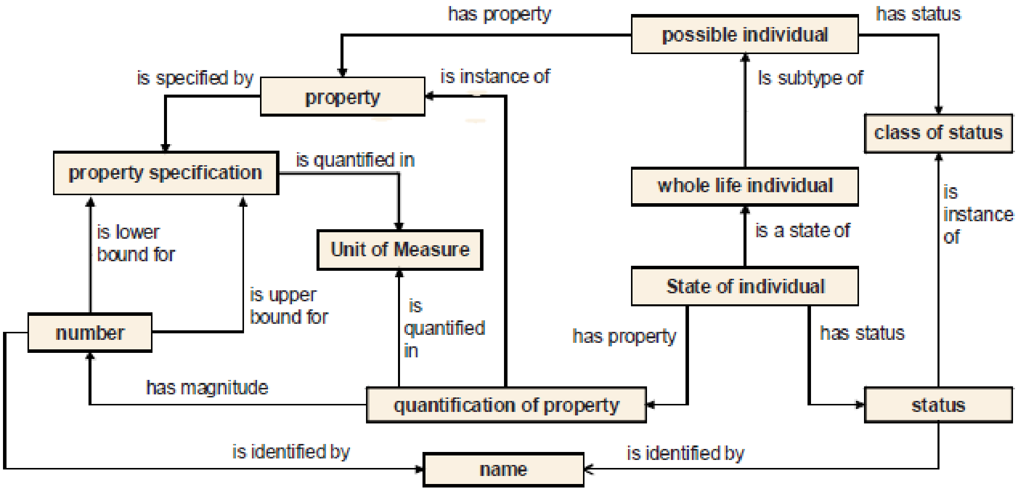


Fig. K2 A generic SE information model for all identified classes

Appendix L A SE ontology

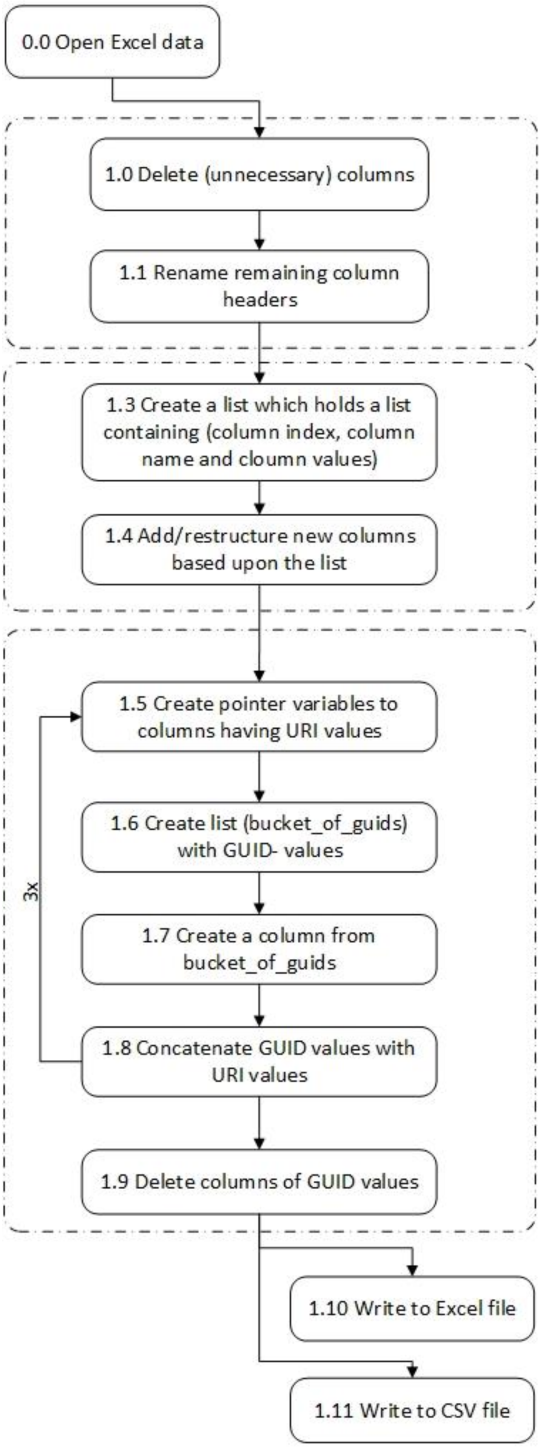
```
<?xml version="1.0"?>

<!DOCTYPE rdf:RDF [
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY owl2xml "http://www.w3.org/2006/12/owl2-xml#" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
    <!ENTITY Ontology1460824241639
"http://www.semanticweb.org/rakesh/ontologies/2016/3/Ontology1460824241639.
owl#" >
]>

<rdf:RDF
xmlns="http://www.semanticweb.org/rakesh/ontologies/2016/3/Ontology14608242
41639.owl#"

xml:base="http://www.semanticweb.org/rakesh/ontologies/2016/3/Ontology14608
24241639.owl"
     xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"

xmlns:Ontology1460824241639="http://www.semanticweb.org/rakesh/ontologies/2
016/3/Ontology1460824241639.owl#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:owl="http://www.w3.org/2002/07/owl#">
    <owl:Ontology rdf:about=""/>

    <owl:ObjectProperty rdf:about="#categorizedBy">
        <rdfs:range rdf:resource="#Condition"/>
        <rdfs:domain rdf:resource="#Requirement"/>
        <rdfs:subPropertyOf rdf:resource="#topObjectProperty"/>
    </owl:ObjectProperty>


    <owl:ObjectProperty rdf:about="#responsibilityFor">
        <rdfs:range rdf:resource="#Organization"/>
        <rdfs:domain rdf:resource="#Requirement"/>
        <rdfs:subPropertyOf rdf:resource="#topObjectProperty"/>
    </owl:ObjectProperty>

    <owl:ObjectProperty rdf:about="#specifies">
        <rdfs:range rdf:resource="#Condition"/>
        <rdfs:domain rdf:resource="#Requirement"/>
        <rdfs:subPropertyOf rdf:resource="#topObjectProperty"/>
    </owl:ObjectProperty>

    <owl:ObjectProperty rdf:about="#specifiesBoundaries">
        <rdfs:range rdf:resource="#QuantityValue"/>
        <rdfs:domain rdf:resource="#Requirement"/>
        <rdfs:subPropertyOf rdf:resource="#topObjectProperty"/>
    </owl:ObjectProperty>

    <owl:ObjectProperty rdf:about="#topObjectProperty"/>

    <owl:DatatypeProperty rdf:about="#hasUnit">
        <rdfs:subPropertyOf rdf:resource="#topDataProperty"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>
    <owl:DatatypeProperty rdf:about="#hasValue">
        <rdfs:subPropertyOf rdf:resource="#topDataProperty"/>
        <rdfs:range rdf:resource="&xsd;float"/>
    </owl:DatatypeProperty>

    <owl:DatatypeProperty rdf:about="#lowerboundary">
        <rdfs:domain rdf:resource="#QuantityValue"/>
        <rdfs:subPropertyOf rdf:resource="#topDataProperty"/>
        <rdfs:range rdf:resource="&xsd;float"/>
    </owl:DatatypeProperty>

    <owl:DatatypeProperty rdf:about="#topDataProperty"/>

    <owl:DatatypeProperty rdf:about="#unit">
        <rdfs:domain rdf:resource="#QuantityValue"/>
        <rdfs:subPropertyOf rdf:resource="#topDataProperty"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>

    <owl:DatatypeProperty rdf:about="#upperBoundary">
        <rdfs:domain rdf:resource="#QuantityValue"/>
        <rdfs:subPropertyOf rdf:resource="#topDataProperty"/>
        <rdfs:range rdf:resource="&xsd;float"/>
    </owl:DatatypeProperty>

    <owl:Class rdf:about="#Condition">
        <rdfs:subClassOf rdf:resource="&owl;Thing"/>
    </owl:Class>

    <owl:Class rdf:about="#Organization">
        <rdfs:subClassOf rdf:resource="&owl;Thing"/>
    </owl:Class>

    <owl:Class rdf:about="#QuantityValue">
        <rdfs:subClassOf rdf:resource="&owl;Thing"/>
    </owl:Class>

    <owl:Class rdf:about="#Requirement">
        <rdfs:subClassOf rdf:resource="&owl;Thing"/>
    </owl:Class>

    <owl:Class rdf:about="#Room">
        <rdfs:subClassOf rdf:resource="&owl;Thing"/>
    </owl:Class>

    <owl:Class rdf:about="&owl;Thing"/>
</rdf:RDF>
```

Appendix M A partial building RDF- graph

```
?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
   xmlns:express="http://purl.org/voc/express#"
   xmlns:ifcowl="http://www.buildingsmart-tech.org/ifcOWL/IFC2X3_TC1#"
   xmlns:list="http://www.co-ode.org/ontologies/list.owl#"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
>
  <rdf:Description rdf:about="http://linkedbuildingdata.net/ifc/resources20160610_001315/IfcLocalPlacement_85447">
    <ifcowl:relativePlacement_IfcLocalPlacement
rdf:resource="http://linkedbuildingdata.net/ifc/resources20160610_001315/IfcAxis2Placement3D_110625"/>
    <rdf:type rdf:resource="http://www.buildingsmart-tech.org/ifcOWL/IFC2X3_TC1#IfcLocalPlacement"/>
    <ifcowl:placementRelTo_IfcLocalPlacement
rdf:resource="http://linkedbuildingdata.net/ifc/resources20160610_001315/IfcLocalPlacement_110222"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://linkedbuildingdata.net/ifc/resources20160610_001315/IfcLengthMeasure_List_93304">
    <list:hasContents rdf:resource="http://linkedbuildingdata.net/ifc/resources20160610_001315/IfcLengthMeasure_62800"/>
    <list:hasNext rdf:resource="http://linkedbuildingdata.net/ifc/resources20160610_001315/IfcLengthMeasure_List_93305"/>
    <rdf:type rdf:resource="http://www.buildingsmart-tech.org/ifcOWL/IFC2X3_TC1#IfcLengthMeasure_List"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://linkedbuildingdata.net/ifc/resources20160610_001315/IfcAxis2Placement3D_110390">
    <ifcowl:location_IfcPlacement rdf:resource="http://linkedbuildingdata.net/ifc/resources20160610_001315/IfcCartesianPoint_84856"/>
    <rdf:type rdf:resource="http://www.buildingsmart-tech.org/ifcOWL/IFC2X3_TC1#IfcAxis2Placement3D"/>
    <ifcowl:refDirection_IfcAxis2Placement3D
rdf:resource="http://linkedbuildingdata.net/ifc/resources20160610_001315/IfcDirection_47672"/>
    <ifcowl:axis_IfcAxis2Placement3D rdf:resource="http://linkedbuildingdata.net/ifc/resources20160610_001315/IfcDirection_47670"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://linkedbuildingdata.net/ifc/resources20160610_001315/IfcAxis2Placement3D_28886">
    <ifcowl:axis_IfcAxis2Placement3D rdf:resource="http://linkedbuildingdata.net/ifc/resources20160610_001315/IfcDirection_2547"/>
    <rdf:type rdf:resource="http://www.buildingsmart-tech.org/ifcOWL/IFC2X3_TC1#IfcAxis2Placement3D"/>
    <ifcowl:refDirection_IfcAxis2Placement3D
rdf:resource="http://linkedbuildingdata.net/ifc/resources20160610_001315/IfcDirection_2549"/>
    <ifcowl:location_IfcPlacement rdf:resource="http://linkedbuildingdata.net/ifc/resources20160610_001315/IfcCartesianPoint_28880"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://linkedbuildingdata.net/ifc/resources20160610_001315/IfcRepresentation_List_69693">
    <rdf:type rdf:resource="http://www.buildingsmart-tech.org/ifcOWL/IFC2X3_TC1#IfcRepresentation_List"/>
    <list:hasContents rdf:resource="http://linkedbuildingdata.net/ifc/resources20160610_001315/IfcShapeRepresentation_25910"/>
  </rdf:Description>
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
<rdf:Description rdf:about="http://linkedbuildingdata.net/ifc/resources20160610_001315/IfcExtrudedAreaSolid_28135">
    <ifcowl:extrudedDirection_IfcExtrudedAreaSolid
rdf:resource="http://linkedbuildingdata.net/ifc/resources20160610_001315/IfcDirection_19"/>
    <rdf:type rdf:resource="http://www.buildingsmart-tech.org/ifcOWL/IFC2X3_TC1#IfcExtrudedAreaSolid"/>
    <ifcowl:depth_IfcExtrudedAreaSolid
rdf:resource="http://linkedbuildingdata.net/ifc/resources20160610_001315/IfcPositiveLengthMeasure_59185"/>
    <ifcowl:position_IfcSweptAreaSolid
rdf:resource="http://linkedbuildingdata.net/ifc/resources20160610_001315/IfcAxis2Placement3D_1801"/>
    <ifcowl:sweptArea_IfcSweptAreaSolid
rdf:resource="http://linkedbuildingdata.net/ifc/resources20160610_001315/IfcIShapeProfileDef_28133"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://linkedbuildingdata.net/ifc/resources20160610_001315/IfcSpace_346">
    <ifcowl:representation_IfcProduct
rdf:resource="http://linkedbuildingdata.net/ifc/resources20160610_001315/IfcProductDefinitionShape_344"/>
    <ifcowl:longName_IfcSpatialStructureElement
rdf:resource="http://linkedbuildingdata.net/ifc/resources20160610_001315/IfcLabel_58277"/>
    <ifcowl:compositionType_IfcSpatialStructureElement rdf:resource="http://www.buildingsmart-tech.org/ifcOWL/IFC2X3_TC1#ELEMENT"/>
    <ifcowl:name_IfcRoot rdf:resource="http://linkedbuildingdata.net/ifc/resources20160610_001315/IfcLabel_58276"/>
    <ifcowl:globalId_IfcRoot rdf:resource="http://linkedbuildingdata.net/ifc/resources20160610_001315/IfcGloballyUniqueId_58275"/>
    <ifcowl:interiorOrExteriorSpace_IfcSpace rdf:resource="http://www.buildingsmart-tech.org/ifcOWL/IFC2X3_TC1#INTERNAL"/>
    <ifcowl:ownerHistory_IfcRoot rdf:resource="http://linkedbuildingdata.net/ifc/resources20160610_001315/IfcOwnerHistory_41"/>
    <ifcowl:objectPlacement_IfcProduct
rdf:resource="http://linkedbuildingdata.net/ifc/resources20160610_001315/IfcLocalPlacement_333"/>
    <rdf:type rdf:resource="http://www.buildingsmart-tech.org/ifcOWL/IFC2X3_TC1#IfcSpace"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://linkedbuildingdata.net/ifc/resources20160610_001315/IfcMember_62024">
    <ifcowl:globalId_IfcRoot rdf:resource="http://linkedbuildingdata.net/ifc/resources20160610_001315/IfcGloballyUniqueId_85304"/>
    <ifcowl:representation_IfcProduct
rdf:resource="http://linkedbuildingdata.net/ifc/resources20160610_001315/IfcProductDefinitionShape_62014"/>
```

```
  <rdf:type rdf:resource="http://www.buildingsmart-tech.org/ifcOWL/IFC2X3_TC1#IfcMember"/>
  <ifcowl:name_IfcRoot rdf:resource="http://linkedbuildingdata.net/ifc/resources20160610_001315/IfcLabel_85305"/>
  <ifcowl:objectPlacement_IfcProduct
rdf:resource="http://linkedbuildingdata.net/ifc/resources20160610_001315/IfcLocalPlacement_62023"/>
  <ifcowl:ownerHistory_IfcRoot rdf:resource="http://linkedbuildingdata.net/ifc/resources20160610_001315/IfcOwnerHistory_41"/>
  <ifcowl:objectType_IfcObject rdf:resource="http://linkedbuildingdata.net/ifc/resources20160610_001315/IfcLabel_59179"/>
  <ifcowl:tag_IfcElement rdf:resource="http://linkedbuildingdata.net/ifc/resources20160610_001315/IfcIdentifier_85306"/>
 </rdf:Description>
 <rdf:Description rdf:about="http://linkedbuildingdata.net/ifc/resources20160610_001315/IfcRelDefinesByProperties_59562">
  <ifcowl:globalId_IfcRoot rdf:resource="http://linkedbuildingdata.net/ifc/resources20160610_001315/IfcGloballyUniqueId_84252"/>
  <rdf:type rdf:resource="http://www.buildingsmart-tech.org/ifcOWL/IFC2X3_TC1#IfcRelDefinesByProperties"/>
  <ifcowl:ownerHistory_IfcRoot rdf:resource="http://linkedbuildingdata.net/ifc/resources20160610_001315/IfcOwnerHistory_41"/>
  <ifcowl:relatingPropertyDefinition_IfcRelDefinesByProperties
rdf:resource="http://linkedbuildingdata.net/ifc/resources20160610_001315/IfcPropertySet_59560"/>
  <ifcowl:relatedObjects_IfcRelDefines rdf:resource="http://linkedbuildingdata.net/ifc/resources20160610_001315/IfcMember_59556"/>
 </rdf:Description>
 <rdf:Description rdf:about="http://linkedbuildingdata.net/ifc/resources20160610_001315/IfcLocalPlacement_57497">
  <ifcowl:placementRelTo_IfcLocalPlacement
rdf:resource="http://linkedbuildingdata.net/ifc/resources20160610_001315/IfcLocalPlacement_102494"/>
  <rdf:type rdf:resource="http://www.buildingsmart-tech.org/ifcOWL/IFC2X3_TC1#IfcLocalPlacement"/>
  <ifcowl:relativePlacement_IfcLocalPlacement
rdf:resource="http://linkedbuildingdata.net/ifc/resources20160610_001315/IfcAxis2Placement3D_102855"/>
 </rdf:Description>
```

Appendix N Flowcharts for data transformation and conversion


0.0 Open Excel data
1.0 Delete (unnecessary) columns
1.1 Rename remaining column headers
1.3 Create a list which holds a list containing (column index, column name and cloumn values)
1.4 Add/restructure new columns based upon the list
1.5 Create pointer variables to columns having URI values
1.6 Create list (bucket_of_guids) with GUID- values
1.7 Create a column from bucket_of_guids
1.8 Concatenate GUID values with URI values
3x
1.9 Delete columns of GUID values
1.10 Write to Excel file
1.11 Write to CSV file

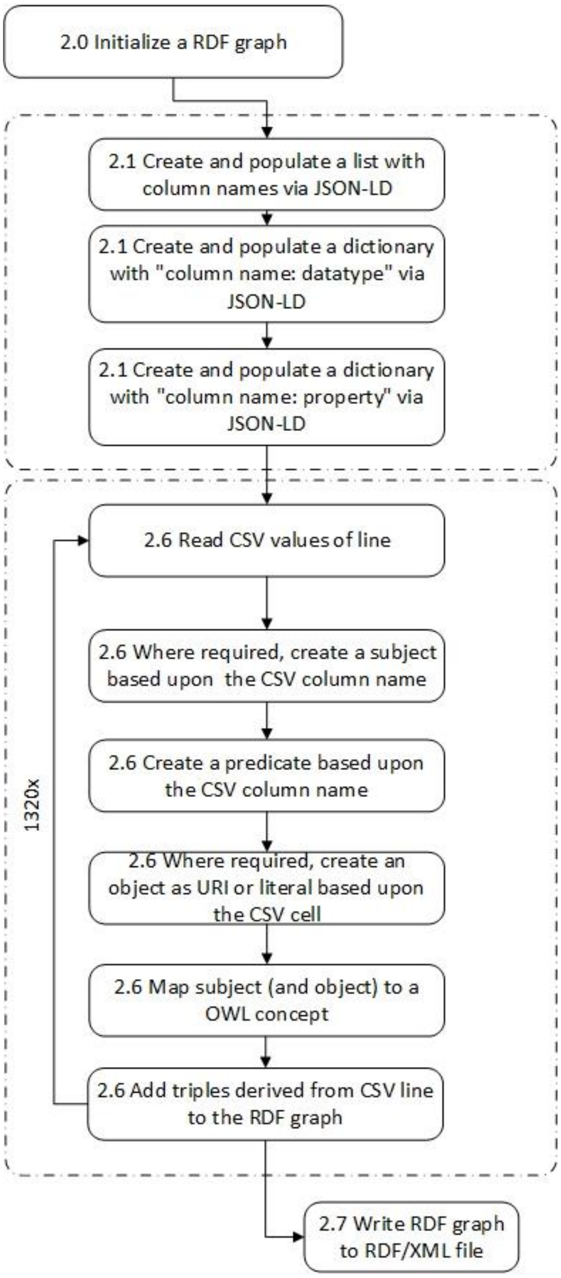

2.0 Initialize a RDF graph
2.1 Create and populate a list with column names via JSON-LD
2.1 Create and populate a dictionary with "column name: datatype" via JSON-LD
2.1 Create and populate a dictionary with "column name: property" via JSON-LD
2.6 Read CSV values of line
2.6 Where required, create a subject based upon the CSV column name
2.6 Create a predicate based upon the CSV column name
2.6 Where required, create an object as URI or literal based upon the CSV cell
2.6 Map subject (and object) to a OWL concept
2.6 Add triples derived from CSV line to the RDF graph
1320x
2.7 Write RDF graph to RDF/XML file

Fig. N1 A simplified flow chart which depicts cleansing and transformation procedures
The numbers refer parts of the programming code. Thereby, the number refers to the actual script.

Fig. N2 A simplified flow chart which depicts the CSV to RDF conversion. The numbers refer parts of the programming code. Thereby, the number refers to the actual script.

Appendix O A script for sensor data transformation

```
import pandas as pd
import uuid
from pandas import DataFrame

import datetime
import pandas.io.data
from scipy.stats.mstats import mode
import numpy as np
import matplotlib.pyplot as plt

import glob

#0.0 Read Excel
df = pd.read_excel(r'INTRO.xlsx', sheetname = "INTRO")


#0.1 Divide values
# df["Waarde"] = df["Waarde"].values/10.0

#0.2 Create statistic summary
# statistic_summary = df.describe()

#1.0 Delete columns
df = df.drop(["AangevuldeData", "DataGemist","IntervalGewijzigd"], axis=1)

#1.1 Rename Excel- column headers
headers = {"Systeemtijd":"inXSDDateTime", "Waarde":"numericValue"}

#1.2 Rename column headers
for old, new in headers.iteritems():
    df=df.rename(columns = {old:new})

#1.3 Create a list which holds lists holding: column index, column header, column
values (URIS)
ssn_table = []
ssn_table.append([0, "observedBy", "http://example.com/id/Sensor/SensorName"])
ssn_table.append([1, "hasLocation", "http://example.com/id/PhysicalPlace/RoomName"])
ssn_table.append([2, "observationSamplingTime", "http://example.com/id/Instant/"])
ssn_table.append([4, "observationResult", "http://example.com/id/SensorOutput/"])
ssn_table.append([5, "hasValue", "http://example.com/id/ObservationValue/"])
ssn_table.append([7 , "unit", "http://example.com/id/Unit/CelsiusDegrees"])
ssn_table.append([8 , "observedProperty", "http://dbpedia.org/resource/temperature"])
ssn_table.append([9, "featureOfInterest", "http://dbpedia.org/resource/air"])

#1.4 Add columns to Excel file
for x in ssn_table:
    index = x[0]
    column_header = x[1]
    value = x[2]
    df.insert(index, column_header, value)

#1.5 Create pointer variables to columns having URI's
column_indices = [2, 3, 4]

#Loop 3 times whereby the following actions are performed
    #1.6 Creation of a list holding random GUID values
    #1.7 Creation of a column from this list
    #1.8 Concatenation of the URIs with the GUIDs
for number in range(3):
    column_list_index = column_indices[number]
    column_header = ssn_table[column_list_index][1]

    sLength = len(df['observedBy'])
    bucket_of_guids = []
    del bucket_of_guids[:]

    for xx in range(sLength):
        guid = str(uuid.uuid1())
        bucket_of_guids.append((guid))

    df[str(number)] = pd.Series(bucket_of_guids, index=df.index)
    df[column_header] = df[column_header].map(str) + df[str(number)]

# 1.9 Deletion of the columns holding the GUIDS
df = df.drop(["0", "1", "2"], axis=1)

#1.10 Write to new Excel file
writer = pd.ExcelWriter('T6_OUTPUT.xlsx')
df.to_excel(writer, sheet_name='Sheet1', index = False)
writer.save()

#1.11 Write to new CSV file
df.to_csv("PRO_OUTPUT.csv",  index = False)


#NOTE: A check if every read Excel contains the same number of rows (1320)
# row_count = len(df.index)
# print (row_count)
```

Appendix P A script for SE data transformation

```
import pandas as pd
import uuid

import glob

#0.0 Read Excel
df = pd.read_excel(r'se_requirements_definitief.xlsx', sheetname =
"se_requirements_definitief")

#Loop 3 times whereby the following actions are performed
    #0.1 Creation of a list holding random GUID values
    #0.2 Creation of a column from this list
    #0.3 Concatenation of the URIs with the GUIDs
bucket_of_guids = []
sLength = len(df['specifies'])
for xx in range(sLength):
    guid = str(uuid.uuid1())
    bucket_of_guids.append((guid))
df["guids"] = pd.Series(bucket_of_guids, index=df.index)
df["specifiesBoundaries"] = df["specifiesBoundaries"].map(str) + df["guids"]

#0.4 Deletion of the columns holding the GUIDS
df = df.drop(["guids"], axis=1)

#0.5 Write to new Excel file
writer = pd.ExcelWriter('se_requirements_definitief_OUTPUT.xlsx')
df.to_excel(writer, sheet_name='Sheet1', index = False)
writer.save()

#0.6 Write to new CSV file
df.to_csv("se_requirements_definitief_OUTPUT.csv",  index = False)
```

Appendix Q Partial tabular data transformation results

observedBy	hasLocation
http://example.com/id/Sensor/OS12GRFMET115	http://example.com/id/PhysicalPlace/KrijgsmachtbredeThemaruimteIntroductie
http://example.com/id/Sensor/OS12GRFMET115	http://example.com/id/PhysicalPlace/KrijgsmachtbredeThemaruimteIntroductie

observationSamplingTime	inXSDDateTime
http://example.com/id/Instant/338d0dae-1d9a-11e6-83f6-240a64020db4	2016-02-01T00:04:00
http://example.com/id/Instant/338d34c0-1d9a-11e6-a84e-240a64020db4	2016-02-01T00:12:00

observationResult	hasValue
http://example.com/id/SensorOutput/33976df0-1d9a-11e6-932c-240a64020db4	http://example.com/id/ObservationValue/339ff970-1d9a-11e6-9379-240a64020db4
http://example.com/id/SensorOutput/33976df0-1d9a-11e6-9eca-240a64020db4	http://example.com/id/ObservationValue/33a02080-1d9a-11e6-8215-240a64020db4

numericValue	unit
20.1	http://example.com/id/Unit/CelsiusDegrees
20.1	http://example.com/id/Unit/CelsiusDegrees

observedProperty	featureOfInterest
http://dbpedia.org/resource/temperature	http://dbpedia.org/resource/air
http://dbpedia.org/resource/temperature	http://dbpedia.org/resource/air

Fig. Q1 A part of the transformed (and enriched) sensor data set which now can be used for the data conversion.

specifies	categorizedBy
http://example.com/id/Room/Intro-Experience	http://example.com/id/Condition/ThermischComfortVerblijfsruimten
http://example.com/id/Room/Hoofdthema1NederlandEnDeWereld	http://example.com/id/Condition/ThermischComfortVerblijfsruimten
http://example.com/id/Room/Hoofdthema2DeWereldVanDeKrijgsmacht	http://example.com/id/Condition/ThermischComfortVerblijfsruimten
http://example.com/id/Room/Pronkzaal	http://example.com/id/Condition/ThermischComfortVerblijfsruimten
http://example.com/id/Room/Hoofdthema3MilitairenInDeSchijnwerpers	http://example.com/id/Condition/ThermischComfortVerblijfsruimten
http://example.com/id/Room/Hoofdthema4DeWereldVanDeTechniek	http://example.com/id/Condition/ThermischComfortVerblijfsruimten

specifiesBoundaries	Lower Boundary	Upper Boundary	unit	responsibilityFor
http://example.com/id/QuantityValue/1ac88651-1a94-11e6-888b-240a64020db4	18	25	Celcius	http://example.com/id/Organization/Heijmans
http://example.com/id/QuantityValue/1ac88651-1a94-11e6-8754-240a64020db4	18	25	Celcius	http://example.com/id/Organization/Heijmans
http://example.com/id/QuantityValue/1ac88651-1a94-11e6-be97-240a64020db4	18	25	Celcius	http://example.com/id/Organization/Heijmans
http://example.com/id/QuantityValue/1ac88651-1a94-11e6-b309-240a64020db4	18	25	Celcius	http://example.com/id/Organization/Heijmans
http://example.com/id/QuantityValue/1ac88651-1a94-11e6-8a89-240a64020db4	18	25	Celcius	http://example.com/id/Organization/Heijmans
http://example.com/id/QuantityValue/1ac88651-1a94-11e6-b384-240a64020db4	18	25	Celcius	http://example.com/id/Organization/Heijmans

Fig. Q2 A part of the the transformed (and enriched) SE data set which now can be used for the data conversion.

Appendix R JSON Linked Data for sensor data conversion

```
{
  "@id" : "http://semmtech.nl/ssn/csvw.csv",
  "@context": ["http://www.w3.org/ns/csvw",
     {
       "@language": "en",
       "xsd": "http://www.w3.org/2001/XMLSchema#",
       "dcterms": "http://purl.org/dc/terms/",
       "ssn":"http://purl.oclc.org/NET/ssnx/ssn#",
       "time": "http://www.w3.org/2006/time#",
       "qudt":"http://qudt.org/schema/qudt#",
       "dul": "http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#"
     }
  ],
  "delimiter": ";",
  "@type":["Table","dcat:DataSet"],
  "url": "http://semmtech.nl/ssn/csvw.csv",
  "dcterms:title": "SSN sample",
  "dcterms:description": "A ssn sample table",
  "dcterms:keywords": ["ssn","sample"],
  "dcterms:modified": "2016-04-17",

   "tableSchema": {
      "columns": [
      {
        "name": "observedBy",
        "title": "observedBy",
        "dcterms:description": "A sensor id",
        "propertyUrl": "ssn:observedBy",
        "required": true
      },
      {
        "name": "hasLocation",
        "title": "hasLocation",
        "dcterms:description": "The room where the sensor is located",
        "propertyUrl": "ssn:hasLocation",
        "required": true
      },
      {
        "name": "observationSamplingTime",
        "title": "observationSamplingTime",
        "dcterms:description": "A timestamp id of an observation",
        "propertyUrl": "ssn:observationSamplingTime",
        "required": true
      },
      {
        "name": "inXSDDateTime",
        "title": "inXSDDateTime",
        "dcterms:description": "The numerical timestamp of an observation",
        "datatype": "xsd:dateTime",
        "propertyUrl": "time:inXSDDateTime",
        "required": true
      },
      {
        "name": "observationResult",
        "title": "observationResult",
        "dcterms:description": "The result of an observation",
        "propertyUrl": "ssn:observationResult",
        "required": true
      },
      {
        "name": "hasValue",
        "title": "hasValue",
        "dcterms:description": "The pointer to the URI which holds the numerical value of an observation",
        "propertyUrl": "ssn:hasValue",
        "required": true
      },
      {
        "name": "numericValue",
        "title": "numericValue",
        "dcterms:description": "The numerical value of an observation",
        "datatype": "xsd:double",
        "propertyUrl": "qudt:numericValue",
        "required": true
      },
      {
        "name": "unit",
        "title": "unit",
        "dcterms:description": "The unit in which an observation is expressed",
        "propertyUrl": "qudt:unit",
        "required": true
      },
      {
        "name": "observedProperty",
        "title": "observedProperty",
        "dcterms:description": "The observed property of a feature of interest.",
        "propertyUrl": "ssn:observedProperty",
        "required": true
      },
      {
        "name": "featureOfInterest",
        "title": "featureOfInterest",
        "dcterms:description": "The feature of an observed property",
        "propertyUrl": "ssn:featureOfInterest",
        "required": true
      }],
      "primaryKey": ["observationSamplingTime"],

"aboutUrl":"http://semmtech.nl/ssn/csvw.csv/Observation.{_row}"
    }
  }
```

Appendix S JSON Linked Data for SE data conversion

```
{
   "@id" : "http://semmtech.nl/se/csvw.csv",
   "@context": ["http://www.w3.org/ns/csvw",
      {
         "@language": "en",
         "xsd" : "http://www.w3.org/2001/XMLSchema#",
         "dcterms" : "http://purl.org/dc/terms/",
         "se":"http://semmtech.nl/se/ontology/"
      }
   ],
   "delimiter": ";",
   "@type":["Table","dcat:DataSet"],
   "url": "http://semmtech.nl/se/csvw.csv",
   "dcterms:title": "SE sample",
   "dcterms:description": "A SE sample table",
   "dcterms:keywords": ["Systems Engineering","sample"],
   "dcterms:modified": "2016-04-19",

    "tableSchema": {
        "columns": [
        {
          "name": "categorizedBy",
          "title": "categorizedBy",
          "dcterms:description": "A certain condition mentioned in
a program of requirements",
          "propertyUrl": "se:categorizedBy",
          "required": true
        },

        {
          "name": "specifies",
          "title": "specifies",
          "dcterms:description": "Points to a room",
          "propertyUrl": "se:specifies",
          "required": true
        },

        {
          "name": "specifiesBoundaries",
          "title": "specifiesBoundaries",
          "dcterms:description": "Points to the boundaries of a
requirement",
          "propertyUrl": "se:specifiesBoundaries",
          "required": true
        },

        {
          "name": "lowerBoundary",
          "title": "lowerBoundary",
          "dcterms:description": "Points to the numerical value of a
           lower bound",
          "datatype": "xsd:double",
          "propertyUrl": "se:lowerBoundary",
          "required": true
        },

       {
          "name": "upperBoundary",
          "title": "upperBoundary",
          "dcterms:description": "Points to the numerical value of a
            bound",
          "datatype": "xsd:double",
          "propertyUrl": "se:upperBoundary",
          "required": true
        },

        {
          "name": "unit",
          "title": "unit",
          "dcterms:description": "The unit of the bounds",
          "datatype": "xsd:string",
          "propertyUrl": "se:unit",
          "required": true
        },

        {
          "name": "responsibilityFor",
          "title": "responsibilityFor",
          "dcterms:description": "The party which is responsible
           for compliance of a condition to a requirement",
          "propertyUrl": "se:responsibilityFor",
          "required": true
        }

        ],
        "primaryKey": ["Requirement"],
        "aboutUrl":"http://example.com/id/Requirement/{_row}"
      }
    }
```

Appendix T A script for sensor data conversion

```
from rdflib import Graph, Literal, URIRef, Namespace,OWL, XSD, RDF, RDFS, BNode
import csv
import urllib
import re
import uuid
from operator import itemgetter
import sys

from rdflib import Graph, plugin
from rdflib.serializer import Serializer

MAX_LINES_TO_PROCESS = -1

CSVW = Namespace("http://www.w3.org/ns/csvw#")
DCAT = Namespace("http://www.w3.org/ns/dcat#")
DC = Namespace("http://purl.org/dc/terms/")
SSN = Namespace("http://purl.oclc.org/NET/ssnx/ssn#")
TIME = Namespace("http://www.w3.org/2006/time#")
QUDT = Namespace("http://qudt.org/schema/qudt#")
DUL = Namespace("http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#")

class CSVWtoRDF:
    '''
    Provides operation to add triples extracted from CSV files using an implementation
of a subset of the
    CSV on the Web standard to a provided Graph.
    Assumptions on the input files:
     * CSVW meta file is provided as JSON-LD serialisation
     * if not specified otherwise, loadCSW() looks for the CSVW meta file for
'/path/to/input.csv' at
        '/path/to/input.csv.csvw'
     * the CSVW meta description RDF graph contains exactly one resource where the a
suffix of the IRI is identical
        to the filename of the provided CSV input and this resource is the 'root' of
the CSVW mapping description
        (otherwise, provide the resource name explicitly using the mappingResouceIRI
parameter of the constructor)
    * only the CSVW datatype definitins 'anyURI', 'string' and 'double' are supported
at the moment
    creators: Brian Walshe (Trinity College Dublin, KDEG)
              Markus Ackermann (University Leipzig, AKSW)
    '''
    def __init__(self, rdfGraph):
        self.graph = rdfGraph

#Add ".csvw" to the .csv if no .csvw is provided
    def loadCSVW(self, csvFilename, csvwFilename=None, mappingResourceIRI=None):

        if csvwFilename == None:
            csvwFilename = csvFilename + ".csvw"

#2.0.0 Initialize a graph
        self.graph.parse(csvwFilename, format='json-ld')
#2.0.1 Create a meta- graph
        self.graph.serialize(open("metadata.rdf", "w"), "xml")

#2.0.2 Find IRI (=subject) of the JSDON-LD table
        tableNode = self.graph.value(predicate=RDF.type, object=CSVW.Table)
#2.0.3 Find the denoted csv delimiter in the json-ld table
        delim =  self.graph.value(tableNode, CSVW.delimiter)
```

```
#2.0.4 Read the csv data
        csvFile = open(csvFilename, "r")
        csvData = csv.reader(csvFile, delimiter=str(delim))
#2.0.5 Get the object which represent the characteristics of the json-ld model
        schemaRes = self.graph.value(tableNode, CSVW.tableSchema)
#2.0.6 Create triples from the json-ld that describe the csv data in general
        dcatNode = BNode()
        self.graph.add((dcatNode, RDF.type, DCAT.Distribution))
        self.graph.add((dcatNode, DCAT.downloadURL, URIRef(csvFilename)))
        self.graph.add((tableNode, DCAT.distribution, dcatNode))

#2.1.0 List the column names
        mappedColumnsNames = []
#2.1.1 List the columnnames together with their datatypes
        datatypeForColumn = dict()
#2.1.2 List the columnnames together with their URI (propertyUrl)
        propertyForColumn = dict()
        columnList =  self.graph.value(schemaRes, CSVW.column)
#2.1.3 set up for iteration through rdf:rest
        while columnList != None and self.graph.value(columnList, RDF.first)!=None:
#2.1.4 Get the column description (via rdf:first)
            column = self.graph.value(columnList, RDF.first)
#2.1.5 Get the columName from the column description
            columnName = self.graph.value(column, CSVW.name)
#2.1.6 Get the propertyURL (URI) of the column description
            propertyRes = self.graph.value(column, CSVW.propertyUrl)
#2.1.7 State that the propertyURL (URI) is of RDF:property
            # self.graph.add((propertyRes, RDF.type, RDF.Property))
#2.1.8 Create a triple stating a rdfs:label
            if self.graph.value(column, CSVW.title) != None:
              self.graph.add((propertyRes, RDFS.label, self.graph.objects(column, 
CSVW.title)))
#2.1.9 Create a triple stating a dc:description
            if self.graph.value(column, DC.description) != None:
               self.graph.add((propertyRes, DC.description, self.graph.value(column, 
DC.description)))
#2.1.10 Adding values to the previous defined list/dictionaries as follows:
    #(1)add column name
    #(2)add pairs with columnname(name) and datatype (value)
    #(3)add column names using propertyURL
            mappedColumnsNames += columnName
            datatypeForColumn[str(columnName)] = str(self.graph.value(column, 
CSVW.datatype))
            propertyForColumn[str(columnName)] = propertyRes
#2.1.11 Go to the next column description (rdf:rest)
            columnList = self.graph.value(columnList, RDF.rest)

#2.2 Store aboutURL of the json-ld table into a variable
        urlTemplate = self.graph.value(schemaRes, CSVW.aboutUrl)

#2.3 Slice the URI in three parts
        groups = re.match("^(.*?)\{([A-Za-z0-9\-_]+)\}(.*)$", urlTemplate)
        # pre = http://semmtech.nl/ssn/csvw.csv/Observation
        pre = groups.group(1)
        # post = none
        post = groups.group(3)
        # nameCol = ".{_row}"
        nameCol = groups.group(2)

#2.4 Store all csv column headers in variable
        csvHeader = csvData.next()
```

```
#2.5 Map numerical csv column indexes to each csvHeader
        cellname2Index = self._cellToIndexMapping(csvHeader)
        idIndex=-1

#not relevant in this case
        if nameCol!="_row":
            idIndex = csvHeader.index(nameCol)

#2.6.0 Iterate over each observation in the csv data
        linesRead = 0
        for line in csvData:
            linesRead += 1
#2.6.1 Create a GUID for each observation
            id_fragment = str(uuid.uuid1())
            if idIndex!=-1:
                id_fragment = urllib.quote_plus(line[idIndex])
            subject = URIRef("%s%s%s" % (pre, id_fragment, post))
#2.6.2 Create row instance
            self.graph.add((tableNode, CSVW.row, subject))
#2.6.3 Map a row to the Observation concept
            self.graph.add((subject, RDF.type, SSN.Observation))
#2.6.4 Create a dictionary for mapping the data instances to a concept
            mapping = {
                        "Sensor": "http://purl.oclc.org/NET/ssnx/ssn#Sensor",
                        "Instant": "http://www.w3.org/2006/time#Instant",
                        "SensorOutput":
"http://purl.oclc.org/NET/ssnx/ssn#SensorOutput",
                        "ObservationValue":
"http://purl.oclc.org/NET/ssnx/ssn#ObservationValue",
                        "Temperatuur": "http://purl.oclc.org/NET/ssnx/ssn#Property",
                        "Q7391292":
"http://purl.oclc.org/NET/ssnx/ssn#FeatureOfInterest",
                        "PhysicalPlace" :
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#PhysicalPlace",
                        "Unit" : "http://data.nasa.gov/qudt/owl/unit#DegreeCelsius",
                       }
#2.6.5 For each cell value of a csv row, define a standard subject
            for cellname, i in cellname2Index.items():
                #print (cellname2Index.items())
                subject = URIRef("%s%s%s" % (pre, id_fragment, post))
#not relevant in this case
                if cellname not in propertyForColumn.keys():
                   #print "no hit for (%s)"%cellname
                   continue
#2.6.6 Redefine standard subject based upon csv column header
                if cellname == "hasValue":
#subject = URIRef(SSN.SensorOutput)
#subject = URIRef("%s/%s%s" % (SSN.SensorOutput, id_fragment, post))
#subject = self.makeObject(datatypeForColumn["observationResult"], line[2])
                    subject = URIRef(line[4])
                elif cellname == "numericValue":
                    subject = URIRef(line[5])
                elif cellname == "unit":
                    subject = URIRef(line[5])
                elif cellname == "inXSDDateTime":
                    subject = URIRef(line[2])
                elif cellname == "hasLocation":
                    subject = URIRef(line[0])
#2.6.7 Define a property based upon csv column header
                predicate = propertyForColumn[cellname]
```

```
#2.6.8 Get + define a object as a URI or literal
                if "/" in line[i]:
                    obj = URIRef(line[i])
                else:
                    obj = self.makeObject(datatypeForColumn[cellname], line[i])
                #if cellname == "hasValue":
                #    obj =  URIRef(line[4])
                #if cellname == "hasQuantityValue":
                #    obj = URIRef(line[5])
#2.6.9 Add a RDF- data instance to the graph
                self.graph.add((subject, predicate, obj))
#2.6.10 Check + Map a RDF-instance to a concept
                concept_URI = ""
                if "/" in str(obj):
                    concept = obj.split('/')[4]
                    if concept in mapping.keys():
                        concept_URI = (mapping.get(concept))
#2.6.11 Add a RDF-type to the graph
                        self.graph.add((URIRef(obj), RDF.type, URIRef(concept_URI)))
                        if concept == "ObservationValue":
                            self.graph.add((URIRef(obj), RDF.type,
URIRef(QUDT.QuantityValue)))
            if(MAX_LINES_TO_PROCESS > 0 and linesRead >= MAX_LINES_TO_PROCESS): break

    def _cellToIndexMapping(self, csvHeader):

        return dict(zip(csvHeader, range(0, len(csvHeader))))

#    def _sanitizeSID(self, sid):
#        return sid.replace('_', '-')

    def makeObject(self, datatypeStr, val):
        if datatypeStr == "anyURI":
            return URIRef(val)
        else:
            return Literal(val, datatype=datatypeStr)

    def printN3(self):
        print(self.graph.serialize(format='n3'))

    def writeToFile(self, fileName, format="xml"):
        self.graph.serialize(open(fileName, "w"), format)

#2.7 Write to a rdf file
g = Graph()
converter = CSVWtoRDF(g)
converter.loadCSVW("T5_INPUT_CSV_dummy_lower.csv")
converter.writeToFile("T5_INPUT_CSV_dummy_lower.rdf")
```

Appendix U A script for SE data conversion

```
from rdflib import Graph, Literal, URIRef, Namespace, XSD, RDF,OWL, RDFS, BNode
import csv
import urllib
import re
import uuid
from operator import itemgetter
import sys

from rdflib import Graph, plugin
from rdflib.serializer import Serializer

MAX_LINES_TO_PROCESS = -1

CSVW = Namespace("http://www.w3.org/ns/csvw#")
DCAT = Namespace("http://www.w3.org/ns/dcat#")
DC = Namespace("http://purl.org/dc/terms/")
SE = Namespace("http://semmtech.nl/se/ontology/")
class CSVWtoRDF:
    '''
    Provides operation to add triples extracted from CSV files using an implementation
of a subset of the
    CSV on the Web standard to a provided Graph.
    Assumptions on the input files:
     * CSVW meta file is provided as JSON-LD serialisation
     * if not specified otherwise, loadCSW() looks for the CSVW meta file for
'/path/to/input.csv' at
        '/path/to/input.csv.csvw'
     * the CSVW meta description RDF graph contains exactly one resource where the a
suffix of the IRI is identical
        to the filename of the provided CSV input and this resource is the 'root' of
the CSVW mapping description
        (otherwise, provide the resource name explicitly using the mappingResouceIRI
parameter of the constructor)
    * only the CSVW datatype definitins 'anyURI', 'string' and 'double' are supported
at the moment
    creators: Brian Walshe (Trinity College Dublin, KDEG)
              Markus Ackermann (University Leipzig, AKSW)
    '''
    def __init__(self, rdfGraph):
        self.graph = rdfGraph


#Provide the parameters in order to construct the function
    def loadCSVW(self, csvFilename, csvwFilename=None, mappingResourceIRI=None):

        if csvwFilename == None:
            csvwFilename = csvFilename + ".csvw"

#2.0.0 Initialize a graph
        #csvwLD = Graph().parse(csvwFilename, format='json-ld')
        self.graph.parse(csvwFilename, format='json-ld')
#2.0.1 Create a meta- graph
        self.graph.serialize(open("metadata.rdf", "w"), "xml")
#2.0.2 Find IRI (=subject) of the JSDON-LD table
        tableNode = self.graph.value(predicate=RDF.type, object=CSVW.Table)
#2.0.3 Find the denoted csv delimiter in the json-ld table
        delim =  self.graph.value(tableNode, CSVW.delimiter)
#2.0.4 Read the csv data
        csvFile = open(csvFilename, "r")
        csvData = csv.reader(csvFile, delimiter=str(delim))
```

```
#2.0.5 Get the object which represent the characteristics of the json-ld model
        schemaRes = self.graph.value(tableNode, CSVW.tableSchema)
#2.0.6 Create triples from the json-ld that describe the csv data in general
        dcatNode = BNode()
        self.graph.add((dcatNode, RDF.type, DCAT.Distribution))
        self.graph.add((dcatNode, DCAT.downloadURL, URIRef(csvFilename)))
        self.graph.add((tableNode, DCAT.distribution, dcatNode))

#2.1.0 List the column names
        mappedColumnsNames = []
#2.1.1 List the columnnames together with their datatypes
        datatypeForColumn = dict()
#2.1.2 List the columnnames together with their URI (propertyUrl)
        propertyForColumn = dict()
        columnList =  self.graph.value(schemaRes, CSVW.column)
#2.1.3 set up for iteration through rdf:rest
        while columnList != None and self.graph.value(columnList, RDF.first)!=None:
#2.1.4 Get the column description (via rdf:first)
            column = self.graph.value(columnList, RDF.first)
#2.1.5 Get the columName from the column description
            columnName = self.graph.value(column, CSVW.name)
#2.1.6 Get the propertyURL (URI) of the column description
            propertyRes = self.graph.value(column, CSVW.propertyUrl)
#2.1.7 State that the propertyURL (URI) is of RDF:property
            # self.graph.add((propertyRes, RDF.type, RDF.Property))
#2.1.8 Create a triple stating a rdfs:label
            if self.graph.value(column, CSVW.title) != None:
              self.graph.add((propertyRes, RDFS.label, self.graph.objects(column,
CSVW.title)))
#2.1.9 Create a triple stating a dc:description
            if self.graph.value(column, DC.description) != None:
               self.graph.add((propertyRes, DC.description, self.graph.value(column,
DC.description)))
#2.1.10 Adding values to the previous defined list/dictionaries as follows:
    #(1)add column name
    #(2)add pairs with columnname(name) and datatype (value)
    #(3)add column names using propertyURL
            mappedColumnsNames += columnName
            datatypeForColumn[str(columnName)] = str(self.graph.value(column,
CSVW.datatype))
            propertyForColumn[str(columnName)] = propertyRes
#2.1.11 Go to the next column description (rdf:rest)
            columnList = self.graph.value(columnList, RDF.rest)

#2.2 Store aboutURL of the json-ld table into a variable
        urlTemplate = self.graph.value(schemaRes, CSVW.aboutUrl)

#2.3 Slice the URI in three parts
        groups = re.match("^(.*?)\{([A-Za-z0-9\-_]+)\}(.*)$", urlTemplate)
        # pre = http://semmtech.nl/ssn/csvw.csv/Requirement
        pre = groups.group(1)
        # post = none
        post = groups.group(3)
        # nameCol = ".{_row}"
        nameCol = groups.group(2)

#2.4 Store all csv column headers in variable
        csvHeader = csvData.next()

#2.5 Map numerical csv column indexes to each csvHeader
        cellname2Index = self._cellToIndexMapping(csvHeader)
        idIndex=-1
```

```
#not relevant in this case
        if nameCol!="_row":
            idIndex = csvHeader.index(nameCol)

#2.6.0 (Start with) iterating over each observation in the csv data
        linesRead = 0
#2.6.1 Create a specific requirement for each room
        rooms = ["Intro-Experience", "Hoofdthema1NederlandEnDeWereld",
"Hoofdthema2DeWereldVanDeKrijgsmacht",
                 "Pronkzaal", "Hoofdthema3MilitairenInDeSchijnwerpers",
"Hoofdthema4DeWereldVanDeTechniek",
                 "Hoofdthema5Operaties", "Hoofdthema6SamenlevingEnKrijgsmacht"]
        for line in csvData:
            Requirement = ["OperatieveTemperatuur_" + rooms[linesRead]]
            # id_fragment=str(linesRead)
            id_fragment = Requirement
            if idIndex!=-1:
                id_fragment = urllib.quote_plus(line[idIndex])
            subject = URIRef("%s%s%s" % (pre, id_fragment, post))
#2.6.2 Create row instance
            self.graph.add((tableNode, CSVW.row, subject))
#2.6.3 Map a row to the Observation concept
            self.graph.add((subject, RDF.type, SE.Requirement))
#2.6.4 Create a dictionary for mapping the data instances to a concept
            mapping = {
                "Condition": "http://semmtech.nl/se/ontology/Condition",
                "Room": "http://semmtech.nl/se/ontology/Room",
                "QuantityValue": "http://semmtech.nl/se/ontology/QuantityValue",
                "Organization": "http://semmtech.nl/se/ontology/Organization",
            }
#2.6.5 For each cell value of a csv row, define a standard subject
            for cellname, i in cellname2Index.items():
                subject = URIRef("%s%s%s" % (pre, id_fragment, post))
#not relevant in this case
                if cellname not in propertyForColumn.keys():
                   #print "no hit for (%s)"%cellname
                   continue
#2.6.6 Redefine standard subject based upon csv column header
                if cellname == "lowerBoundary":
                    subject = URIRef(line[2])

                if cellname == "upperBoundary":
                    subject = URIRef(line[2])

                if cellname == "unit":
                    subject = URIRef(line[2])
#2.6.7 Define a property based upon csv column header
                predicate = propertyForColumn[cellname]
#2.6.8 Get + define a object as a URI or literal
                if "/" in line[i]:
                    obj = URIRef(line[i])
                else:
                    obj = self.makeObject(datatypeForColumn[cellname], line[i])
                #if cellname == "hasValue":
                #    obj =  URIRef(line[4])
                #if cellname == "hasQuantityValue":
                #    obj = URIRef(line[5])
#2.6.9 Add a RDF- data instance to the graph
                self.graph.add((subject, predicate, obj))
#2.6.10 Check + Map + Add a RDF-type to the graph
                concept_URI = ""
                if "/" in str(obj):
                    concept = obj.split('/')[4]
                    if concept in mapping.keys():
                        concept_URI = (mapping.get(concept))
                        self.graph.add((URIRef(obj), RDF.type, URIRef(concept_URI)))
            linesRead += 1
            if(MAX_LINES_TO_PROCESS > 0 and linesRead >= MAX_LINES_TO_PROCESS): break

    def _cellToIndexMapping(self, csvHeader):
        #return dict(zip([x.strip() for x in csvHeader], range(0, len(csvHeader))))
        return dict(zip(csvHeader, range(0, len(csvHeader))))

#    def _sanitizeSID(self, sid):
#        return sid.replace('_', '-')

    def makeObject(self, datatypeStr, val):
        if datatypeStr == "anyURI":
            return URIRef(val)
        else:
            return Literal(val, datatype=datatypeStr)

    def printN3(self):
        print(self.graph.serialize(format='n3'))

    def writeToFile(self, fileName, format="xml"):
        self.graph.serialize(open(fileName, "w"), format)

#2.7 Write to a rdf file
g = Graph()
converter = CSVWtoRDF(g)
converter.loadCSVW("se_requirements_definitief_OUTPUT.csv")
converter.writeToFile("se_requirements_definitief_OUTPUT.rdf")
```

Appendix V A partial sensor RDF- graph

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
   xmlns:csvw="http://www.w3.org/ns/csvw#"
   xmlns:dcat="http://www.w3.org/ns/dcat#"
   xmlns:dcterms="http://purl.org/dc/terms/"
   xmlns:qudt="http://qudt.org/schema/qudt#"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:ssn="http://purl.oclc.org/NET/ssnx/ssn#"
   xmlns:time="http://www.w3.org/2006/time#"
>
  <rdf:Description rdf:about="http://example.com/id/SensorOutput/3398f48f-1d9a-11e6-8982-240a64020db4">
    <rdf:type rdf:resource="http://purl.oclc.org/NET/ssnx/ssn#SensorOutput"/>
    <ssn:hasValue rdf:resource="http://example.com/id/ObservationValue/33a1800f-1d9a-11e6-ad99-
240a64020db4"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://example.com/id/Observation/19c49eae-2e7b-11e6-942b-240a64020db4">
    <ssn:observedProperty rdf:resource="http://dbpedia.org/resource/temperature"/>
    <ssn:observationResult rdf:resource="http://example.com/id/SensorOutput/3398a66e-1d9a-11e6-9e45-
240a64020db4"/>
    <ssn:observationSamplingTime rdf:resource="http://example.com/id/Instant/338f3091-1d9a-11e6-965b-
240a64020db4"/>
    <rdf:type rdf:resource="http://purl.oclc.org/NET/ssnx/ssn#Observation"/>
    <ssn:featureOfInterest rdf:resource="http://dbpedia.org/resource/air"/>
    <ssn:observedBy rdf:resource="http://example.com/id/Sensor/OS12GRFMET115"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://semmtech.nl/ssn/csvw.csv">
    <csvw:row rdf:resource="http://example.com/id/Observation/1b050df0-2e7b-11e6-bf45-240a64020db4"/>
    <csvw:row rdf:resource="http://example.com/id/Observation/1986f970-2e7b-11e6-ae74-240a64020db4"/>
    <csvw:row rdf:resource="http://example.com/id/Observation/1bda5f51-2e7b-11e6-9fc3-240a64020db4"/>
    <csvw:row rdf:resource="http://example.com/id/Observation/1a39808f-2e7b-11e6-8c35-240a64020db4"/>
    <csvw:row rdf:resource="http://example.com/id/Observation/1befe321-2e7b-11e6-8fab-240a64020db4"/>
    <csvw:row rdf:resource="http://example.com/id/Observation/1ca3c9cf-2e7b-11e6-bcb0-240a64020db4"/>
    <csvw:row rdf:resource="http://example.com/id/Observation/1b41c8d1-2e7b-11e6-9242-240a64020db4"/>
    <csvw:row rdf:resource="http://example.com/id/Observation/1c44b9e1-2e7b-11e6-9cef-240a64020db4"/>
    <csvw:row rdf:resource="http://example.com/id/Observation/19e25fe1-2e7b-11e6-80e3-240a64020db4"/>
    <csvw:row rdf:resource="http://example.com/id/Observation/193e309e-2e7b-11e6-9c8b-240a64020db4"/>
    <csvw:row rdf:resource="http://example.com/id/Observation/1bc94851-2e7b-11e6-8de6-240a64020db4"/>
    <csvw:row rdf:resource="http://example.com/id/Observation/1ca96f1e-2e7b-11e6-87c6-240a64020db4"/>
    <csvw:row rdf:resource="http://example.com/

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------- ----------------------------------------

<rdf:Description rdf:about="http://example.com/id/Observation/19ec23de-2e7b-11e6-8679-240a64020db4">
    <ssn:observedBy rdf:resource="http://example.com/id/Sensor/OS12GRFMET115"/>
    <ssn:observationResult rdf:resource="http://example.com/id/SensorOutput/33991b9e-1d9a-11e6-80c1-
240a64020db4"/>
    <rdf:type rdf:resource="http://purl.oclc.org/NET/ssnx/ssn#Observation"/>
    <ssn:observationSamplingTime rdf:resource="http://example.com/id/Instant/338ff3de-1d9a-11e6-a6d4-
240a64020db4"/>
    <ssn:observedProperty rdf:resource="http://dbpedia.org/resource/temperature"/>
    <ssn:featureOfInterest rdf:resource="http://dbpedia.org/resource/air"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://example.com/id/Sensor/OS12GRFMET115">
    <ssn:hasLocation
rdf:resource="http://example.com/id/PhysicalPlace/KrijgsmachtbredeThemaruimteIntroductie"/>
    <rdf:type rdf:resource="http://purl.oclc.org/NET/ssnx/ssn#Sensor"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://example.com/id/Instant/3395c040-1d9a-11e6-9500-240a64020db4">
    <time:inXSDDateTime rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime">2016-02-
16T02:12:00</time:inXSDDateTime>
    <rdf:type rdf:resource="http://www.w3.org/2006/time#Instant"/>
    <time:inXSDDateTime rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime">2016-02-
16T01:16:00</time:inXSDDateTime>
  </rdf:Description>
  <rdf:Description rdf:about="http://example.com/id/Observation/1acc71c0-2e7b-11e6-9439-240a64020db4">
    <ssn:observedBy rdf:resource="http://example.com/id/Sensor/OS12GRFMET115"/>
    <ssn:featureOfInterest rdf:resource="http://dbpedia.org/resource/air"/>
    <rdf:type rdf:resource="http://purl.oclc.org/NET/ssnx/ssn#Observation"/>
    <ssn:observationResult rdf:resource="http://example.com/id/SensorOutput/339b1770-1d9a-11e6-aa36-
240a64020db4"/>
    <ssn:observedProperty rdf:resource="http://dbpedia.org/resource/temperature"/>
    <ssn:observationSamplingTime rdf:resource="http://example.com/id/Instant/3391efb0-1d9a-11e6-8950-
240a64020db4"/>
  </rdf:Description>
```

Appendix W A partial SE RDF- graph

```
?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
   xmlns:csvw="http://www.w3.org/ns/csvw#"
   xmlns:dcat="http://www.w3.org/ns/dcat#"
   xmlns:dcterms="http://purl.org/dc/terms/"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:se="http://semmtech.nl/se/ontology/"
>
  <rdf:Description
rdf:about="http://example.com/id/Requirement/OperatieveTemperatuur_Hoofdthema5Operaties">
    <se:categorizedBy rdf:resource="http://example.com/id/Condition/ThermischComfortVerblijfsruimten"/>
    <se:responsibilityFor rdf:resource="http://example.com/id/Organization/Heijmans"/>
    <se:specifiesBoundaries rdf:resource="http://example.com/id/QuantityValue/1ac88651-1a94-11e6-8f78-
     240a64020db4"/>
    <rdf:type rdf:resource="http://semmtech.nl/se/ontology/Requirement"/>
    <se:specifies rdf:resource="http://example.com/id/Room/Hoofdthema5Operaties"/>
  </rdf:Description>
  <rdf:Description rdf:nodeID="Ne52e96d6b5574bec93b2413b69b9b834">
    <csvw:name>specifies</csvw:name>
    <csvw:required rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean">true</csvw:required>
    <csvw:propertyUrl rdf:resource="http://semmtech.nl/se/ontology/specifies"/>
    <dcterms:description xml:lang="en">Points to a room</dcterms:description>
  </rdf:Description>
  <rdf:Description rdf:about="http://semmtech.nl/se/ontology/upperBoundary">
    <dcterms:description xml:lang="en">Points to the numerical value of a bound</dcterms:description>
  </rdf:Description>

--------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------

<rdf:Description rdf:about="http://example.com/id/QuantityValue/1ac88651-1a94-11e6-b384-240a64020db4">
    <se:unit rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Celcius</se:unit>
    <se:lowerBoundary rdf:datatype="http://www.w3.org/2001/XMLSchema#double">18.0</se:lowerBoundary>
    <rdf:type rdf:resource="http://semmtech.nl/se/ontology/QuantityValue"/>
    <se:upperBoundary rdf:datatype="http://www.w3.org/2001/XMLSchema#double">25.0</se:upperBoundary>
  </rdf:Description>
  <rdf:Description rdf:nodeID="N4de662dd29d64a379d4b9c54a7e1c1c2">
    <csvw:primaryKey>Requirement</csvw:primaryKey>
    <csvw:aboutUrl rdf:resource="http://example.com/id/Requirement/{_row}"/>
    <csvw:column rdf:nodeID="N7dc02fe309834c7c8c124ae909bab4dc"/>
  </rdf:Description>
```

Appendix X A script for linking RDF graphs

```
from rdflib import Graph, Literal, URIRef, Namespace,OWL, XSD, RDF, RDFS, BNode
import csv
import urllib
import re
from operator import itemgetter
import sys
import pprint

#3.0 Initialize a graph
link_graph = Graph()

#3.1 Order the rooms to be mapped together
ifcRooms = ["458", "191", "221", "246", "271", "433", "296", "408"]
ssnRooms = ["KrijgsmachtbredeThemaruimteIntroductie", "KrijgsmachtbredeThemaruimte1",
"KrijgsmachtbredeThemaruimte2", "KrijgsmachtbredeThemaruimtePronkzaal",
"KrijgsmachtbredeThemaruimte3", "KrijgsmachtbredeThemaruimte4",
"KrijgsmachtbredeThemaruimte5", "KrijgsmachtbredeThemaruimte6"]
seRooms = ["Intro-Experience", "Hoofdthema1NederlandEnDeWereld",
"Hoofdthema2DeWereldVanDeKrijgsmacht", "Pronkzaal",
"Hoofdthema3MilitairenInDeSchijnwerpers", "Hoofdthema4DeWereldVanDeTechniek",
"Hoofdthema5Operaties", "Hoofdthema6SamenlevingEnKrijgsmacht"]

#3.2 Map the rooms via a list index
for index in range(len(ifcRooms)):
    ifc =
URIRef("http://linkedbuildingdata.net/ifc/resources20160515_194757/IfcSpace_" +
ifcRooms[index])
    ssn = URIRef("http://example.com/id/PhysicalPlace/" + ssnRooms[index])
    se = URIRef("http://example.com/id/Room/" + seRooms[index])

    link_graph.add((ifc, OWL.sameAs, ssn))
    link_graph.add((ifc, OWL.sameAs, se))

#3.3 write the graph to a rdf- file
link_graph.serialize("link_Graph.rdf")
```

Appendix Y A partial link RDF- graph

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
   xmlns:ns1="http://www.w3.org/2002/07/owl#"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
>
  <rdf:Description rdf:about="http://linkedbuildingdata.net/ifc/resources20160610_001315/IfcSpace_191">
    <ns1:sameAs rdf:resource="http://example.com/id/Room/Hoofdthema1NederlandEnDeWereld"/>
    <ns1:sameAs rdf:resource="http://example.com/id/PhysicalPlace/KrijgsmachtbredeThemaruimte1"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://linkedbuildingdata.net/ifc/resources20160610_001315/IfcSpace_296">
    <ns1:sameAs rdf:resource="http://example.com/id/Room/Hoofdthema5Operaties"/>
    <ns1:sameAs rdf:resource="http://example.com/id/PhysicalPlace/KrijgsmachtbredeThemaruimte5"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://linkedbuildingdata.net/ifc/resources20160610_001315/IfcSpace_271">
    <ns1:sameAs rdf:resource="http://example.com/id/PhysicalPlace/KrijgsmachtbredeThemaruimte3"/>
    <ns1:sameAs rdf:resource="http://example.com/id/Room/Hoofdthema3MilitairenInDeSchijnwerpers"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://linkedbuildingdata.net/ifc/resources20160610_001315/IfcSpace_246">
    <ns1:sameAs rdf:resource="http://example.com/id/PhysicalPlace/KrijgsmachtbredeThemaruimtePronkzaal"/>
    <ns1:sameAs rdf:resource="http://example.com/id/Room/Pronkzaal"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://linkedbuildingdata.net/ifc/resources20160610_001315/IfcSpace_221">
    <ns1:sameAs rdf:resource="http://example.com/id/PhysicalPlace/KrijgsmachtbredeThemaruimte2"/>
    <ns1:sameAs rdf:resource="http://example.com/id/Room/Hoofdthema2DeWereldVanDeKrijgsmacht"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://linkedbuildingdata.net/ifc/resources20160610_001315/IfcSpace_371">
    <ns1:sameAs rdf:resource="http://example.com/id/Room/Hoofdthema6SamenlevingEnKrijgsmacht"/>
    <ns1:sameAs rdf:resource="http://example.com/id/PhysicalPlace/KrijgsmachtbredeThemaruimte6"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://linkedbuildingdata.net/ifc/resources20160610_001315/IfcSpace_346">
    <ns1:sameAs rdf:resource="http://example.com/id/PhysicalPlace/KrijgsmachtbredeThemaruimteIntroductie"/>
    <ns1:sameAs rdf:resource="http://example.com/id/Room/Intro-Experience"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://linkedbuildingdata.net/ifc/resources20160610_001315/IfcSpace_321">
    <ns1:sameAs rdf:resource="http://example.com/id/Room/Hoofdthema4DeWereldVanDeTechniek"/>
    <ns1:sameAs rdf:resource="http://example.com/id/PhysicalPlace/KrijgsmachtbredeThemaruimte4"/>
  </rdf:Description>
</rdf:RDF>
```

Appendix Z A script for a SPARQL rule & IFC visualization

```
import urllib2
from xml.dom.minidom import parse, parseString

import ifcopenshell
import ifcopenshell.geom

#4.0 create a password manager
username = "rakeshkalpoe"
password = "n8Eh3Hb92Cm7"
endpointURL =
"http://triples.test.semmweb.com/catalogs/rakeshkalpoe/repositories/data"

#4.0.1 Create an OpenerDirector object with support for basic HTTP Authentication
password_mgr = urllib2.HTTPPasswordMgrWithDefaultRealm()
password_mgr.add_password(None,  endpointURL, username, password)

#4.0.2 Install the object globally and incorporate it into urlOpen method
auth_handler = urllib2.HTTPBasicAuthHandler(password_mgr)
opener = urllib2.build_opener(auth_handler)
urllib2.install_opener(opener)

#4.1.1 Formulate a SPARQL query to retrieve the malfunctioning rooms
#4.1.2 Based upon the most recent temperature sensor values that
#4.1.3 are not in the domain specified by the lower and upper boundary
query = """
SELECT DISTINCT ?IfcString ?SsnRoom ?Float ?max_DateTime
                           WHERE
                           {
                    {
                      SELECT (MAX(?DateTime) AS ?max_DateTime)
                      WHERE {?Instant time:inXSDDateTime ?DateTime}
                    }

                    {
                      ?IfcRoom a ifcowl:IfcSpace.
                      ?IfcRoom ifcowl:globalId_IfcRoot ?IfcGuid.
                      ?IfcGuid express:hasString ?IfcString.

                      ?IfcRoom owl:sameAs ?SsnRoom.

                      ?SsnRoom a dul:PhysicalPlace.
                      ?Sensor ssn:hasLocation ?SsnRoom.
                      ?Observation ssn:observedBy ?Sensor.

                      ?Observation ssn:observationSamplingTime ?Instant.
                      ?Instant time:inXSDDateTime ?max_DateTime.

                      ?Observation ssn:observationResult ?SensorOutput.
                      ?SensorOutput ssn:hasValue ?ObservationValue.
                      ?ObservationValue qudt:numericValue ?Float.

                      ?IfcRoom owl:sameAs ?SeRoom.
                      ?Requirement se:specifies ?SeRoom.
                      ?Requirement se:specifiesBoundaries ?QuantityValue.
                      ?QuantityValue se:lowerBoundary ?LowerBoundary.
                      ?QuantityValue se:upperBoundary ?UpperBoundary.
                   }

                FILTER ( (?Float < ?LowerBoundary || ?Float > ?UpperBoundary) )
                }
```

```
                              ORDER BY desc(?DateTime)
"""

#4.2.0 Format the SPARQL- query with respect to the REST- rules
escapedQuery = urllib2.quote(query)
requestURL = endpointURL + "?query=" + escapedQuery

#4.2.1 Send and retrieve results via the REST- query using HTTP
request = urllib2.Request(requestURL)
result = urllib2.urlopen(request)
xmlResult = result.read()

#4.3.0 Convert the xml- result in a DOM object
#4.3.0 so it can be accessed by DOM- functionalities
domResult = parseString(xmlResult)

#4.3.1 Loop through the results
#4.3.1 and append the GUIDS through a list
failedRooms = domResult.getElementsByTagName("result")
guidsOfFailedRooms = []

for failedRoom in failedRooms:
    guidOfDefect = failedRoom.getElementsByTagName("literal")[0]
    guidsOfFailedRooms.append(guidOfDefect.firstChild.data)

#4.4.0 Specify to return pythonOCC shapes from ifcopenshell.geom.create_shape()
settings = ifcopenshell.geom.settings()
settings.set(settings.USE_PYTHON_OPENCASCADE, True)

#4.4.1 Initialize a graphical display window
occ_display = ifcopenshell.geom.utils.initialize_display()

#4.4.2 Open the IFC file using IfcOpenShell
ifc_file = ifcopenshell.open("160609_National Military Museum.ifc")

#4.4.3 Display the geometrical contents of the file using Python OpenCascade
products = ifc_file.by_type("IfcProduct")
#4.4.4 Loop through the geometry and
#4.4.4 if IfcSpace and its GUID is in guidsOfFailedRooms, give it a red color
#4.4.4 else, color ifcSpace green
#4.4.5 Set geometry other than ifcSpace to a certain transparancy
for product in products:
    if product.Representation and product.is_a("IfcSpace"):
        shape = ifcopenshell.geom.create_shape(settings, product).geometry
        if product.GlobalId in guidsOfFailedRooms:
            clr = (1,0,0)
        else:
            clr = (0, 1, 0)
        display_shape = ifcopenshell.geom.utils.display_shape(shape, clr)
        ifcopenshell.geom.utils.set_shape_transparency(display_shape, 0.0)
    elif product.Representation:
        shape = ifcopenshell.geom.create_shape(settings, product).geometry
        clr = (1, 1, 1)
        display_shape = ifcopenshell.geom.utils.display_shape(shape, clr)
        ifcopenshell.geom.utils.set_shape_transparency(display_shape, 0.8)

#4.5 Enter the main loop so that the user can navigate
ifcopenshell.geom.utils.main_loop()
```