

Ingenieurfakultät Bau Geo Umwelt
Lehrstuhl für Computergestützte Modellierung und Simulation
Prof. Dr.-Ing. André Borrmann

Development of an integrated data management in civil engineering with the help of methods of the Systems Engineering

Barth, Alexander

Masterthesis
for the Master of Science in Civil Engineering

Author:	Barth, Alexander
Student number:	
Advisor:	Prof. Dr.-Ing. André Borrmann
	Maciej Trzeciak, M.Sc.
	Cornelius Preidel, M.Sc.
Date of issuance:	01. March 2018
Date of submission:	10. July 2018

Zusammenfassung

In den nächsten Jahren werden die Bauvorhaben immer komplexer und größer. Hierdurch wächst auch die Menge der zu verarbeitenden Daten und Informationen. Gleichzeitig erhöht sich dadurch die Komplexität. Eine sehr große Vielfalt an Informationen macht es schwierig, verschiedene Planer zusammenzubringen. Die Qualität und Aktualität von Informationen kann unter einem häufigen Informationsaustausch leiden.

Die Herausforderung bei dem Entwickeln eines ganzheitlichen Datenmanagement im Ingenieurbau ist die sinnvolle Strukturierung und Verknüpfung von Informationen und Daten.
Diese Arbeit schlägt die Kombination des Semantic Webs und der Methoden des System Engineerings vor. Die Umsetzung dieses Ansatzes mit Hilfe eines Prototypen zeigt, dass es Alternativen zu einem Common Data Environment gibt.
Dieser Ansatz wird in dieser Thesis zunächst theoretisch betrachtet: Wie können Projekte mit Hilfe von Breakdown Strukturen strukturiert werden und wie können die einzelne Strukturknoten mit Hilfe der Idee des Semantic Webs miteinander verbunden werden. Anschließend wird das 3D Modell einer Brücke als Fallbeispiel verwendet. Hiermit wird die Umsetzbarkeit des theoretische Ansatz überprüft und evaluiert.

Dieses Fallbeispiel zeigt, dass es möglich ist, ein Projekt mittels Breakdown Strukturen zu analysieren und nach verschiedenen Gesichtpunkten zu gruppieren. Des Weiteren wird an Hand von diesem Beispiel gezeigt, wie verschiedene Strukturknoten miteinander verbunden werden können. Bei kleinen Projekten kann dieser Ansatz die Zusammenarbeit und den Datenaustausch fördern. Bei größeren und komplexen Projekten muss jedoch viel Zeit für die Strukturierung und das Verknüpfen von Informationen und die damit verbundenen Kosten investiert werden.

Abstract

Over the next few years, construction projects will increasingly become more complex and large. This also increases the amount of data and information to be processed. At the same time, this increases complexity. A very wide variety of information makes it difficult to bring together different planners. The quality and timeliness of information can suffer from a frequent exchange of information.

The challenge in developing integrated data management in civil engineering is the sensible structuring and linking of information and data. This thesis proposes the combination of the Semantic Web and the methods of system engineering. The implementation of this approach with the help of a prototype shows that there are alternatives to a common data environment. This approach is initially considered theoretically in this thesis: How can projects be structured with the help of Breakdown Structures and how can the individual structure nodes be linked together with the help of the idea of the Semantic Web. The 3D model of a bridge is then used as a case study. This is used to check and evaluate the realizability of the theoretical approach.

This case study shows that it is possible to analyze a project using Breakdown Structures and group them according to different aspects. This example also shows how different structure nodes can be connected to each other. For small projects, this approach can promote cooperation and data exchange. For larger and complex projects, however, a lot of time must be invested in structuring and linking information and the associated costs.

Contents

List of Figures

Listings

List of abbreviations

KIF Knowledge Interchange Format
RDF Resource Description Framework
OWL Web Ontology Language
URI Universal Resource Identifier
W3C World Wide Web Consortium
IFC Industry Foundation Classes
CAD Computer Aided Design
BIM Building Information Modeling
BMVI *Federal Ministry of Transport and Digital Infrastructure*
OKSTRA *Object catalogue for road and traffic engineering*
SBS System Breakdown Structure
PBS Product Breakdown Structure
WBS Work Breakdown Structure
OBS Organization Breakdown Structure

Chapter 1

Introduction and Motivation

1.1 Motivation

According to various studies, digital change is one of the major challenges facing the construction industry at international and national level [16]. An essential background for this change is that the complexity of construction tasks is also growing continuously. In particular the production and construction sectors, which are important for the industry, have an increased potential to benefit from such digitization.

With the increasing complexity of a construction project the amount of data and information that needs to be processed is also growing. At the same time this increases the complexity. A large variety of information makes it difficult to bring together different planners. The quality and timeliness of information can suffer from a frequent exchange of information. This can be for example because information is no longer up-to-date or is lost.

For this reason, a way must be found to store the amount of data and information that has to be processed in a clear and structured manner. Not only the storage is of great interest, but also the exchange of information is important. In order not to lose the quality and timeliness of data in a frequent exchange of information, a way must be found to solve this problem.

1.2 Goal of the thesis

In this work a basic structure for cross-domain cooperation is being developed using the example of an infrastructure structure. The exchange and linking of information from individual points of view will be examined in more detail using the example of a bridge construction.

A prototype based on the RESTful API from Allplan Bimplus, which is a digital platform for collaboration in construction projects, is to be developed in order to demonstrate to demonstrate the realizability of this thesis.

1.3 Structure of the thesis

This Masters thesis is structured as follows: First, chapter 2 deals with the digital methods of the construction sector. This includes the definition of BIM and then how BIM is implemented in infrastructure projects. The terms IFC and Model View Definition are also briefly described. The standard OKSTRA, which is related to linked data and infrastructure projects, is also presented. The next chapter (chapter 3) describes the technical basis for the implementation of this Masters thesis. The corresponding topics are Semantic Web and System Engineering. Since this thesis deals with the structuring of data, this chapter also describes how classification is defined in the German standard and which other standards are available for this purpose. The actual state of the art concerning the used methods will be described in the chapter 4. The chapter 5 contains a solution for the optimal structuring of data in infrastructure projects. In chapter 6 a case study on the solution approach is carried out and evaluated. The 7th chapter (chapter 7) finally explains the implementation of a prototype. The individual components that are necessary for the conversion are described here. The Masters thesis is completed with chapter 8 (chapter 8). This should once again reflect the complete thesis. In addition, the Outlook section will describe how the prototype can be expanded.

Chapter 2

Digital Methods in the building environment

2.1 BIM

The basic idea concerning Building Information Modeling (BIM) exists since 1970. It is a method of mapping buildings with all their relevant information using a consistent, digital building model over their entire life cycle. These include not only physical but also functional properties. If all these properties are combined in a database, each component can be precisely identified and described with the stored information during the entire life cycle of a building. Furthermore, this information database helps to make decisions regarding the building [17].

2.2 BIM for infrastructure

On behalf of the *Federal Ministry of Transport and Digital Infrastructure* (BMVI) the *planen-bauen 4.0* creates a step-by-step plan. This should define a common understanding of the BIM method. Cooperation with BIM is a prerequisite, that the data exchanged between the different parties is compatible. A corresponding standard already exists for building construction. It is called Industry Foundation Classes (IFC) (see also section 2.3).

In infrastructure projects the national *Object catalogue for road and traffic engineering* (OKSTRA) is currently used for a uniform data exchange. In the following section the concept of OKSTRA will be descriped more precisely. To ensure the comprehensive applicability of the international standard IFC also in the infrastructure sector, the object catalogue is currently being expanded with support from the BMVI. Since mid-2017 more and more transport infrastructure projects have to fulfill the BIM requirements of performance level 1.

Performance level 1 includes the following minimum requirements:

- The principal must specify in the *Auftraggeber-Informations-Anforderungen* (AIA) exactly which data he needs and when
- All services to be provided are to be delivered in digital form on the basis of 3D model-based work
- The data supplied by the contractors must be checked for compliance with the AIA
- Manufacturer-neutral data formats must be required in the tender to ensure data exchange
- The required hardware and software must be generally available
- BIM must be included in the contract as an applicable planning instrument

The step-by-step plan, which was mentioned before, provides for various phases. The individual phases of the step-by-step plan can be seen in Figure 2.1.

The first phase is described as a *preparatory phase* and will last from 2015 to 2017. The second one is descriped as *extended pilot phase* and will last from 2017 to 2020. In this phase, the number of pilot projects is to be significantly increased in order to collect experience across all planning and construction phases. Numerous guidelines, checklists and samples will also be developed, which can be used by all projects in the future.

The last phase is described as *BIM Level I for new projects to be planned* and is to apply as of 2020. As soon as the basic conditions are met, BIM with a performance level is to be regularly applied in all transport infrastructure construction projects to be planned from the end of 2020 according to the step-by-step plan [1].

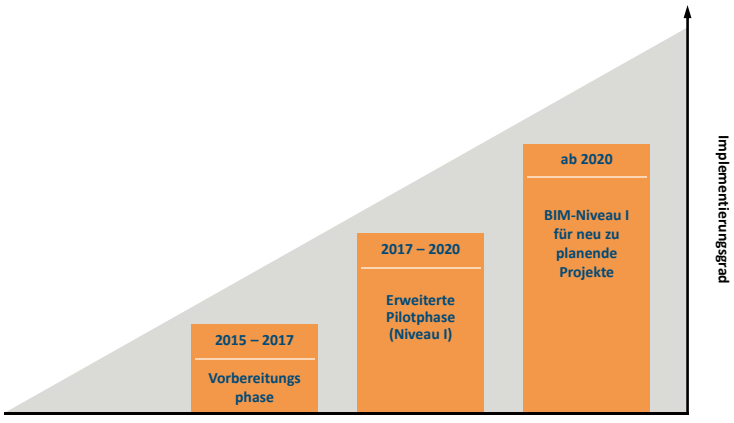


Figure 2.1: Schematic representation of the step-by-step plan [1]

In October 2016 the BMVI commissioned the BIM4INFRA 2020 working group to create the conditions for the implementation of the step-by-step plan. The most important prerequisites for this are [18]:

- The development of an achievable performance level for the introduction of BIM
- Monitoring of the pilot projects and expansion of the pilot phase
- Investigation of legal issues and preparation of recommendations for future contracts,
- Provision of relevant guidelines and samples for the award and processing of BIM services, in particular BIM applications
- Identification of requirements for uniform data structures for the infrastructure sector, development of a uniform database concept and BIM library
- Information and Public Relations

In January 2017 the BMVI published a first progress report on the gradual plan. It describes that four pilot projects on BIM have been funded since 2015. Two of these projects are railway projects: a tunnel in Rastatt and a bridge in Filstal. The other two projects are road construction projects: a bridge on Lake Petersdorf and one in the Auenbach valley.

Since 2016, one year after its introduction, two further road construction projects have been supported. These pilot projects help to describe, analyse and evaluate structures, processes and interactions of project participants in the application of BIM.

The conclusion of the progress report is that important areas of BIM in infrastructure construction can already be implemented very well. To further develop standardization in the area of infrastructure construction, the BMVI supports a German participation in the development of the IFC for road, rail and bridge. The next version of IFC (IFC 5) should contain the Infrastructure area [19].

2.3 IFC

IFC is a acronym for Industry Foundation Classes. It represents an open standard that makes it possible to digitally describe building models. This standard is published by buildingSMART International. It became necessary because more and more data had to be exchanged between different systems. In the beginning any changes were entered into the system manually. As the buildings and the time pressure increased over time, the organization designed this standard.

Today, numerous systems support this standard, which has greatly facilitated the exchange of building data. The standard is used in various areas, such as CAD systems, static and energy calculations or quantity and cost calculations.

In contrast to classic CAD programs, the IFC file contains information about the building in terms of walls, windows or rooms with their respective properties. Classic CAD programs only export lines, points or blocks without their explicit properties. IFC has been an international standard under ISO 16739 since Release IFC4.

With IFC, a distinction must be made between the IFC file and the IFC data schema. The IFC file is used as a container for the transfer in step format. The IFC data schema defines the specifications in the EXPRESS format [20].

The IFC data schema is modelled in the EXPRESS language. EXPRESS is defined in ISO 10303. It displays the information using file types, entities, rules and relationships, and other objects. Thus it is possible to describe an IFC data schema object-oriented. In the further development of the IFC standard, it is particularly important to add and update further objects. Since 2014 the IFC standard version 4 is available.

2.4 IFC Model View Definition

A IFC Model View Definition, also called Model View Definition, is a subset of an IFC schema. It is required to meet one or more exchange requirements in the construction industry. BuildingSMART uses the Information Delivery Manual (ISO 29481) to define such exchange requirements [21]. The Information Delivery Manual contains the information, which each project participant in each part of the development process should provide [22].

An official Model View Definition was published by buildingSMART in the mvdXML format. With a new version of the mvdXML format it is also possible to validate the data content of an IFC file using an mvdXML file. In such an mvdXML file different exchange requirements are written down [23].

The manual creation of such exchange requirements is complex and complicated. For this reason there are a number of MVDs specified as legal subsets of the IFC scheme according to general use cases or purposes. Furthermore there are only a few software solutions for writing such a MVD File. Figure 2.2 shows an example of different Model View Definitions.

With the first Model View Definition shown in the figure, the user only gets a coordinate view of the structure. The second Model View Definition is particularly interesting for structural engineers, since a IFC Structual Analysis View is generated here. The last Model View Definition is for the energy analysis of a building. Here the The IFC Thermal Analysis View is being displayed.

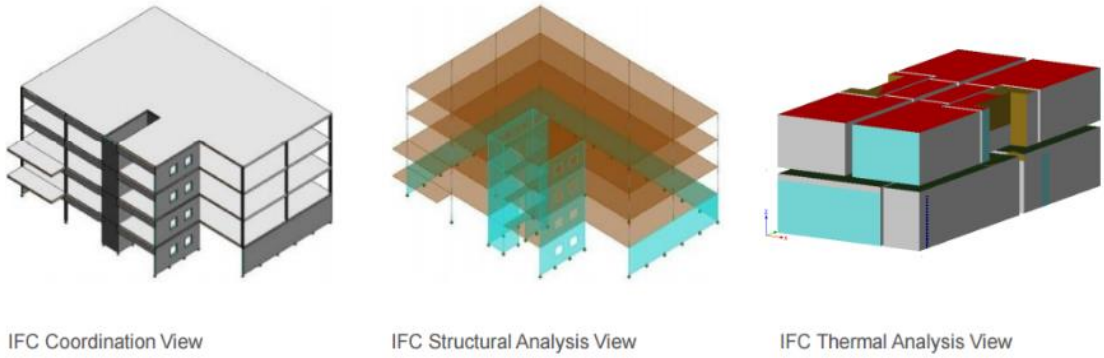


Figure 2.2: Example for different Model View Definitions [2]

2.5 OKSTRA

OKSTRA stands for *Objektkatalog für das Straßen- und Verkehrswesen* and is a collection of objects from the field of road and transport. This object catalog was created in order to have a uniform understanding of the objects contained in the corresponding departments. The Federal Highway Research Institute is in charge of this project. A component in the object catalog is formally described with its attributes and relations in a data schema. This object catalogue is an open standard and is continuously developed further [24].

Chapter 3

Technical Background

3.1 Semantic Web

Besides the classic Web of Documents, the World Wide Web Consortium (W3C) helps to set up a new standard to support a Web of Data, as you find it in databases, for example. The goal of Web of Data is to exchange data across the local network. The W3C is an international community that develops open standards to ensure a continuous development of the web. It was founded in 1994 by Tim Berners-Lee at the MIT Laboratory for Computer Science in Cambridge.

The term Semantic Web refers to the vision of the W3C from the web of linked data. This technology makes it possible for users to create data storage on the web and to write rules for the handling of your data. Linked data is supported by technologies such as RDF, SPARQL, OWL and SKOS [25]. By linking data it is possible to prove their origin clearly and to check the credibility of data.

3.1.1 Ontologies

The term ontology is often used nowadays; especially in connection with Semantic Web. Originally it comes from philosophy and means *teaching of being*. A clear definition does not exist in philosophy since many different philosophers use this term. They define the ontology in a different way, depending on the point of view. Some philosophical problems can be considered problems of ontology.

The purpose of ontology in philosophy is to represent the world with its objects and connections between them [3].

In the Stanford Encyclopedia of Philosophy four parts of the great discipline of ontology were finished [26]:

- The study of ontological commitment, i.e. what we or others strive for
- The study of what there is
- The study of the most general characteristics of what exists and how the things that exist are metaphysically connected in the most general way
- The study of meta ontology, that is, to say what task it is that the discipline of ontology should aim, if anything, to understand how the questions it wants to answer should be understood and with what methodology they can be answered

In computer science, ontology is understood as an analogy or metaphor. An ontology is often used to formalize and utilize a meaning of information [27]. Tim Berners-Lee has summarized the following requirements that can apply to the description languages of ontology [28]. These languages must:

- have a relatively compact syntax
- have a well-defined syntax so that you can say exactly what is displayed
- have enough expressiveness to represent human knowledge
- have an efficient, powerful and understandable argumentation mechanism
- can be used to build up large knowledge bases

In connection with Semantic Web, ontologies should enable and improve communication between different applications, but also between people. Ontologies are encoded using different languages. The most common languages are: Knowledge Interchange Format (KIF), Resource Description Framework (RDF) and Web Ontology Language (OWL). In the following both chapters only the current languages are being defined that are used in the Semantic Web area and have become a standard.

RDF

RDF stands for Resource Description Framework and is a standard model for data exchange on the Web. It was introduced by the W3C and contains functions that, among other things, facilitate the merging of data. This is possible even if the existing schemes are different. It also supports the possibility of updating schemas without having to change all data consumers.

It extends the connection structure of the web by an Universal Resource Identifier (URI) to name the relationship between two elements and additionally their ends. The combination of two elements and their combination is called RDF triple. Each *triple* has a subject, predicate and object. An example represents Figure 3.1. Both structured and unstructured data can be linked via these connections.

All connections combined result in a directed, labeled graph. This graphical representation is the simplest mental model for RDF and is often used in easy-to-understand visual explanations [29].

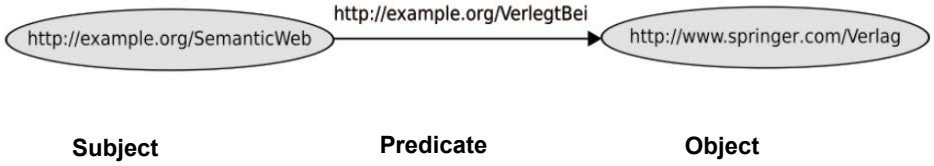


Figure 3.1: RDF triple [3]

Web Ontology Language

The field of application of RDF is quite limited, since it only allows the representation of ontological knowledge. With this *language* it is possible to express subclasses and property hierarchies with their domain and area definitions. However, there is no way to express specific properties of classes such as symmetry or Boolean combinations.

The Web Ontology Language OWL is an extension of RDF. It is a standard of the W3C and is currently (as of 2017) the most widely used ontology language. It was published in 2004 by the W3C. OWL extends the RDF by defining additional vocabulary to describe properties and classes. It contains some logical primitives such as universal or existential quantifiers and ways to limit properties [3].

OWL has three *sub-languages* with different expressiveness. Each individual language fulfils different requirements. Figure 3.2 shows the different levels.

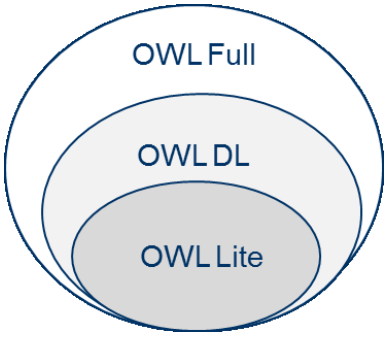


Figure 3.2: OWL sub languages [3]

OWL Full is intended for the user who wants to use the whole variety of OWL. It has been developed to be compatible with RDF documents.

OWL Description Logic (OWL DL) is a subset of OWL Full. It contains the maximum expressiveness, but is not fully compatible with RDF documents.

OWL Lite represents an even smaller subset of OWL Full. It should support those users who require a classification hierarchy and simple restrictions. For example, it allows cardinality restrictions, but only cardinality values 0 and 1; it should be easier to develop tools for OWL Lite than for its more expressive relatives; and OWL Lite provides a quick way to migrate thesauri and other taxonomies. OWL Lite also has a lower formal complexity than OWL DL [3].

A second version was released in 2012. OWL 2 is an extension and revision of the old version [30].

3.1.2 Linked Data

The internet is a network of websites. Linked Data, on the other hand, describes a complex network of data. The methods of such a network has already been described in subsection 3.1.1. The concept behind Linked Data was developed by Tim Berners Lee. He laid down four rules [31]:

- URI to describe objects,
- HTTP use URIs to look up names,
- When someone looks up a URI, useful information should be presented using the standards (RDF *, SPARQL),
- Add links to other URIs so that more objects can be detected.

Each component includes URIs, HTTP, RDF, and XML to retrieve information. In connection with linked data one can often read the term linked open data. Linked Open Data is a powerful combination of Linked Data and Open Data. Open Data is data that can be freely used and distributed by anyone. The requirement for this data is usually only to provide it with attributes and to share it with others. This open data can be made available to anyone without being linked to other data. In addition, they can also be linked internally so that they are not freely available to the public.

To counter internal linking, the W3C community and all data openness advocates have sought to provide a Linked Open Data Cloud. The Linked Open Data Cloud uses both linked data and open data sources. A graph database[1] as an example is able to process large amounts of raw data from different sources and link them to open data. This enables more extensive queries and results in data management and analysis [32].

An example for Linked Open Data is DBpedia[2]. It is a crowdsourcing community that tries to extract structured content from Wikimedia projects. The data is provided as Linked Data and can be viewed via the web page and via SQL similar queries, such as Sparql (subsection 3.1.3) [33].

As the topics of Semantic Web Technologies and Linked Data in Architecture, Engineering and Construction (AEC) become more and more interesting, a number of research institutions and initiatives have been established. One of these initiatives is the workshop series *Linked Data in Architecture and Construction* (LDAC). They have created two different community groups, the *W3C Community Group on Linked Building Data* and the *BuildingSMART Linked Data Working Group*.

The *W3C Community Group on Linked Building Data* is managed by the W3C and aims to define existing and future use cases and requirements for Linked Data based applications over the life cycle of buildings. Participants in this group are experts from the fields of BIM and *Web of Data Technologies*.

The *buildingSMART Linked Data Working Group* is responsible for creating and maintaining an ifcOWL ontology as a derivation of the IFC STEP EXPRESS schema. This group is part of the *Technical Room* of buildingSMART and works closely with the other working groups within this organization [34].

ifcOWL

In 2013 a new version of the IFC schema was published (IFC 4) and became also as ISO standard. Not only the EXPRESS scheme was developed, but also a XSD scheme. One goal of the XSD schema should be that IFC4 can be used more flexibly in XML-based environments.

Through the development of the Semantic Webs, various authors proposed an ontology for the IFC schema. It is known as ifcOWL. The essential point in the development of this ontology was to develop it as close as possible to the EXPRESS scheme of IFC [35].

[1] https://ontotext.com/products/graphdb/
[2] http://wiki.dbpedia.org

In the meantime, the proposed ifcOWL ontology of *buildingSMART International* has been taken up and further developed. With the help of this ontology building data can be displayed using Semantic Web. The resulting IFC data is thus available in linked and labeled graphics. With such a graph and the underlying data, building data can be easily linked with other graphs, for example material data, manufacturer data of products.

The result is a very large network of linked data, which offers great possibilities for data management and exchange in the building industry and beyond [36]. In Figure 3.3 you can see an example for ifcOWL. This figure demonstrates how complex a linked data network can become.

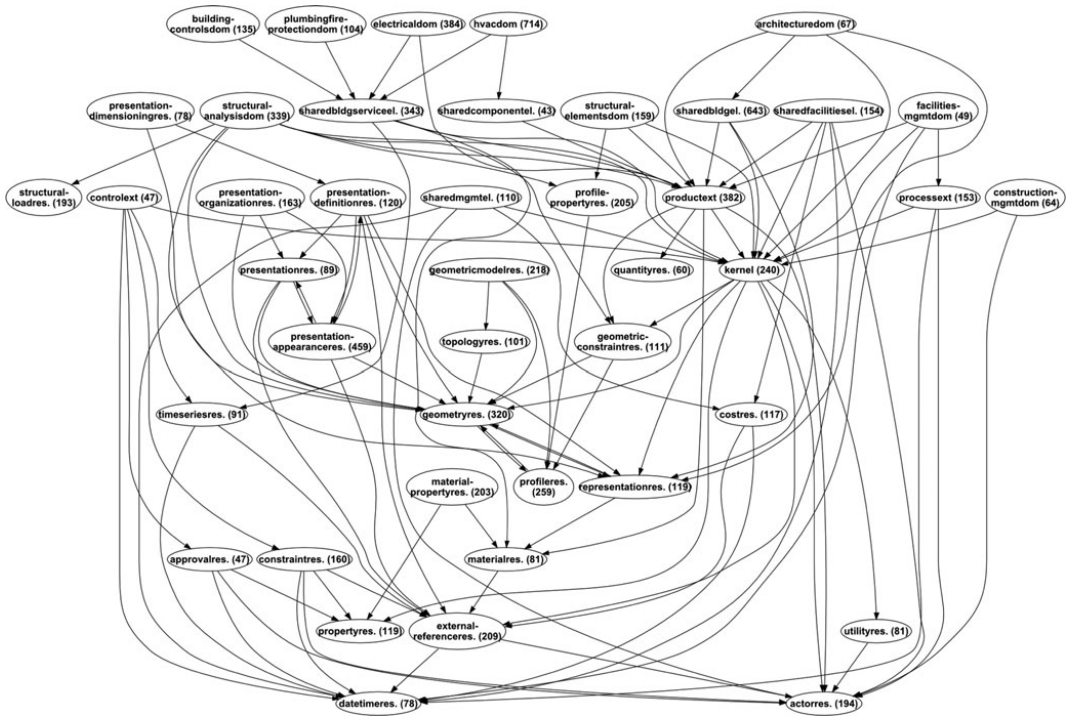


Figure 3.3: ifcOWL Ontologie [4]

3.1.3 Sparql

SPARQL is a W3C standard and a generic query language. It is used to express queries across different data sources. It is not important whether the data is saved natively as RDF or viewed as an RDF graph via links. The SPARQL query language also contains functions for querying required and optional diagram patterns as well as their conjunctions and disjunctions. It also supports aggregation, subqueries, negation and the generation of values using expressions [13].

Compared to domain-specific languages, such as MySQL, SPARQL is particularly suitable for scenarios in which data from multiple sources is required. Examples of this are:

- All building objects should be classified according to a certain type
- Where are the locations of the companies that produce the materials for the walls in the hall?

These two examples not only require building data as the source, but also data from other fields or documents. They are easier to implement with RDF and SPARQL technologies without the need for proprietary systems [14]. The example (Listing 3.1) shows a SPARQL query to find the title of a book from the given data graph. The query consists of two parts:

SELECT identifies the variables to be displayed in the result,

WHERE provides the basic diagram pattern corresponding to the data graph.

The basic diagram pattern in this example consists of a single triple pattern with a single variable (? title) in the object position.

```
1 SELECT ?title
2 WHERE
3 {
4   <http://example.org/book/book1> <http://purl.org/dc/elements/1.1/
        title> ?title .
5 }
```

Listing 3.1: Example Query [13]

If SPARQL is applied to the construction industry, it reaches its limits. Typing, properties, spatial relationships and many other things that are defined or described in a building model are difficult or impossible to find using this query language.

3.2 Classification systems

3.2.1 DIN 32705

The DIN 32705 is a DIN standard that descripes rules for the creation and further development of classification systems. It defines the classification system as follows [37]:

classification system is the structured representation of classes and the conceptual relationships between them

class is the summary of those terms that have at least one identical characteristic

characteristic is a term element that is defined by a statement about the property of an object

notation in the classification system is a character string formed according to certain rules, which represents a class, a term or a combination of terms and maps their position in a systematic context.

As already described above, a classification system is a tool to sort objects or even information about them. The DIN 32705 distinguishes three types of sorting:

1. technical-practical: Arrange, Classify, Group
2. scientific: Use logical tools to adequately represent statements about objects
3. cognitive: pointing out connections

A classification system is structured by the relationship between classes and terms. There are also two different types of relationships: The formal relationship that connects things based on external characteristics and the material relationship that connects things based on content characteristics. A structure is developed from top to bottom. At the top is the main class, which groups the whole system in a certain way.

The system is specified further and further down. When specifying, it is important that the individual classes are mutually exclusive so that a unique structure is created. Such a structure could look like this [37]:

- office machines
    - typewriter
    - calculator
    - copying machine

This structure describes the office machines. The first level groups the next level. In the second level, the machines are titled more precisely. The third level could be a further refinement of the individual machines.

3.2.2 ISO 12006-2:2015

The ISO 12006 is an international standard dealing with the structuring of information for the construction industry. It consists of three parts, of which part one has never been published.

Part two of this standard was published in 2015. It defines a framework for the development of classification systems for structures by defining a framework and a set of recommended table titles supported by definitions. It identifies classes for the organization of information and specifies how these classes are related. Part three of the ISO 12006 contains a framework for object-oriented information.

The second part of the ISO 12006 lists the tables recommended for the development of built environment classification systems. There are tables for the different object classes according to particular views. For example, there are some by form or function. In addition this standard describes how the individual object classes are related [38].

This Standard does not constitute a complete operational classification system, nor does it provide the contents of the tables. Nevertheless it includes some examples. It is rather intended for organizations that develop and publish such classification systems and tables at national or regional level. Classification tables may vary in detail to meet local needs [38].

The definitions included apply to the entire life cycle of structures, including design, production, maintenance and demolition, as well as to building construction and civil engineering.

3.2.3 OmniClass

The *OmniClass Construction Classification System*, also called OmniClass, is a new classification system for the construction industry. It is helpful for various applications, such as: Organizing book or project information or developing a classification structure for electronic databases.

This system was developed to provide a standardized basis for the classification of information created and used by the North American architecture, engineering and construction industries over the entire life cycle of the structure. The concept for OmniClass is based on the international framework ISO 12006-2 (see subsection 3.2.2). This international framework has a direct impact on OmniClass, as the *OCCS Development Committee* has closely followed this standard.

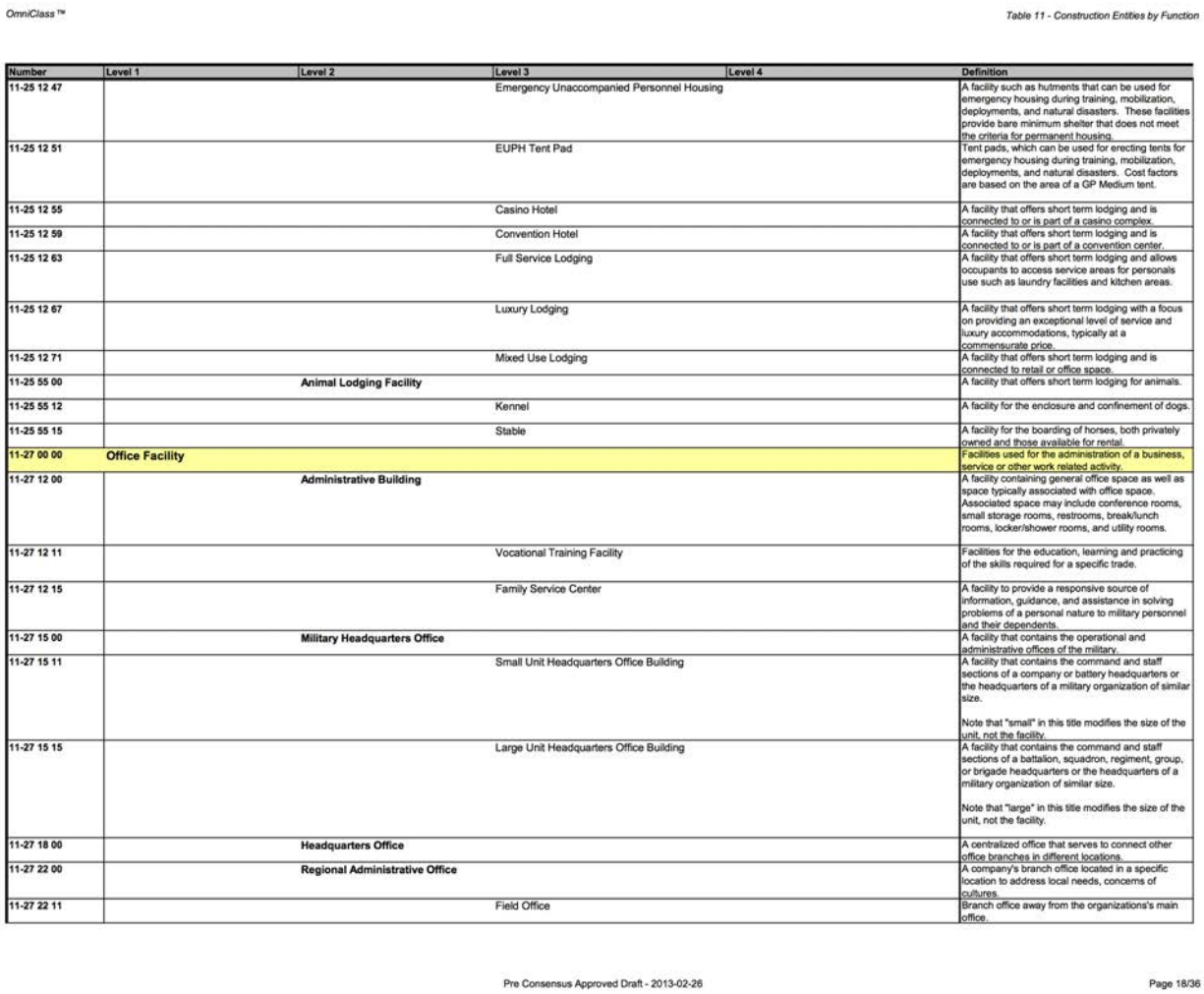
The following guidelines were drawn up by the *OCCS Development Committee*:

- It is an open and extensible standard available to the AEC industry in general
- There is a comprehensive and open exchange of information between participants of development of *OmniClass*.
- It is being developed and updated with the broad participation of industry
- Development is open to any person or organisation willing to actively participate in development
- The industry as a whole and not just one organization will control the development and deployment of OmniClass
- It focuses on North American terminology and practice
- It is compatible with the relevant international standards for classification systems
- Efforts being made in other parts of the world will be reviewed and, if necessary, adapted
- Existing classification systems, references and research materials relevant to the development of OmniClass are taken into account in the formulation of OmniClass

The development of OmniClass is an ongoing process that is accessible to all interested parties and whose content can be expanded over time to meet unmet needs as they arise [39].

The idea behind the OmniClass was to combine the various existing classification systems such as MasterFormat, Uniformat, Uniclass and other classifications. Today it contains 15 tables covering the complete building life cycle, from building design to building dismantling. The high degree of detail in the individual tables of the classification systems makes it difficult to combine the various classification systems [40].

The following figure shows one page of the OmniClass Tables which can be found in the *Table 11 - Construction Entities by Function*.



OmniClass™

Table 11 - Construction Entities by Function

Number	Level 1	Level 2	Level 3	Level 4	Definition
11-25 12 47			Emergency Unaccompanied Personnel Housing		A facility such as hutments that can be used for emergency housing during training, mobilization, deployments, and natural disasters. These facilities provide bare minimum shelter that does not meet the criteria for permanent housing.
11-25 12 51			EUPH Tent Pad		Tent pads, which can be used for erecting tents for emergency housing during training, mobilization, deployments, and natural disasters. Cost factors are based on the area of a GP Medium tent.
11-25 12 55			Casino Hotel		A facility that offers short term lodging and is connected to or is part of a casino complex.
11-25 12 59			Convention Hotel		A facility that offers short term lodging and is connected to or is part of a convention center.
11-25 12 63			Full Service Lodging		A facility that offers short term lodging and allows occupants to access service areas for personals use such as laundry facilities and kitchen areas.
11-25 12 67			Luxury Lodging		A facility that offers short term lodging with a focus on providing an exceptional level of service and luxury accommodations, typically at a commensurate price.
11-25 12 71			Mixed Use Lodging		A facility that offers short term lodging and is connected to retail or office space.
11-25 55 00		**Animal Lodging Facility**			A facility that offers short term lodging for animals.
11-25 55 12			Kennel		A facility for the enclosure and confinement of dogs.
11-25 55 15			Stable		A facility for the boarding of horses, both privately owned and those available for rental.
11-27 00 00	**Office Facility**				Facilities used for the administration of a business, service or other work related activity.
11-27 12 00		**Administrative Building**			A facility containing general office space as well as space typically associated with office space. Associated space may include conference rooms, small storage rooms, restrooms, break/lunch rooms, locker/shower rooms, and utility rooms.
11-27 12 11			Vocational Training Facility		Facilities for the education, learning and practicing of the skills required for a specific trade.
11-27 12 15			Family Service Center		A facility to provide a responsive source of information, guidance, and assistance in solving problems of a personal nature to military personnel and their dependents.
11-27 15 00		**Military Headquarters Office**			A facility that contains the operational and administrative offices of the military.
11-27 15 11			Small Unit Headquarters Office Building		A facility that contains the command and staff sections of a company or battery headquarters or the headquarters of a military organization of similar size. Note that "small" in this title modifies the size of the unit, not the facility.
11-27 15 15			Large Unit Headquarters Office Building		A facility that contains the command and staff sections of a battalion, squadron, regiment, group, or brigade headquarters or the headquarters of a military organization of similar size. Note that "large" in this title modifies the size of the unit, not the facility.
11-27 18 00		**Headquarters Office**			A centralized office that serves to connect other office branches in different locations.
11-27 22 00		**Regional Administrative Office**			A company's branch office located in a specific location to address local needs, concerns of cultures.
11-27 22 11			Field Office		Branch office away from the organizations's main office.

Pre Consensus Approved Draft - 2013-02-26

Page 18/36

Figure 3.4: Table 11 – Construction Entities by Function - OmniClass [5]

3.2.4 Uniclass

UniClass was originally released in 1997. It can be used to structure project information according to a recognized standard. In 2015, a new version was released by NBS as part of the BIM Toolkit project.

NBS stands for National Building Specification and is a UK-based system of building specifications used by architects and other construction professionals to describe the materials, standards and execution of a building project [41].

This original version of the UniClass has been extensively revised to make it more suitable for use in modern construction practice. It should also be compatible with BIM. The original version was only available in book form and had to be bought. The new version, on the other hand, has now been made available digitally [42].

Uniclass (Version 2015) is a uniform classification system for British industry covering the entire construction sector. Similar to OmniClass (see subsection 3.2.3), it contains different tables, each dealing with a different object class. According to the NBS, the current version of UniClass offers the following advantages [43]:

- A uniform classification system for the construction industry. For the first time, buildings, landscape and infrastructure can be classified in a uniform scheme
- A hierarchical set of tables that support classification from a university campus or street network to a floor tile or curb unit.
- A numbering system that is flexible enough to meet future classification requirements
- A system according to ISO 12006-2 that is mapped to NRM1 and will support mapping to other classification systems in the future.
- A classification system maintained and updated by NBS
- The BIM toolkit contains a database of synonyms to make it as easy as possible to find the required classification using industry-standard terminology.

3.3 Systems Engineering

3.3.1 Definition

Systems Engineering is an interdisciplinary approach to develop and realize complex systems in large projects. Large projects can no longer be managed by individuals. For this reason a new methodology was sought.

The *Methodology for Systems Engineering*, which was published by Arthur Hall in 1962, received much attention. A few years later, Professor A. Büchel of the ETH Zurich adopted and reinterpreted this approach [44]. In the early 1970s this reinterpretation was further developed into an independent System Engineering methodology.

It is a structured methodology for controlling problem-solving processes in the context of complex issues. All in all, it makes sense to use this system engineering methodology for projects with large project complexity and scope in which many participants are involved.

The aim of this methodology is to gain a holistic view of the problem situation from different perspectives in order to subsequently transfer the initial situation into an optimal target state[6].

The System Engineering methodology is based on two basic methodological concepts: the life phase model and the problem-solving cycle. Its main goal is the optimal interaction of both basic concepts.

The life phase model is a rough grid that describes the purpose for the individual phases. In each of these phases the results to be expected are specified. In technical systems the individual phases are divided as follows:

- Development
- Realization
- Use
- Disposal

The problem-solving cycle describes several steps that are important to get from the task to the solution in a questionnaire. The three main steps are:

- Target search (target formulation)
- Solution search (concept analysis)
- Selection (decision)

If the life-cycle model is combined with the problem-solving cycle, it can be seen that some tasks need to be solved in each individual phase. An example for coordinated processing of tasks is shown in Figure 3.5. The hatched arrows represent the coordination.

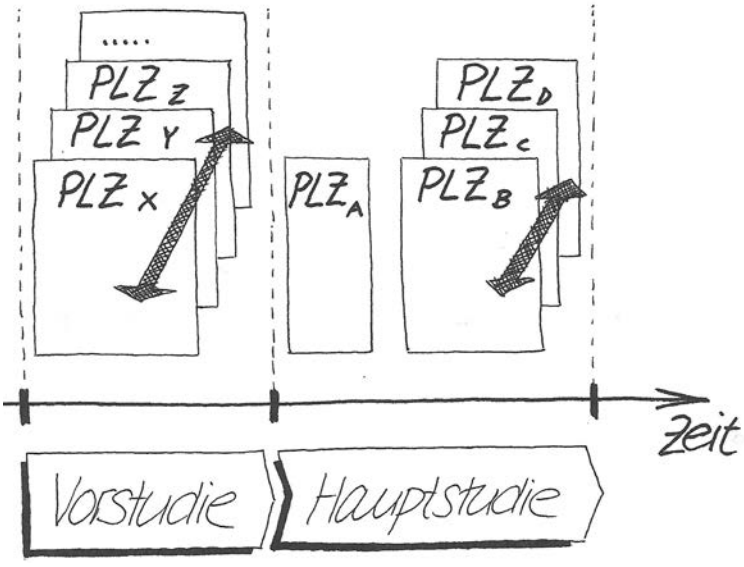


Figure 3.5: Coordinated processing of subtasks in every phase of life [6]

3.3.2 Breakdown Structure

When working with complex projects, it is crucial to understand the relationship between the product and the division of labor. This makes it possible to overcome the uncertainties regarding the effects on the project. They also help to balance the resources needed to develop and implement the system [45].

These relationships are often defined in the project as System Breakdown Structure (SBS), Product Breakdown Structure (PBS), Work Breakdown Structure (WBS) and Organization Breakdown Structure (OBS) [46].

System Breakdown Structure

The SBS is a logical decomposition of the system. At definition of the SBS it is necessary to consider:

- How the system is procured
- How the system is designed
- What the system has to do
- How they manage the critical interfaces of the system

The SBS helps to document and communicate the functionality of the system. As soon as the project has developed towards the implementation, the individual points of SBS are traced back to the corresponding points of the PBS.

Product Breakdown Structure

The PBS is created to ensure that all required products are considered in the planning. It starts with the end product at the top of the hierarchy and describes the products that are to be manufactured and delivered. Furthermore, it serves to communicate the results of a project and to show exactly what the project will achieve.

Work Breakdown Structure

The WBS is a hierarchical decomposition of the work necessary for the completion of a project. Together with PBS the WBS addresses how each part of system or product defined in PBS should be finished. The WBS serves the overall planning of the project and supports the project management, the cost estimation, the project status reporting as well as the

deadline and resource management. As a rule the WBS is made in early phases of a project life cycle.

In Figure 3.6 the first two level of a WBS for an automobile project is represented as an example. The first level describes the overall task: Complete a automobile. The following level specifies the individual tasks to complete this automobile: Body and styling, Central electrics, Chassis and suspensions, Engine and fuel system, Gears and transmission.

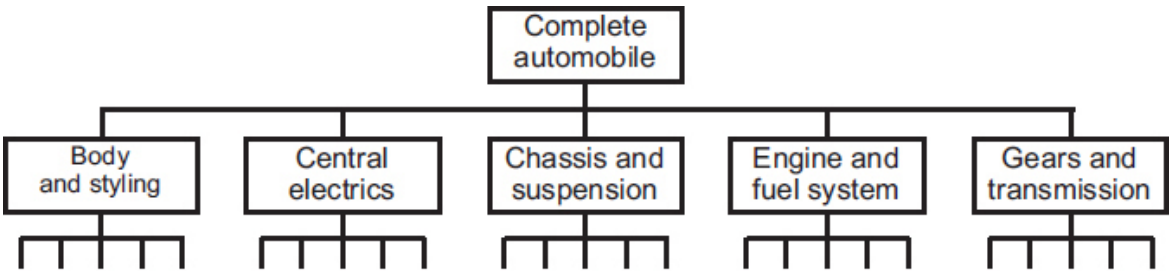


Figure 3.6: WBS for an automobile project [7]

Organization Breakdown Structure

The OBS used in connection with the WBS represents the organizational structure for the project on the way to completion. OBS covers the organization or persons carrying out the works defined in WBS. When creating a OBS it is important that the organizational structure takes into account the required resources and connected team members. Under the definition of an OBS the WBS should be connected with the lowest level of project responsibility within the OBS hierarchy.

Each of the structures described above can, if used separately, be used to analyze the complexity of the project and help with system and project analysis. Once the SBS, PBS, WBS and OBS are applied together, the weak points in the project become visible that need to be adjusted to achieve a better balance between cost, resources and planning [47].

Chapter 4

State of the art

4.1 BIM

Open interfaces are particularly important for the topicality and success of BIM. Up to version 4.0, the IFC exchange format only supports building models.

Only with the next version of IFC it will be possible to descripe buildings of the infrastructure such as roads, bridges and tunnels [48]. To be able to work in the infrastructure area before the release of IFC 5, the IFC alignment project was carried out. The goal of this project was to develop a data model for the description of a route.

Researchers at the TU Munich and the French CSTB, as well as companies such as AEC3 Germany and Bentley Systems, have been involved in the implementation of this project.

Over time, three new data models were created from this data model:

- IFC-Road
- IFC-Bridge
- IFC-Tunnel

The following image (Figure 4.1) shows how the standards IFC-Road, IFC-Bridge and IFC-Tunnel will form the basis for data model IFC-Alignment.

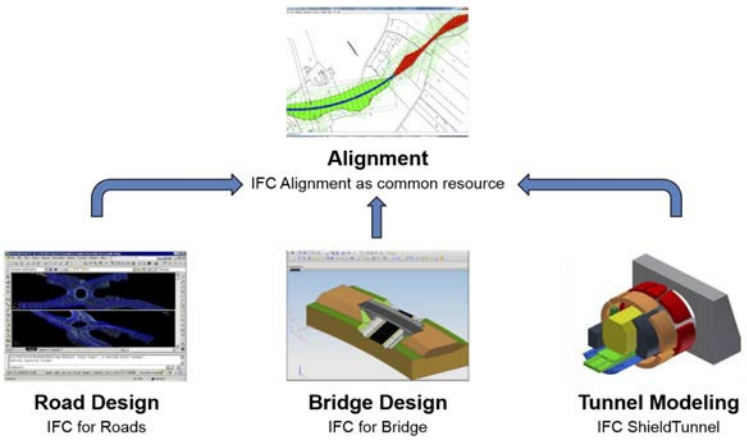


Figure 4.1: Data model IFC-Alignment [8]

An adaptation of the data format standard OKSTRA (see section 2.5) is also relevant for the further development in the area of infrastructure. This is especially necessary to establish compatibility between this data format standard and the IFC alignment project.

In order to create this compatibility the Federal Highway Research Institute has commissioned the *Lehrstuhl für Computergestützte Modellierung und Simulation* at the Technical University of Munich. The idea is to develop a way to convert between these two formats.

The TU Munich has developed the Open Infra Platform as a solution approach. With this platform the routing data can be viewed in different file formats. Furthermore, it is possible to convert the data between different file formats [49]. Figure 4.2 shows the currently supported file formats.

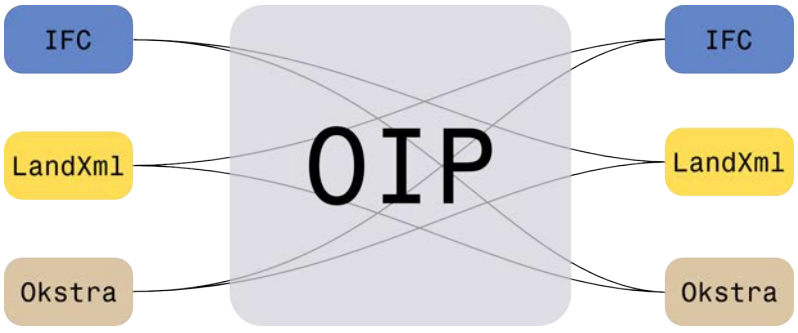


Figure 4.2: Supported file formats [9]

According to a study [50], the measures to create databases with different types of information generated in construction projects and their linking are still inadequate. The aim of this study is to establish a construction information database system based on BIM technology. This database system is intended to enable comprehensive management of information generated

during the construction phase. The result of this study is that such a database system can be used for the collection and use of construction information.

The study also presents the problems of information diversity. Documents contain numerous and diverse information that often overlap. Examples of this are work reports and safety instructions. Work reports contain a lot of general information about the project that is recurring. However, they also contain information during construction, such as information on construction progress. The information is categorized using the Space Breakdown Structure and the Information Breakdown Structure. They are also connected to the BIM model for easy management.

A similar approach is described in the paper by Jaehyun Park and Hubo Caib [51]. The integration of construction documents into BIM is a challenge due to their heterogeneous and unstructured data formats. Jaehyun Park and Hubo Caib present a database design based on a project structure plan in their paper. It should enable the creation of a dynamic and multi-dimensional BIM database for the timely integration of construction records. The design records are defined as an additional dimension. In addition, an automatic linking mechanism between construction projects and BIM objects is created to generate the multi-dimensional BIM database. The aim is to develop a WBS-based BIM database structure to ensure that the entire as-built documentation contains all construction documents during the construction phases in a timely manner. The implemented WBS coding structure is based on the MasterFormat and UniFormat. This makes it easy to generate WBS codes using the reference tables. The result of this paper is that this method makes it possible to integrate the construction documents collected during the construction phase into a dynamic BIM database to create a complete inventory documentation.

Building Information Modeling, in both four- and five-dimensional forms, has recently attracted much attention in the architecture, engineering and construction industries. Although current 5D models link cost items and scheduling activities to BIM elements, it is still necessary to link and integrate cost and scheduling data manually. The paper by Su-Ling Fan, Chen-Hua Wu and Chien-Chun Hun [52] proposes a model for the automatic linking of cost and appointment data after their linking with BIM elements. The dependency between schedule and costs is obvious, as costs and schedules are closely linked in terms of their management process, as they share common data such as costs, resources and quantities. In practice, however, they often remain two independent functions and use two different control structures: the Work Breakdown Structure and the cost break structure. The different level of detail of the individual structures leads to a fundamental difference in the maintenance of cost and schedule data. The data must be connected manually by the project manager. The proposed solution in this paper is to link cost items and schedule activities with BIM elements. This approach provides a simplified cost and schedule integration process. The system can thus automatically define the relationship between schedule and costs.

4.2 Semantic Web

4.2.1 Progression of the web

In 2007 Nova Spivack published a timeline on his web page (see Figure 4.3) in which he presents the development of the Internet.

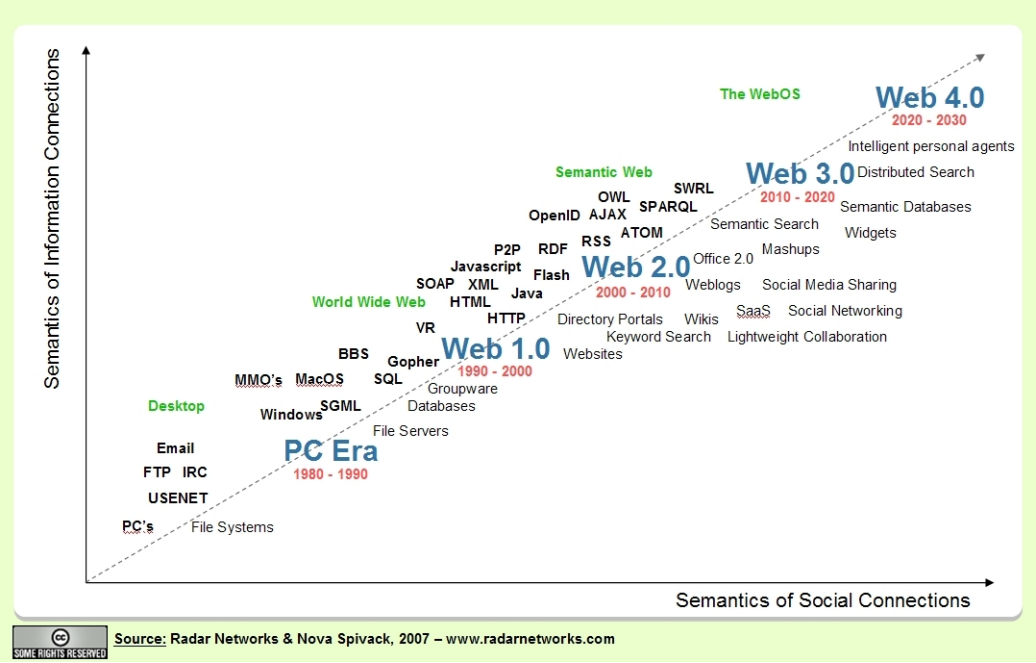


Figure 4.3: Timeline of the Internet [10]

He describes Web 3.0 as the third decade of the Internet in which several key technologies will be widely used. According to Nova Spivack the focus is on RDF and the technologies of the emerging Semantic Web [10]. According to the timeline Nova Spivack sees Web 4.0 as the WebOS. Key technologies are in *Distributed Search* and *Intelligent personal agents*. He predicts Web 4.0 in 2020 - 2030.

4.2.2 Database services

Since the introduction of the RDF in 2004, several service providers have provided graphic database services. One of the newest is Amazon Neptune[1]. Released in November 2017, it is a fast, reliable, fully managed graphic database service.

[1] https://aws.amazon.com/de/neptune

According to the product page, this service makes it very easy to create and run applications that work with highly related records. Amazon Neptune supports the most common diagram models, such as RDF from W3C and the associated query languages TinkerPop Gremlin and SPARQL. An example of an application is a knowledge diagram application. These can be used to store information in a diagram model. This allows the user to navigate through data records that are linked together in a complex manner.

Figure 4.4 shows how a knowledge diagram could look schematically. As you can see on following figure, the individual nodes are linked with a predicate. This corresponds exactly to the RDF schema.

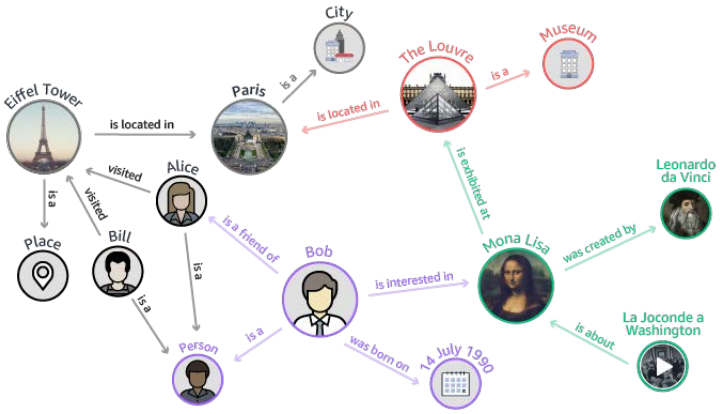


Figure 4.4: Amazon Neptun knowledge diagram [11]

Another platform is Mobi[2]. It is a free and open platform. With this platform it is possible to link native data sources to a knowledge graph. Mobi was built with Apache Karaf and uses OWL 2 to create ontologies. Like Amazon Neptun, Mobi uses SPARQL query language for data search [53].

Other useful applications for creating RDF graphics are:

- D2RQ[3]
- ontop[4]
- Karma[5]

[2] https://mobi.inovexcorp.com
[3] http://d2rq.org
[4] http://ontop.inf.unibz.it
[5] http://usc-isi-i2.github.io/karma/

4.2.3 Using SPARQL for BIM

In the chapter 3 it was described that SPARQL in connection with the construction industry reaches its limits. To counteract this C. Zhang and J. Beetz describe in their research report, *Querying Linked Building Data Using SPARQL with Functional Extensions* [14] a way to extend the query language SPARQL.

They extend SPARQL with new *words* from the construction industry, so that the query (see Listing 4.1) becomes possible. The query in Listing 4.1 calculates the ratio between the area of protected openings and the area of all external walls of a floor.

```
SELECT (SUM(?windowArea)/(SUM(?wallArea)))
WHERE
{
  ?wall qrw:isContainedIn ?storey .
  ?wall a ifc:IfcWall .
  ?wall pset:isExternal true .
  ?wall pdt:hasGrossWallArea
  ?wallArea .
  OPTIONAL {
    ?window qrw:isPlacedIn
    ?wall .
    ?window pset:protected true .
    ?window pdt:hasWindowArea ?windowArea
  }
}
```

Listing 4.1: Example Query [14]

SPARQL has also been expanded in other areas of the construction industry. The Open Geospatial Consortium (OGC), for example, has published Geo-SPARQL as a series of extended functions for geodata [54]. It is a representation and query of spatial data. This makes it possible to query which sights are within a defined range.

```
SELECT ?a
WHERE {
  ?a a ex:Attraction;
      geo:hasGeometry ?ageo .
      FILTER(geof:within(?ageo,
        "POLYGON((
          -77.089005 38.913574,
          -77.029953 38.913574,
          -77.029953 38.886321,
          -77.089005 38.886321,
          -77.089005 38.913574
          ))"^^sf:wktLiteral))
```

```
13 }
```

Listing 4.2: Example Query [15]

4.2.4 PANDORA

In connection with the long-term project *Johann Friedrich Blumenbach - online*, which is a project within the framework of the Academy Program of the Union of German Academies of Sciences in Götting, the software architecture PANDORA LOD Framework is developed.

PANDORA stands for *Presentation (of) Annotations (and) Notations (in a) Digital Object Repository Architecture*. The basis for this framework are digital images of texts and objects which are stored in a Fedora Commons repository. The Fedora Commons Repository is an open source repository that allows you to manage and access digital objects in electronic archives. This is primarily used in libraries and universities [55].

The PANDORA project itself is a collection of open source applications that use a common document, called a manifesto, to organize the presentation of data to the user. This manifest consists of a JSON-LD document and is generated from a digital object repository through the dynamic use of SPARQL queries [56].

Chapter 5

Approach & Methodology

5.1 Introduction

The amount of data and information that needs to be processed is increasing when construction projects become more complex. It is difficult to bring different planers together because of the very large variety of information. The quality and timeliness of information can suffer from a frequent exchange of information, for example because information is no longer up-to-date or is lost.

A promising approach is the creation and use of domain models as a primary means of information exchange between engineers without the need for a purely document-based exchange of information. The information of the individual models must be linked together in order to create a holistic digital image of the project or product to be created. For this linkage of the information the Linked Data approach is suitable.

This represents an approach for the construction industry, with the help of which data can be structured and linked. In this way data is connected holistically and allows information to be queried comprehensively when required. This methodology is originally derived from the Linked Open Data. The difference between these two is the access rights. Linked Open Data are informations or data that are freely accessible [32]. The data in the Linked Data however represent a kind of closed project space, so that only authorized persons can access them.

Compared to the common data environment, this approach is intended to be another way of making information available to the entire team at a common point. In the following the chosen solution approach and how it differs from a common data environment is described in more detail.

5.2 Approach

The previous chapters explained a lot about the basics of the Semantic Web and System Engineering. The question now arises as to how and why these two systems can now be linked together.

In addition it must be explained why a new approach is being developed, although there is already a comparable approach with the Common Data Environment.

A common data environment is a central storage location for all project information. Because all project information is stored centrally, previous projects can always be accessed. An administrator assigns different access rights to the storage folders. In addition a workflow for distributing and publishing data must be defined beforehand in a common data environment. The responsibility and management of such a system are defined in so-called *Employer's Information Requirements*. These requirements define the information that an employer needs for a successful project.

On the other hand there is this approach. Complex and large projects are first broken down into their individual components and then classified using structures. The individual components are sorted, for example, by function for a System Breakdown Structure or by work packages for a Work Breakdown Structure. The division or structuring according to functions is carried out by the project manager. The Work Breakdown Structures, on the other hand, are created by the individual disciplines that work on the construction site. Each discipline creates its own structure. They classify and structure their tasks on the construction site according to work packages. This classification originally comes from project management and describes a method of System Engineering. Besides the Breakdown Structures mentioned above, there are other types, as already described in subsection 3.3.2.

By structuring and classifying the project, individual areas become visible and can be viewed and edited in detail. As soon as a complex and large construction project is classified into the individual Breakdown Structures according to this method, these structures are linked together. The procedure of Semantic Web is particularly well suited for this. The individual nodes of the different structures are connected with a kind of RDF triple. This means that the node of the first structure represents the subject and the node of the other associated structure represents the object. In order to be able to describe such a connection clearly, it must be provided with a predicate.

For example, a node of a Work Breakdown Structure that represents a particular work package is linked to a node of a System Breakdown Structure that describes a particular functionality. This link is established because a certain work package fulfils a certain function of the project. Additional work packages may also be required for this particular function. This link shows which tasks are necessary to perform this function. This is of great interest for cost estimation

or time planning, for example. Another link could be between two nodes of a Work Breakdown Structure and a Product Breakdown Structure. The connection visualizes which products are necessary to fulfill the corresponding work package.

The connection is only described Requirements in a one-directional way, since the connection in the opposite direction could have a different meaning. This means that if a bidirectional connection is to be established, an additional connection must be established. This precise description of the connection makes it possible for the creator or third parties to trace it again and again. With Semantic Web it is possible to connect data over a large network.

Even here there is a difference to the Common Data Environment. The CDE creates *only* a document structure with different access rights. With the solution approach described here the project is broken down into its individual components and then classified according to requirements using structures.

However, it is not sufficient to structure a project in such a way that it makes sense to work with it, since only a breakdown by functions or work packages is available up to now. The actual data exchange between one or more disciplines has not yet taken place. To do this, the required information or data must also be assigned to the individual nodes. This can be done by adding attributes or tasks, but also by assigning different project components to the node.

Since the nodes of two or more Breakdown Structures have been interconnected, the properties or information need not be exchanged directly. This prevents the loss or falsification of information. As already described above, this could be information on costs in the form of a service specifications or also on the time sequence in the form of a Gantt chart. This exchange is also one of the advantages of the Common Data Environment.

As soon as the user views a structure, the system displays the other structure nodes to which the selected node is connected. In addition to the linked nodes, he also sees the properties and information that are bound to the other nodes. The user himself can only edit or delete his own data. He does not have the necessary rights to delete external properties or information.

Likewise the user himself can only remove his own links. In order to be able to delete information from others, he can make a kind of request. This indicates to the other side that the attached information should be removed if necessary. This request may be accepted by the other party and the information may be deleted or rejected.

By linking the Breakdown Structures of a large and complex building, a large information network is created. This leads to a smooth and fast exchange of properties and information, since files can also be attached to a structure node. These can be contracts or expert opinions, for example.

In order not to be overwhelmed by the amount of information, a kind of filter must also be included. It determines which information is visible to whom. For example, a contract is only important for one particular counterparty. However, a third person might see this or not. This type of model view definition is also comparable to a type of rights management. The user who appends information to a node determines who is allowed to see this information.

The approach described above describes file exchange from a different perspective than a common data environment. In the Common Data Environment, files are stored in a large folder structure and protected by access rights. With this proposed solution the data and information is sorted much more precisely because it is attached to individual structure nodes. These structure nodes can represent a function or a work package, for example. The actual exchange of information only takes place when these two nodes are connected.

The solution described above is to be implemented using a RESTful API and a corresponding front-end. When implementing the RESTful API, the existing functions of the RESTful API of BimPlus are to be integrated. REST stands for *REpresentational State Transfer* and API for Application Programming Interface. This is a programming interface based on the paradigms and behavior of the World Wide Web and describes an approach for communication between client and server in networks.

Allplan Bimplus is an open and cloud-based platform for BIM and can be used in all disciplines to work together efficiently and quickly in construction projects. 3D models, information, documents and tasks can be managed centrally over the entire life cycle of a building.

Since the BimPlus frontend only meets a few requirements for the implementation of the prototype, a new frontend is implemented. Front-end is the website that is made available to the user.

BimPlus offers the possibility to work in a team with 3D models as described above. In addition, structures can be created and attachments or components from the loaded 3D model can be attached to them. It is not possible to link structures and append tasks or free attributes to a structure node to describe the node with additional information.

For these reasons and also for legal reasons, a new frontend is programmed and a RESTful API is also created. All functions that are called via the frontend should also be able to be executed independently with the RESTful API. This has the advantage that the RESTful API can also be used with another frontend. The RESTful API of the BimPlus is implemented despite the lack of possibilities, as it already contains functions such as team collaboration and the ability to upload buildings or similar model data. This means that the newly programmed RESTful API should embed the functions of the RESTful API of BimPlus and add new or existing functions. In order to gain access to the functions of BimPlus, the user only needs a BimPlus user account and an application ID. This is also necessary in order to work in a team and with 3D models.

In the following chapter the implementation of the solution described above is checked using a 3D model. This is followed by a description of the implementation and the exact functioning of the prototype. The programming languages and libraries used in the front-end and the RESTful API are also explained. The chapter also describes how the two systems are made available on the Internet.

Chapter 6

Case Study & Evaluation

6.1 Case Study

The solution approach from the chapter 5 is to be tested with the help of a bridge model. The model used was created by Keshan P. and is freely available via the 3D Warehouse of Google Sketchup[1]. It represents the Kalutara bridge over Kalu Ganga river.

The 3D model used is a massive bridge with two lanes. For this approach it is assumed that this bridge will have to be build new. The method described in the previous chapter should therefore be applied to this 3D model. In Figure 6.1 the used model is shown:

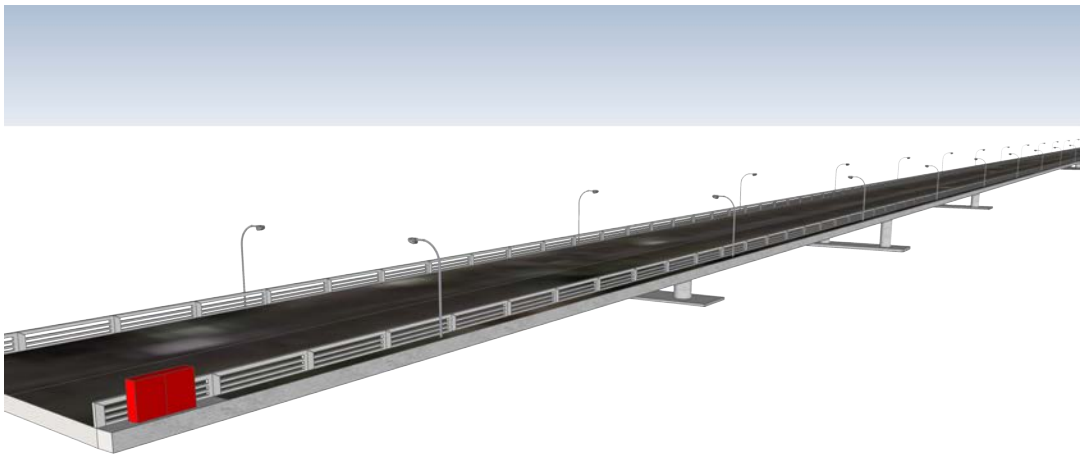


Figure 6.1: Bridge Model

[1] https://3dwarehouse.sketchup.com/model/e231c69bf1a66c16cabf5a7f1ed5cec3/Kalutara-Bridge-Sri-Lanka

First, this construction project is broken down into its individual functions or properties using a System Breakdown Structure, which has been explained in subsection 3.3.2. The project manager takes over this structuring, since he knows the exact function of the construction project and can thus structure the project very well. This is structured as follows. At the top of the structure, the system itself is the construction of a new bridge. The next level is the Sub Systems. The project is classified into an electrical, mechanical and constructional system. The individual components come after the sub-system level. These are in the electrical system for this project: the power supply for the construction site, any lighting systems, communication lines and sensors for the road surface.

The next sub-items for the mechanical system are: Bridge supports, expansion joints and water drains. Most components are available for the constructional system, since they represent the essential part of the construction. These include: All foundations, piers, bridge superstructure, road surface and road safety measures. Figure 6.2 shows the Breakdown Structure described above.

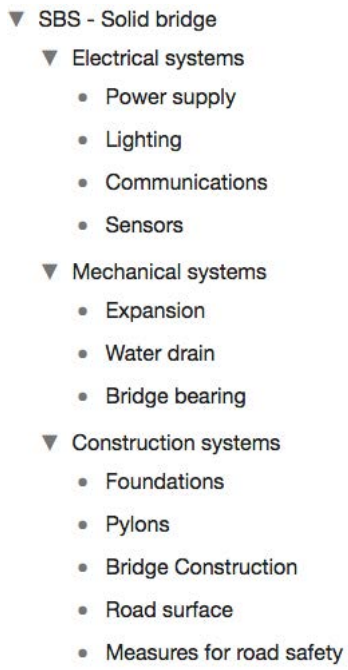


Figure 6.2: SBS - Bridge

After the construction project has been broken down into its individual functions, it is now broken down into the individual work steps required to build the bridge. These individual steps are classified using one or more work breakdown structures. In contrast to the system Breakdown Structure, several Work Breakdown Structures are generated. Each individual discipline that is active on the construction site creates its own Breakdown Structure. They classify and structure their tasks, which must be carried out on the construction site, ac-

cording to work packages. For this solution approach, the structures of two disciplines are considered as examples: The electrician and the shell builder.

The shell builder in this project will take over any construction work that is necessary for the construction of the bridge. The electrician takes care of any services that have to do with electricity or communication according to his description.

The second level of these the two Work Breakdown Structures represents the different construction phases. The individual phases are: Planning phase, preparation phase, construction phase and final phase.

The Work Breakdown Structure for the electrician is as follows: During the planning phase, the architect draws up any plans for laying and installing the pipes based on the architect's plans. In the preparation phase the power supply on the construction site and a basic supply must be provided by means of light and distribution boxes. During the actual construction phase, the various cables and sensors must be installed. As soon as the construction site is completed, the electrician must disconnect the power supply on the construction site in the final phase. In the following picture (Figure 6.3) the structure described above can be seen:

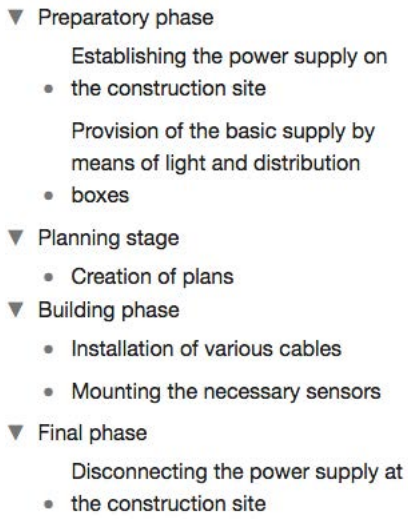


Figure 6.3: WBS - Electricians

In the planning phase the shell builder also draws up his plans based on the architect's plans. In the following preparatory phase he prepares the site for his work. This includes clearing trees or levelling the required areas. During the actual construction phase the shell builder builds the foundations and piers required for the subsequent bridge construction. The expansion joints must also be installed in this phase before the bridge structure can be placed and the road surface can be paved. Once the previous work has been carried out the water drains and any road safety work must be carried out. In the final phase all areas necessary

for the construction of the bridge will be cleared. In the following picture (Figure 6.4) the structure described above is shown.

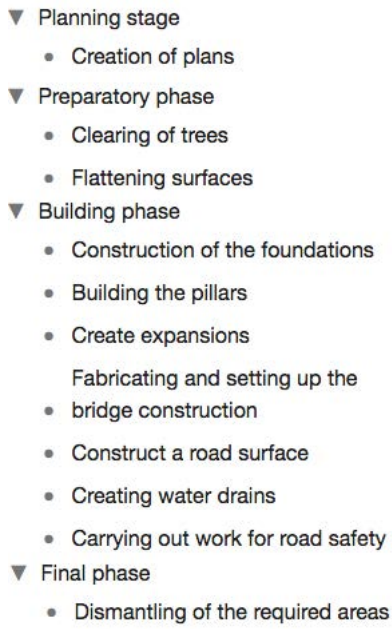


Figure 6.4: WBS - Shell builder

In order to be able to continue working with the structures without problems and effort, they are generated using the JSON format. This format is easy for the creator to read and understand. Such a structure can be written in this format with little effort. In addition structures in JSON format can be interpreted easily and very quickly by computer systems. The following figure (Listing 6.1) shows an extract from the JSON file for the System Breakdown Structure described above. The complete JSON files for the generated structures can be found in the appendix.

```
{
    "name": "SBS – Bridge",
    "type": "Structure",
    "children": [
        {
            "name": "Construction System",
            "type": "Structure",
            "children": [
                {
                    "name": "Foundations",
                    "type": "Structure",
                    "number": 1
                },
                {
                    "name": "Pylons",
```

```
16                                  "type": "Structure",
17                                  "number": 2
18                              },
19                              {
20                                  "name": "Bridge Construction",
21                                  "type": "Structure",
22                                  "number": 3
23                              },
24                              {
25                                  "name": "Road surface",
26                                  "type": "Structure",
27                                  "number": 4
28                              },
29                              {
30                                  "name": "Measures for road safety",
31                                  "type": "Structure",
32                                  "number": 5
33                              }
34                          ],
35                  "number": 1
36              }, {
37                  "name": "Electronical System",
38                  "type": "Structure",
39                  "children": [...],
40                  "number": 2
41              }, {
42                  "name": "Mechanical System",
43                  "type": "Structure",
44                  "children": [...],
45                  "number": 3
46              },
47      ]
48  }
```

Listing 6.1: JSON: SBS - Solid Bridge

Each node of the structure consists of four properties: Name, type, children and number. The number is important for the correct sequence of the nodes. The type indicates that it is a structure. The name contains the name of the node. In the code example can be seen, the nodes of the next level are attached to the children property.

After the structures have been successfully created, the individual nodes must be provided with information and properties. As described above, these can be components, contracts, site plans or attributes and tasks.

In the Figure 6.5 can be seen the System Breakdown Structure with assigned objects from the 3D model. The corresponding components from the 3D model were assigned to each structure node. In the above mentioned picture only exemplary connections between the model and the structure are shown. Many components, such as the electrician's, are not visible in this model. These can still be selected in the programmed tool, since the individual components can be added via a list.

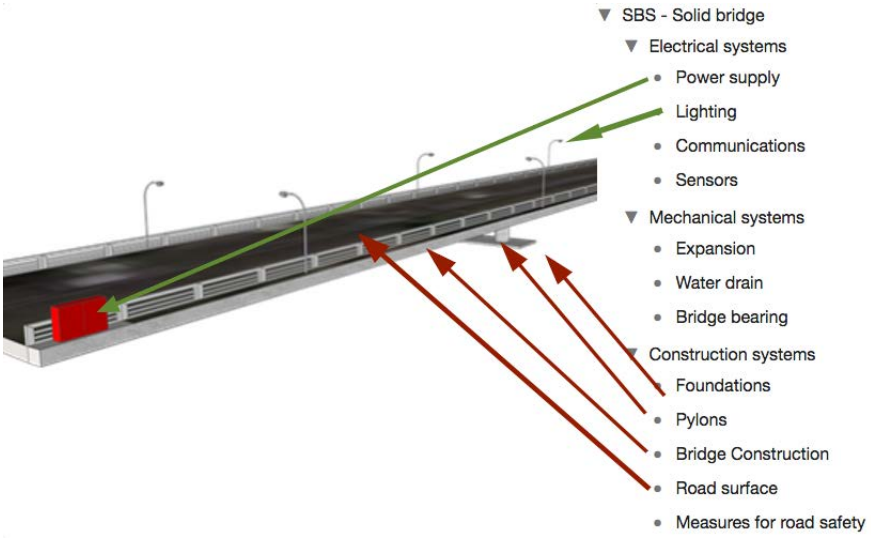


Figure 6.5: SBS - Assigned Objects

Similar to the System Breakdown structure, the corresponding components are also attached to the Work Breakdown structure by the shell builder. In addition, information about the schedule and costs is appended to the nodes. The following figure (Figure 6.6) shows the structure for the shell builder. Here you can see the assigned parts from the 3D model.

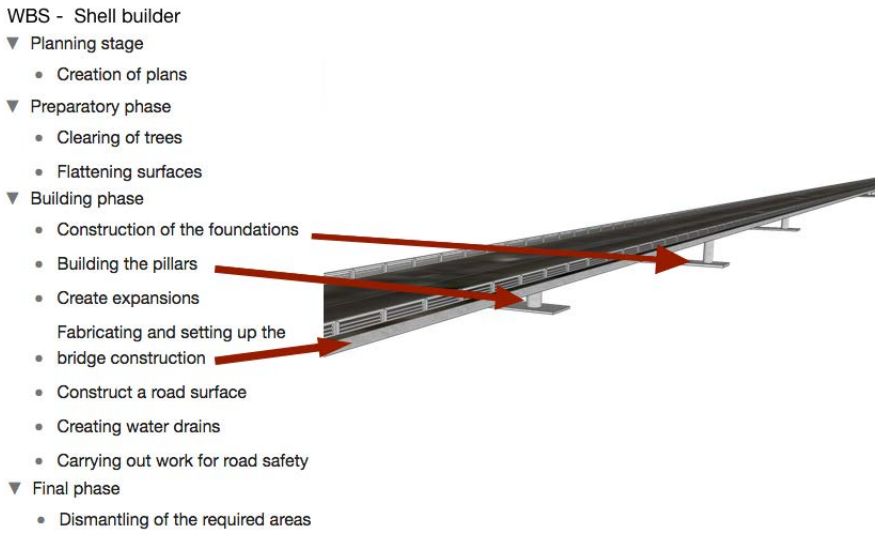


Figure 6.6: Assigned Objects: WBS - Shell builder

A separate model was uploaded for the electrician's discipline. As with the Work Breakdown Structure of the shell builder, the corresponding components are also assigned to the structure of the electrician. In the figure (Figure 6.7) the connections between components and structure are visible.

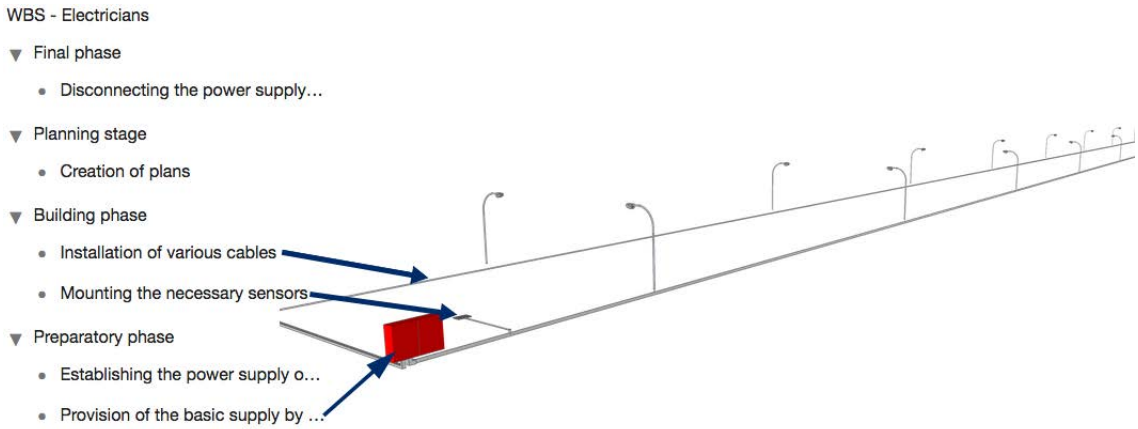


Figure 6.7: Assigned Objects: WBS - Electrican

In addition, the figure (Figure 6.8) shows that the nodes are assigned information about the time required and the possible costs.

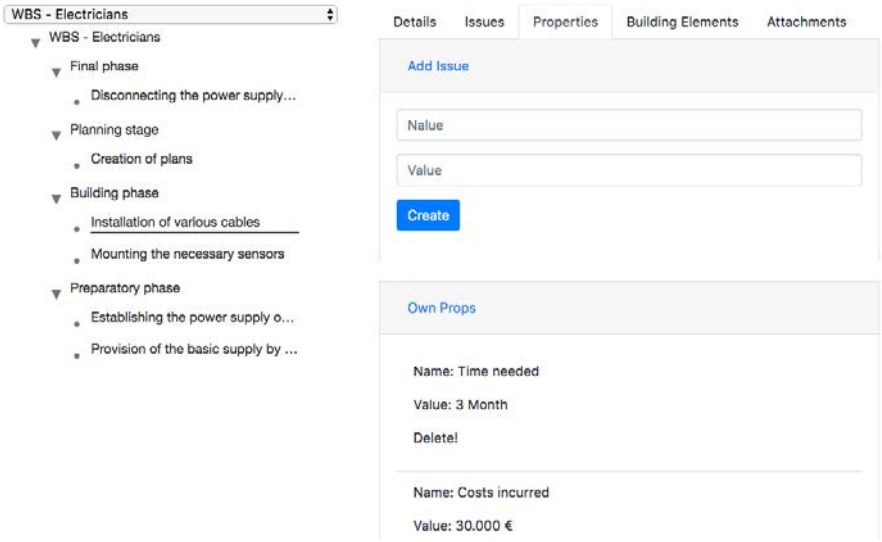


Figure 6.8: Properties: WBS - Electrican

Once all the required information and data have been assigned to each node, the nodes of two structures are connected to each other. To see the costs and also the time required for each function, the work packages are linked to the individual functions. This means that the individual nodes of the Work Breakdown Structure are linked to those of the System Breakdown Structure. The Figure 6.9 shows a list how the individual nodes of the System

Breakdown structure are connected to the Work Breakdown structure of the shell builder. Furthermore, the predicate is to be seen to describe the connection exactly and to generate a kind of RDF triple.

Subject Structure Node	Predicate	Object Structure Node
Pylons	Must be made	Building the pillars
Road surface	Must be made	Construct a road surface
Bridge Construction	Must be made	Fabricating and setting up the bridge construction

Figure 6.9: Connection List: SBS - WBS Shell Builder

These created links can also be displayed with a matrix, since the Work Breakdown structure can be arranged on the left side and the System Breakdown structure on the upper side. The Figure 6.10 shows the Work Breakdown structure of the electrician on the left side and the System Breakdown structure on the top side. The *Installation of various cables* node was connected to four different nodes. The laying of cables is necessary for the function of the sensors, the communication and the light. In addition cables are needed to provide the power supply.

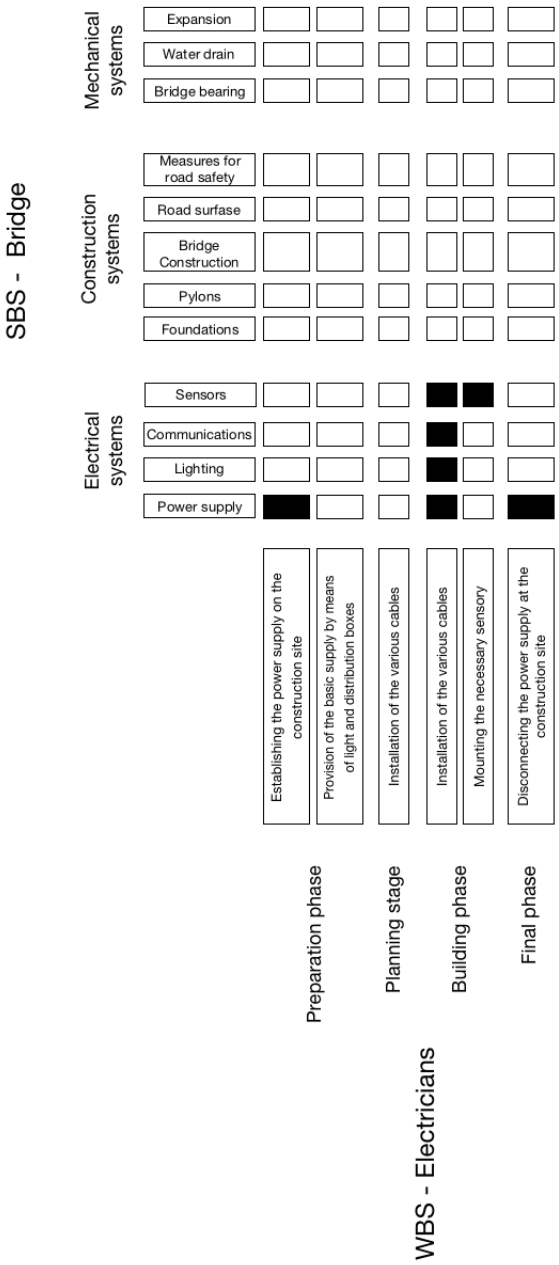


Figure 6.10: Matrix: SBS - WBS Electricans

By connecting two nodes, it is possible, as already described above, to see the information or data, such as costs or even assigned objects from the 3D model, from the other node. This establishes the actual cooperation, since the attached information does not have to be exchanged directly. Instead they are made available indirectly by connecting two nodes of the opposite side. The following figure (Figure 6.11) shows the properties of the node *Sensors* of the System Breakdown Structure. This shows that this node is connected to the *Installation of various cables* node. Furthermore, the user is shown the attached information of all linked nodes. Here, for example, the costs incurred and the time required are displayed as attributes.

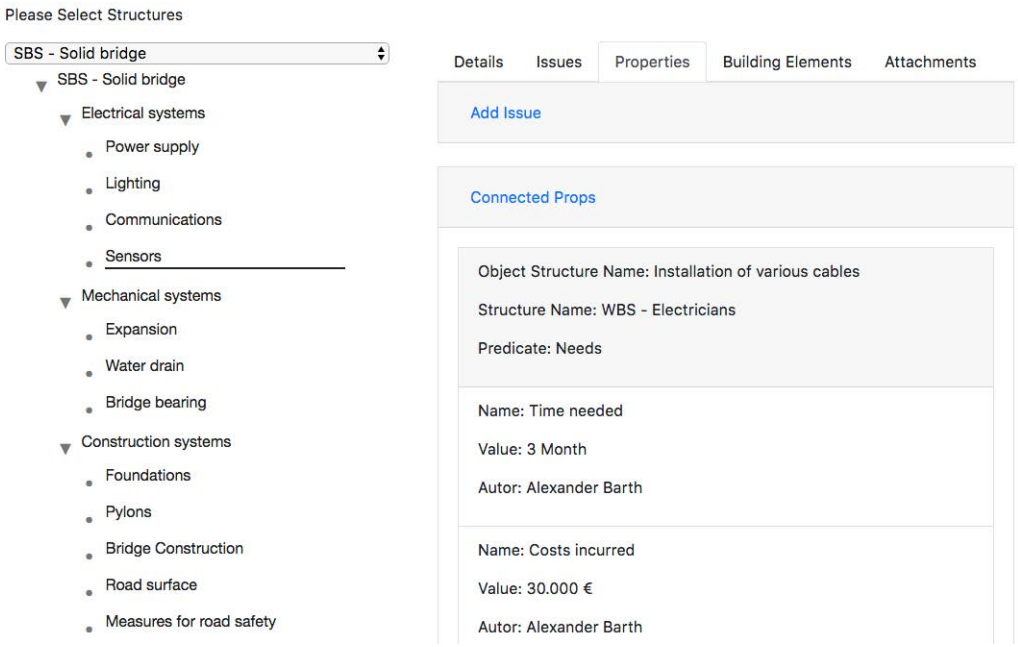


Figure 6.11: Properties: SBS

6.2 Evaluation

In this case study a 3D bridge model was used to divide the project for the construction of a new bridge into the individual Breakdown Structures. The next step was to append the required information and data to the respective nodes. The nodes of two structures were then connected to each other.

The time required to disassemble this project into individual parts is not big. However, attaching the individual information and data to the structure nodes involves more effort. The individual data must not only be stored digitally in the form of documents, but must also be individually assigned to the corresponding nodes in the form of attributes. Depending on how detailed and extensive the variety of data and information should be, a large amount of data comes together. If there is a large amount of data at a node, there is also the risk of losing the overview. To avoid this, additional sorting and grouping would be necessary. In addition to this risk, a larger and more complex project means that more storage space and performance is needed on the servers that provide this programmed tool.

The next step also involves a lot of time, namely connecting two structure nodes. As already described above, the effort for small and not so complex projects, as in this example, is limited. If this approach is applied to a larger and more complex project, a lot of time must be taken into account to connect all nodes together.

Another problematic point is that links between parts and structure could be lost if a new state of a 3D model is uploaded. This depends on whether the component ID changes for a new model or not. Nevertheless, it was shown that it is possible to work with several 3D models.

In summary, this is an interesting further development to the Common Data Environment. The data and information are stored in a work package or in a function of the project itself. This makes it possible to see the required data of a work package and function at a glance and, if necessary, to directly optimize it. However, it should not be forgotten that, depending on the size of the project, the time required is very long, which also increases the costs required to carry out the project. Nevertheless, time and costs can also be saved, as this approach makes more structured work possible.

Chapter 7

Development of a software prototype

7.1 Introduction

This chapter will describe how the approach of chapter 5 can be implemented by means of a prototype. At first it describes how to approach the implementation of the prototype and what functionality both the front-end and the RESTful API should have.

Afterwards the different programming languages, libraries and services will be described. Once the basics have been defined, the development from the prototype and the RESTful API will be described. In addition it will be described how and where the prototype is to be put online.

7.2 Approach to the implementation of the prototype

When implementing the solution approach in a prototype, consideration must be given to which functions should be made available to the user. As already described in the solution approach, the Cloud Platform BimPlus already provides some functions and only the new functions would need to be implemented. Because the source code of BimPlus is not openly available a new platform must be created. However, since the RESTful API of BimPlus is freely accessible, the existing functions can be used. To maintain this separation between front-end and RESTful API and to maintain the current state of the art in such architectures, a RESTful API is also programmed in addition to the front-end.

This has the advantage that the RESTful API can also be used independently from the front-end, as it should be done with the RESTful API of BimPlus in this case.

A web framework is to be used to create the prototype of the front-end. BimPlus is programmed with the Javascript Framework Ember. Currently there are several web frameworks that can be used to implement such a application. The author of this Masters thesis would like to use Angular for this purpose. This is a TypeScript based framework for single web page applications. It is developed by a community of individuals and companies led by Google and published as open source software. Angular is one of the best-known frameworks.

There are also several ways to implement a RESTful API. Since the author has a very good understanding of JavaScript, the Node.js-Framework Express is chosen. Express is implemented in a Node.js application. This makes it possible to quickly and easily provide a RESTful API.

The provision of the front-end and the RESTful API also play an important role in the selection of the system. Since both can be made quickly available online and securely with the container solution of Docker, the selection was only strengthened.

The functionality of the implemented system is described below.

7.3 Functionality

The user should be able to log in with the access data of BimPlus via an upstream login. Then he must select his team in which he wants to work. The RESTful API of BimPlus allows you to work in different teams at the same time. Once the user has selected a team, he must determine on which project he wants to work. Only by selecting the team and the project all functions are available to the user. The data is provided by the implemented RESTful API. Any data stored in BimPlus, such as user data or project data, is passed by the RESTful API of BimPlus to the newly implemented RESTful API. This has the advantage that no separate authentication is required.

After the team and project selection has been made, the user is redirected to a page on which the implemented functions are displayed.

To create a new structure in the project, which represents a very large role in the implementation of this Masters thesis, the user either clicks on the corresponding button on the overview page or he clicks in the top menu on Structures → New structure.

A new structure can be created on this newly loaded page. This must be entered in the text field in the form of a JSON structure. The schema according to which it must be written is displayed above the text field. Once the corresponding structures have been created, the user can decide whether he wants to create the various links between the structure nodes or whether he wants to define the properties or information at the corresponding structure nodes first.

If he wants to create the connections first, he must either click on the corresponding button on the overview page or in the Top Menu. The corresponding menu item in the Top Menu is: Structure → Connections. On this page the user has the option of selecting a structure from two menus. The structures are created by means of different possibilities: Via the method described above, via the RESTful API of BimPlus or via its cloud platform. If the user has selected two structures, they are displayed in the form of a tree structure. On this page the user can now select a node in each structure that he wants to connect to form a triple. In the following image (Figure 7.1) you can see that the user has selected only one structure.

To complete the triple he must specify the predicate in a text field. Unlike the RDF scheme, the user can enter more than one word. For example, this could be a complete sentence describing this connection.

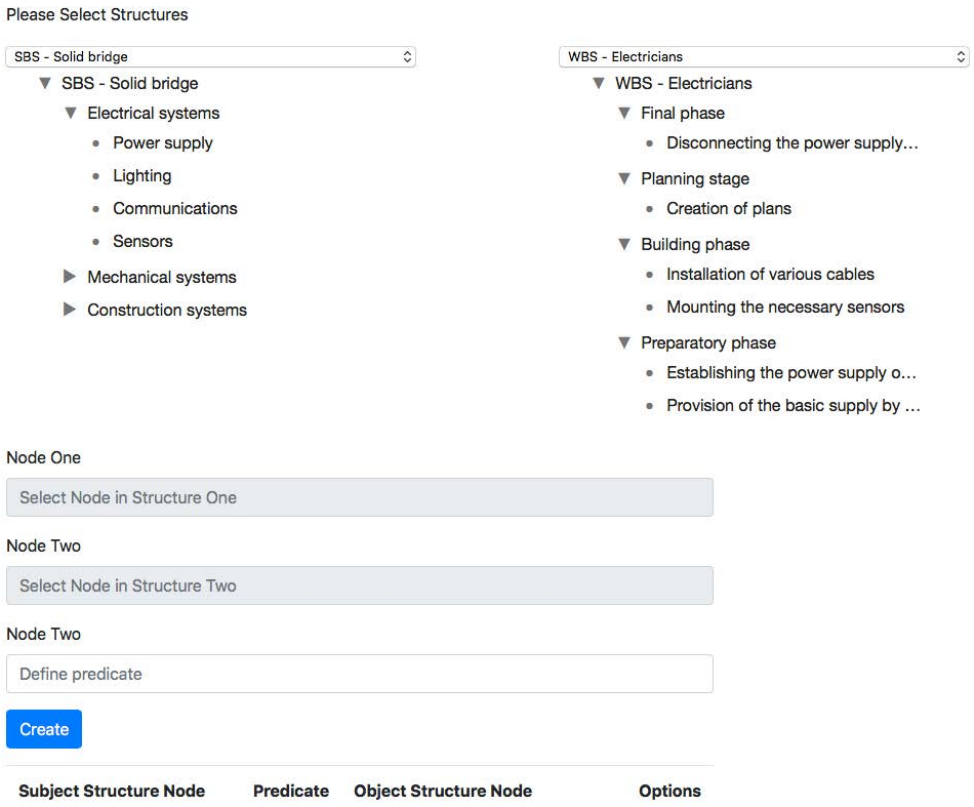


Figure 7.1: Connect Structure Nodes

As soon as the user clicks on Create, the existing links and the new connection between the two selected structures are displayed at the end of the page. If the user clicks on the subject or object in another connection, the tree structure opens at the appropriate position and the node is selected in the structure. This makes it possible to quickly and easily create a new connection for the selected node.

As already described in the solution approach, the connection between two nodes is only in one direction. If the creator wants the connection to be valid in the opposite direction, he must select the structures in the opposite direction.

As soon as all connections have been created, the user can use the Top Menu to switch to the page via which he can adjust the properties that are attached to a respective structure node. The corresponding menu item is called Structures → Properties. On this page you have to select the structure that you want to view or edit. As on the previous page, the structure is displayed in a tree structure. When a node is selected, the general properties of the selected point are displayed in the right panel. These are the name, ID, and type of the selected structure node. In the following image (Figure 7.2) you can see that the user has selected the *Construction Site Logistics* node.

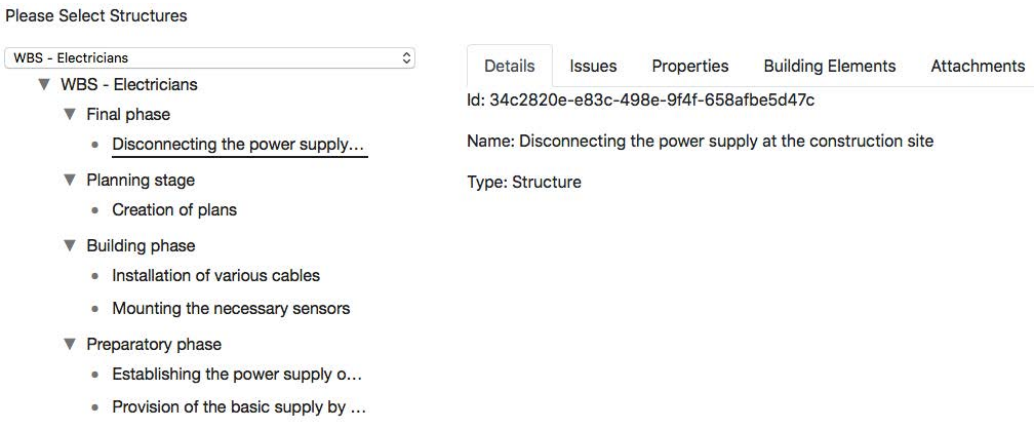


Figure 7.2: Properties of a Structure Nodes

The RESTful API of BimPlus allows you to specify different types for a node. The following types are available to the user: Apartment, Activity, Energy, Schedule, Structure. Depending on whether additional nodes are attached to the point, a different type can be selected during the creation to uniquely describe the node.

In addition to the display of general information, further menu tabs are available. The Issues button to the right of the details allows the user to add existing tasks to a node. If he wants to create a new task, he can do this in a dialog box. They also have the option of creating them via the Cloud Platform BimPlus.

If tasks have already been assigned to the selected node, they are displayed in this tab. If the node was linked to another node in the previous page, the tasks of the connected node are also displayed in this tab as long as a task has already been assigned to it.

Since these structures are also available to other team members, they can also bind tasks to the node. Nevertheless an existing task can only be removed by the producer himself.

A deletion request to remove a task, as described in the solution approach, is provided. It is however not yet available to the user.

The next tab allows the agent to freely add properties to a node. Two text fields are available. It uses the first text field to specify the name of the property and the second one to specify the value. As in the previous tab, the existing properties are displayed. The properties of the connected structure nodes are also displayed. The possibility to delete a property, is of course, also available.

The last two tabs are for adding parts that are present in the 3D model of the selected project and for adding files. If the user wants to connect a new part to the nodes, a dialog box opens and grays the remaining background of the page. In this dialog window all existing parts are sorted and structured according to part types in the form of a tree structure. If the user selects a component to add, the properties of this component are displayed. In addition the component is visualized in a viewer. This allows the user to view the selected object. As you can see in the following image (Figure 7.3) the user has selected a foundation to get the visualization and properties of this building element.

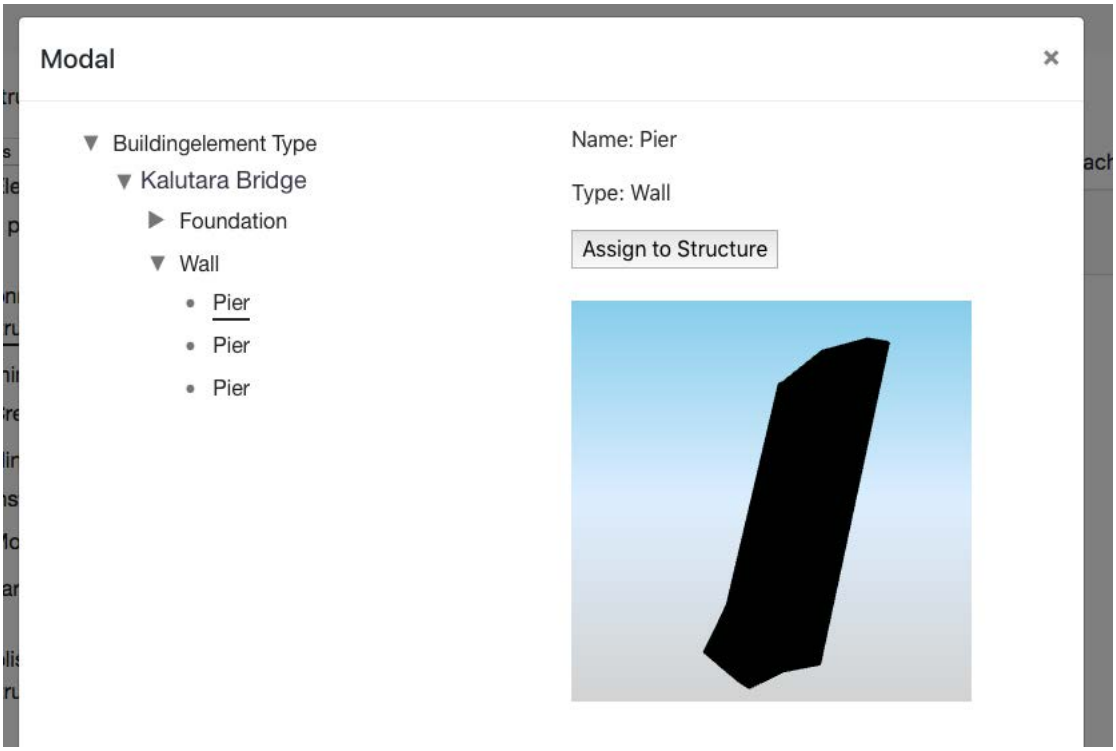


Figure 7.3: Assign of Building Elements to a Structure Nodes

The next step is to click on *Assign to Structure* to assign this building element to the structure node. Only when the user has finished adding building elements and closes the dialog window, he returns to the overview of the already and newly assigned building elements. The functionalities available to the previous tabs are also available at this point.

In the attachments tab, the user can upload files and bind them to the structure node. The attachments would be stored on the server where the RESTful API is hosted. As with the other tabs, the system displays the information, or in this case the files that are attached to the selected node and to those of the connected structures.

The following describes the programming languages and libraries used to create the prototype.

7.4 Programming language and libraries

7.4.1 JavaScript

JavaScript is a scripting language originally developed by Netscape in 1995 for dynamic HTML in web browsers to evaluate user interactions, modify, reload, or generate content to extend the capabilities of HTML and CSS. Today JavaScript is also used outside of browsers, for example on servers [57].

In the course of time some JavaScript frameworks have also developed. The best known frameworks are *ReactJS*[1], *Vue.js*[2].

Frameworks serve as a skeleton for single page applications, allowing developers to focus less on code structure or maintenance while focusing on creating complex interface elements [58].

7.4.2 TypeScript

TypeScript is a programming language developed by Microsoft. It is based on the future ECMAScript-6 standard. TypeScript is similar in structure to JavaScript. A particular advantage of TypeScript over JavaScript is its security type. When initializing variables a corresponding type can be specified. When programming with TypeScript it is possible to include JavaScript libraries.

If a program written in TypeScript is compiled, JavaScript files are available to the user again.

Like JavaScript, TypeScript provides the programmer with various Web frameworks. One of the best-known frameworks is Angular[3].

[1] https://reactjs.org
[2] https://vuejs.org
[3] https://angular.io/

7.4.3 Node.js

Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine. It uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. It is an open source server framework and it runs on various platforms like Windows, Linux, Unix, Mac OS X, etc.

Descriped before it uses JavaScript on the server. To write *Node.js* applications you can use *npm*. It is the package manager for JavaScript and the world's largest software registry.

7.4.4 EXPRESS

Express is a simple and flexible Node.js framework of web applications that provides numerous powerful features and functions for web and mobile applications. It can be installed using the *npm* Package Manager. Using countless HTTP utility methods and middleware functions, creating a powerful API is quick and easy [59].

The following (Listing 7.1) shows an example application for an API server. In the first line the library EXPRESS is implemented and included. Then a server instance of the variable *app* is assigned. The following line defines what happens when the user sends a *Get Request* to the API server. The server returns a *Hello World*.

Finally, the programmer must define which port the server should listen to. In this case it is port 3000.

```
1 const express = require('express');
2 const app = express();
3
4 app.get('/', (req, res) => res.send('Hello World!'));
5
6 app.listen(3000, () => console.log('Example app listening on port
    3000!'));
```

Listing 7.1: Hello World API

7.4.5 Docker

Docker brings the concept of apps to the server. A Docker container contains an application, but also all the resources that it requires at runtime. Docker is particularly well suited for cluster environments and data centers.

Docker is a kind of virtualization. The exact name is containerization. It is an implementation of container technology which is characterized by particularly user-friendly features and has made the term container popular as an alternative to virtual machines in the first place.

A container combines a single application with all its dependencies, such as libraries, utilities and static data, into one image file without including a complete operating system. Therefore containers can be compared to a lightweight virtualization.

Containers require fewer resources than virtual machines because they do not need to start their own operating system. Instead they run in the context of the host operating system instead. Nevertheless, the containers are sealed off from each other and from the host system [60].

To start a container in Docker, a finished image or a self-compiled image is required. On the Docker website you will find some ready-made images that are published by companies and other users.

A self-written image can be read in the Listing 7.2. The creator must first define a basic image on which he wants to build. The user can then define that required packages are to be installed or that files are to be copied.

```
1 FROM httpd
2 COPY ./public-html/ /usr/local/apache2/htdocs/
```

Listing 7.2: Example docker file

Once the image is defined, it can be created and executed by the Docker instance. So that the container can also be reached from outside, corresponding ports must be defined when the container is executed.

7.5 Building the Prototype

As already described above, the prototype consists of two systems, the front-end and a RESTful API. The following describes how the front-end and the RESTful API are programmed.

7.5.1 Front-end

The Web Framework Angular is structured according to the MVC scheme. MVC stands for Model View Controller. This means that the individual components are divided into the corresponding areas. The Model area includes the provision of data. The View area displays the data and the controller provides the functions necessary to handle the available files.

The individual pages are created as components in Angular. This means that a large number of components are created at the beginning. The services are then created. These contain the requests to the RESTful API. For a better overview different services are created according to the functionality. The following services are created:

- service for authentication with the RESTful API
- service for the requests passed through the RESTful API of BimPlus
- Service for processing connections between two structures and
- Service for managing the properties and information of a structure node

These services can be integrated into the components as required. For example, all services except that for authentication are included in the component for viewing and adding properties and information to a structure node. To ensure that the required data is available directly after the individual components have been called, it is loaded automatically when the services are called. Angular provides the function *OnInit* for this.

The open source toolkit Bootstrap is integrated in order to have little to do with the design of the application. This toolkit helps to develop with HTML, CSS and JS. With Sass variables and mixins, a responsive grid system, extensive pre-built components and powerful plug-ins based on jQuery, applications can be created quickly and easily.

Each individual page or component consists of three files: The controller, which is programmed in TypeScript, the view in the form of an HTML document and the corresponding CSS file to adapt the style of the view.

Angular provides the possibility that a component can be integrated into another component. This makes it possible to make the source code clearer and to store recurring functions. This method is used in the view of the properties and information of a structure node, since the different tabs are structured almost identically.

As soon as the application is programmed, it is compiled and put online with the help of Docker. The section 7.6 describes in detail how this is implemented with the container solution.

The following chapter describes how the services are provided with the data via a RESTful API. It also describes how the RESTful API is structured.

7.5.2 RESTful API

The RESTful API is the heart of the prototype. It processes and provides all kinds of information that the individual components require. This encapsulation is especially carried out because the RESTful API can be used independently.

As described above, the Node.js framework EXPRESS is used to create the RESTful API. Before the actual creation of the RESTful API can be started, the required packages and libraries must be included.

In addition to EXPRESS, the following libraries, for example, are included: MySQL to access a MySQL database, Passport to provide authentication, Redis to temporarily store the data of a session or *body-parser* to process the data sent to the RESTful API.

To get a better overview when creating the RESTful API, the routers and controllers are separated from each other. The files for the routers describe which function of the controller is to be called via which address.

The following code example (Listing 7.3) shows the router for processing the connections between two structures.

```
1  const express = require('express');
2
3  const router = express.Router();
4  const structureConnect = require('../controllers/
       structureConnectController');
5
6  router.route('/:rootSubjectStructureId')
7      .get(structureConnect.getConnections)
8      .post(structureConnect.postConnection);
9
10 router.route('/:subjectStructureId/:connectionId')
11     .get(structureConnect.getConnection)
12     .put(structureConnect.editConnection)
13     .delete(structureConnect.deleteConnection);
14
15 module.exports = router;
```

Listing 7.3: Router File for Structure Connecitons

First, the EXPRESS package must be included in order to access the functions it contains. Then the corresponding controller must be integrated, which contains the respective functions.

The following lines define which function of the controller can be called via which address. A distinction is made between which type of request is sent: POST, GET, DELETE. POST

means that data is sent to the RESTful API. GET returns data to the client and DELETE deletes information. *:rootSubjectStructureId* is a placeholder. As the name already indicates, the id of the subject structure is expected here.

A separate function must be created in the controller for each different type of request. A corresponding example for a function or how a controller is constructed shows Listing 7.4.

```
1  const db = require('../../config/db');
2  const boom = require('boom');
3  const uuidv4 = require('uuid/v4');
4  const helper = require('../helper/helperFunctions');
5
6  exports.getConnections = function getConnections(req, res, next) {
7      if (!req.params.rootSubjectStructureId || helper.checkValidUUID(
           req.params.rootSubjectStructureId) === false) {
8          return next(boom.badRequest('Please enter a valid root
               structure id'));
9      }
10
11     db.get().query('SELECT * FROM structureConnect WHERE
           rootSubjectStructureId = ?', [req.params.rootSubjectStructureId
           ], (err, rows) => {
12         if (err) return next(boom.badImplementation(err));
13         res.json(rows);
14     });
15 }; ....
```

Listing 7.4: Controller File for Structure Connecitons

As with the router, the required packages must first be imported. In this case you have to interact with the MySQL database, so you have to import it with the function from the first line 1. The *boon* package helps to process query errors. The *uuuid/v4* package provides a UUID in version 4. Each time this function is called, a new UUID is returned. To avoid having to rewrite recurring functions every time, they are swapped out in a helper function and imported with the function in line 4. The function shown in line 6 returns all existing connections of a certain structure. The function can access the ID of the subject structure via the placeholder in the router. To avoid incorrect requests, the system first checks whether an ID has been transferred and whether it is valid. Only then is a query sent to the database.

So that not all functions have to be reprogrammed, some of them are reused which are already available via the RESTful API of BimPlus. This includes, for example, authentication. The created RESTful API does not provide its own user administration. Rather it uses those of BimPlus. If the user logs in via the RESTful API, the user name, password and application ID are forwarded to BimPlus. If a token is returned by BimPlus, a new token in the form of a JWT token is also created. This contains the BimPlus token and user data.

With this newly created token, the user can authenticate himself to the RESTful API and also to that of BimPlus. The following short excerpt (Listing 7.5) shows the function for querying all existing team members. The information is retrieved via the RESTful API of BimPlus.

```
1  exports.getTeams = function getTeams(req, res, next) {
2      const allplanData = req.user;
3
4      if (!req.session.allplanData || !req.session.allplanData[
           allplanData.body.client_id]) {
5          return next(boom.unauthorized('Please login again'));
6      }
7      const options = {
8          url: 'https://api-dev.bimplus.net/v2/teams',
9          json: true,
10         headers: {
11             'Content-Type': 'application/json',
12             Authorization: `BimPlus ${allplanData.allplanToken}`,
13         },
14     };
15     request(options, (error, response, body) => {
16         if (error) {
17             return next(boom.badImplementation(error));
18         }
19         res.json({ success: true, data: body });
20     });
21 }
```

Listing 7.5: Controller File for BimPlus Queries

If the user sends a request to the server, the token sent is interpreted and stored in the variable *req.user*. To send an HTTP request to a RESTful API, the package *request* is used. The necessary information is defined in the variable *options*, including the token in line 12, which is necessary to authorize the BimPlus RESTful API.

As soon as all functions are written in the respective controllers and the corresponding routers, the RESTful API can be published. The container solution Docker is also used for this purpose.

7.6 Deploy and Hosting

The container solution of Docker is used for publishing the front-end and the RESTful API. The subsection 7.4.5 already explained the idea behind Docker.

This solution is selected because it makes it possible to quickly make Web services available online in a simple manner. All you need is a *Dockerfile*. The Docker container solution takes care of the rest. No server needs to be installed separately.

Because the individual containers are separated from each other, there is more security against attacks from outside. The MySQL and Redis containers are connected directly to the RESTful API container. The two containers have no real interface to the outside world. The RESTful API could also be connected to the front-end. This is not done because the RESTful API should work independently.

```
1 docker run -d -P --name web --link db:db training/webapp python app.py
```

Listing 7.6: Bash Command to execute a docker container

Listing 7.6 shows both how a container can be executed via the command line and how a container, in this case *db*, is linked into that container.

The Docker container for the RESTful API is created with the following *Dockerfile* (Listing 7.7):

```
1 FROM node
2 WORKDIR /usr/src/app
3 COPY package*.json ./
4 RUN npm install --only=production
5 COPY . .
6 EXPOSE 3000
7 CMD [ "npm", "start" ]
```

Listing 7.7: Docker file RESTful API server

In the first line of the *Dockerfile* the base image is defined. In this case it is the *node Image*. In the next line, the directory in which the server is to be executed is defined. For performance reasons, the file in which the server defines which packages it needs is copied separately and then the packages are installed with the command in line 4. Only then are the remaining files copied to the server.

The reasons for this separation are that Docker temporarily stores individual steps for creating the image. As long as the file *package.json* does not change, the state up to this command can be taken over from the old image. This makes it possible to create an image faster.

Before this container can be started, the containers for MySQL and Redis must be started. A finished *Dockerfile* is available for both containers.

Both containers are started with the following command (Listing 7.8):

```
1 docker run --name db -e MYSQL_ROOT_PASSWORD=my-secret-pw -d mysql
2 docker run --name redis -d redis
```

Listing 7.8: Bash Command to execute a mysql and redis docker container

When starting the MySQL Container, a root password is also defined. Without it, it is not possible to access the database.

Compiling and starting the front-end is also done with Docker. This is based on an existing manual[4]

Putting the front-end online is done by a *Nginx server*. For this a separate configuration file (Listing 7.9) must be created in the project:

```
server {
  listen 80;
  location / {
    root /usr/share/nginx/html;
    index index.html index.htm;
    try_files $uri $uri/ /index.html =404;
  }
}
```

Listing 7.9: Nginx configuration

Then it is possible to start the container with the following (Listing 7.10) *Dockerfile*:

```
# Stage 0, based on Node.js, to build and compile Angular
FROM node as node
WORKDIR /app
COPY package.json /app/
RUN npm install
COPY ./ /app/
ARG env=prod
RUN npm run build -- --prod --environment $env

# Stage 1, based on Nginx, to have only the compiled app, ready for
    production with Nginx
FROM nginx
COPY --from=node /app/dist/ /usr/share/nginx/html
COPY ./nginx-custom.conf /etc/nginx/conf.d/default.conf
```

Listing 7.10: Docker file RESTful API server

[4] https://medium.com/@tiangolo/angular-in-docker-with-nginx-supporting-environments-built-with-multi-stage-docker-bu

The special feature of this *Dockerfile* is that the front-end is first compiled and then put online with the *Nginx Server*.

The advantage of Docker Containers is the simple and fast provision of the entire system as well as that the API server can be arbitrarily scaled by Docker's own means in case performance problems occur.

The Docker instance required for the containers was installed on the DigitalOcean cloud computing platform. This platform was chosen because the author already had a corresponding server there.

Chapter 8

Summary, Conclusion and Outlook

8.1 Summary

The topic of this Master's thesis is the development of a holistic data management in civil engineering with the help of methods of the System Engineering.

After the introduction in the chapter 2 the digital methods in the building environment which are necessary for the implementation of this Masters thesis were explained. The term BIM was explained and how it is currently being implemented in infrastructure projects. It has been described that *Federal Ministry of Transport and Digital Infrastructure* has developed a step-by-step plan to gradually implement more and more projects in infrastructure construction with BIM. This phased plan provides for the mandatory use of BIM in federal infrastructure projects from 2020. To further promote this implementation, the BIM4INFRA working group was also set up.

As the exchange of data through larger projects becomes more and more important, the topics IFC and IFC Model View Definition were also explained. Since a uniform definition of components is required for file exchange, the object catalog OKSTRA was also explained.

In the following chapter (chapter 3) the three technical bases Semantic Web, Classification and System Engineering were described. Semantic Web describes the idea of linking data and information via the web to a large network. To better understand this idea, the term ontology was described in this chapter. Originally it comes from philosophy, but it can also be applied to many other areas. In computer science, for example, as already described in the technical basics, an ontology is often used to formalize and use as a meaning of information. The co-founder of the Internet Berners-Lee has defined various requirements for the use of the term ontology in computer science.

Since ontologies can be coded differently in computer science, the most important methods are also described in this chapter. These include RDF and Web Ontology Language. Three components are required to link two pieces of information in RDF. The two information are, respectively, the subject and the object. The third component is the predicate. These three components form an RDF triple. The Web Ontology Language is an extension of RDF and is available in three versions. These differ depending on the complexity and diversity of the information.

The term Semantic Web also includes the term Linked Data. This method describes connecting information across the Internet. Each connection requires several components according to the Linked Data methodology. These include RDF and URI. URI stands for Universal Resource Identifier and is an identifier for corresponding information. Along with Linked Data there is in addition Open Data. Open Data is information that is publicly accessible, but not necessarily linked to each other. The combination of Linked Data and Open Data is Linked Open Data.

Through the development of Linked Open Data various working groups have developed. One of them develops an extension of OWL to be able to use the IFC scheme in connection with Semantic Web. This extension is called ifcOWL. Not only about this but also about SPARQL was written in this chapter. SPARQL is similar to MySQL and is a query language for RDF.

The next major technical basis reported in the chapter 3 is classification. Germany has its own standard on classification systems. In addition to this standard, there is also the International Standard ISO 12006-2:2015, both of which are described in this chapter. In connection with DIN, the structure of a classification system and how the different terms are defined in a classification system were described. Furthermore, it was shown how a classification of office machines could look like.

The International Standard ISO 12006-2:2015 has shown that it contains tables to help develop classification systems for the construction industry. There are different tables for the different object classes, divided according to different views. In addition, this standard describes how the individual object classes are related. In addition to this standard, the OmniClass and Uniclass are also available. Both are classification systems based on an International Standard.

The OmniClass is used by the North American architecture, engineering and construction industries throughout the entire life cycle of the structure. The UniClass, on the other hand, is used by UK-based construction companies.

Last important technical basis represents System Engineering. It takes up the previous basis, since it describes that large and extensive projects are to be classified into structures. The best known method is to divide projects into Breakdown Structures. Different types of Breakdown Structures are available to the user, for example: The System Breakdown Structure, the Work

Breakdown Structure or the Organization Breakdown Structure. The System Breakdown Structure classifies the project according to the functions. As the name suggests, the Work Breakdown Structure describes the system using work packages. It structures the project on the basis of the work steps that are necessary for completion.

In chapter 4 current developments and research projects concerning the methods used were described. This includes, for example, the current development status of IFC or topics of the Semantic Webs.

The chapter 5 describes how a holistic data management can be implemented with the help of System Engineering. This makes it particularly important that data and information can be distributed and exchanged freely and without loss throughout the project. The solution approach in this Master's thesis is a combination of the two components System Engineering and Semantic Web as already mentioned.

As described above, projects can be classified in Breakdown Structures. This helps to highlight individual points in the project. If individual structure nodes of the Breakdown Structure are connected to each other, an extensive network of information is created. When connecting individual structures the technique of Semantic Web is applied. One structure node represents the subject and the other node represents the object. For the RDF schema, it is important that the two nodes are additionally described with a predicate. This results in a valid RDF triple.

Additional information is assigned to the individual nodes. The information can be components of the 3D model of the project or also file, attributes or tasks. Because individual structure nodes are connected to each other, the various information of the object node can be viewed at the subject node. The connection between two nodes is only in one direction and not in both, since the opposite connection could have a different meaning. This means that an additional connection must be created to establish a bidirectional connection. If information is changed or added to a node, it does not have to be exchanged from one structure to another.

After the solution approach has been presented in chapter 5, it is checked for applicability in this chapter with the help of a 3D model. For this review it is assumed that this model will have to be newly build. The new construction project in the form of this 3D model will initially be divided into three Breakdown Structures: one System Breakdown Structure and two Work Breakdown Structures. The two Work Breakdown structures are for the two construction companies: Electrician and shell builder. In a productive application, many more construction companies will carry out the project. Nevertheless an exemplary focus was placed on these two construction companies.

After division into the structures, the information and data is assigned to the individual nodes. These are both components from the 3D model, as well as information, for example,

about the costs or the time required. Finally, the individual nodes are interconnected so that information can be exchanged. In addition to the usability check, evaluation also has been done in this chapter.

In the last chapter (chapter 7) the implementation of the solution approach is described by means of a prototype. First, the planning of the conversion is described, namely that there should be a RESTful API together with a front-end. The RESTful API should include functions from the RESTful API of BimPlus from Allplan. Because the BimPlus code is not openly accessible, no prototype can be implemented here. For the RESTful API only an useraccount and an application id for BimPlus are required, after which all implemented functions can be accessed.

The RESTful API was implemented as a Node.js application with the EXPRESS framework. This combination allows to create a well structured and understandable RESTful API. The implemented functions in the RESTful API of BimPlus are passed through by the new RESTful API. Only the response was adapted according to requirements. The front-end was implemented with the Angular Framework, since this is one of the most current and well-known frameworks based on TypeScript. The individual web pages were programmed as *component*. Because Angular Framework is for single page applications, the user can navigate smoothly through the prototype without reloading the web page. Data processing has been separated from the front-end and takes over the RESTful API. This has the advantage that the RESTful API can work independently.

In addition to the planning and implementation of the prototype, the libraries and services used were also described.

8.2 Conclusion

In this Master's thesis, a prototype was used to show how a holistic data management in civil engineering can function with the help of the Systems Engineer methods. To do this the project must be broken down or structured into its individual parts. If all information, such as attributes or components of the 3D model are assigned to the corresponding nodes, a uniform and sorted system is created. If the user connects a structure node to a node of another structure, an all-encompassing network of information is gradually formed. As the information no longer has to be exchanged directly, no information can be lost, unless data is lost due to a technical problem. In this case there are backups that restore the corresponding data. Data loss or falsification of the information is not possible due to human error.

An important factor is that the project must be classified into individual Breakdown Structures and the information must be assigned to the nodes. Once the individual structure nodes are connected according to their dependencies, data can be distributed without loss.

As already mentioned in the evaluation of the case study, this procedure is possible for small projects without problems. As these projects grow and become more complex, this approach is no longer easily applicable as it is too costly. The individual Breakdown Structures would be too large to be able to see them right away. Adding information and data would also be too time-consuming.

However, once building individual structures and assigning information can be automated, it is a very good alternative to a common data environment, since the required information is much more precisely linked to a work package or function of the project.

8.3 Outlook

As already mentioned, the individual Breakdown Structures must be connected manually. This is no longer possible for larger projects. In this context, it is conceivable that a possibility could be created to automatically interconnect the individual nodes. This automation could gradually become more intelligent through existing connections.

Another possibility to extend this prototype is the native embedding of Allplan's Cloud Platform BimPlus, since some functions already exist in this Cloud Platform and are only passed through by the new RESTful API.

To facilitate the setup of the Breakdown Structures, a kind of template function would also be conceivable. This provides the user with a template to classify the project. As long as the structure is general, these structures could also be copied from one project to the next.

In the solution approach, the topic of IFC Model View Definition has already been addressed. This topic was only partially integrated into the prototype. This topic would be another way to expand the project.

Appendix A

Disk

The following documents are on the enclosed disk:

- This thesis [PDF]
- Files of the prototype (compressed) [ZIP]
- Created Breakdown Structures [JSON]

Bibliography

[1] Bundesministerium für Verkehr und digitale Infrastruktur, "Stufenplan Digitales Planen und Bauen," 2015.

[2] T. Liebich, "IFC overview presentation," http://www.bre.co.uk/filelibrary/events/BREEvents/BIMConferenceSeason/DeliveryofIFC/2-Thomas-Liebich.pdf, accessed: 2018-06-09.

[3] A. Benevolenskiy, *Ontology-based modeling and configuration of construction processes using process patterns.* Technischen Universität Dresden, 2015.

[4] J. Beetz, J. Van Leeuwen, and B. De Vries, "IfcOWL: A case of transforming EXPRESS schemas into ontologies," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AIEDAM*, vol. 23, no. 1, pp. 89–101, 2009.

[5] OmniClass, "Tables - OmniClass."

[6] R. Züst, *Einstieg ins Systems Engineering: optimale, nachhaltige Lösungen entwickeln und umsetzen*, 3rd ed. Zürich: Verl. Industrielle Organisation, 2004, oCLC: 76620803.

[7] D. Lock, *Project Management*, 10th ed. Burlington, VT: Gower, 2013.

[8] J. Amann and A. Borrmann, "Analyse zur Erstellung eines 3D-BIM-konformen Straßenentwurfs auf Basis von IFC-Alignment aus einem OKSTRA konformen 2D- Straßenentwurf unter Einsatz der TUM Open Infra Platform in Verbindung mit der OKSTRA-Klassenbibliothek," Lehrstuhl für Computergestützte Modellierung und Simulation, Tech. Rep., 2016.

[9] J. Amann, F. Schöttl, D. Singer, M. Kern, A. Widner, P. Geisler, D. Below, H. Hecht, N. Gupta, A. Mustafa, v. Markĭc, and A. Borrmann, "TUM Open Infra Platform 2017," 2016.

[10] Nova Spivack, "Web 3.0 - The Best Official Definition Imaginable," http://www.novaspivack.com/technology/web-3-0-the-best-official-definition-imaginable, accessed: 2018-06-09.

[11] Randall Hunt, "Amazon Neptune," http:://aws.amazon.com/de/neptune/, accessed: 2018-06-09.

[12] Lehrstuhl für Computergestützte Modellierung und Simulation der TU München, "SEEBridge," https://www.cms.bgu.tum.de/de/31-forschung/projekte/473-seebridge, accessed: 2018-06-09.

[13] W3C, "SPARQL 1.1 Query Language," https://www.w3.org/TR/sparql11-query/, accessed: 2018-06-09.

[14] C. Zhang and J. Beetz, "Querying Linked Building Data Using SPARQL with Functional Extensions," *eWork and eBusiness in Architecture, Engineering and Construction (ECPPM 2016) - Proceedings of 11th European Conference on Product and Process Modelling. - S.l. : s.n., 2016*, no. October, p. 9, 2016.

[15] R. Battle and D. Kolas, "Enabling the Geospatial SemanticWeb with Parliament and GeoSPARQL," *Semantic Web Journal*, vol. 0, no. 0, pp. 1–17, 2012.

[16] D. Thasarathar, "Die Digitalisierung der Bauindustrie," https://www.autodesk.de/campaigns/the-power-of-digital-for-construction, 2016, accessed: 2018-06-09.

[17] A. Borrmann, M. König, C. Koch, and J. Beetz, *Building Information Modeling: Technologische Grundlagen Und Industrielle Praxis*, ser. VDI-Buch. Springer Vieweg, 2015.

[18] ARGE BIM4INFRA2020, "BIM4INFRA2020," http://bim4infra.de/, accessed: 2018-06-09.

[19] Bundesministerium für Verkehr und digitale Infrastruktur, "Umetzung des Stufenplans Digitales Planen und Bauen," p. 28, 2017.

[20] buildingSMART e.V., "Besser Planen mit BIM," https://www.buildingsmart.de, accessed: 2018-06-09.

[21] buildingSMART, "Model View Definition Summary," http://www.buildingsmart-tech.org/specifications/ifc-view-definition, accessed: 2018-06-09.

[22] G. Costa, "Integration of building product data with BIM modelling," Ph.D. dissertation, Universitat Ramon Llull. ETS d'Arquitectura La Salle, Barcelona, 2017.

[23] buildingSMART, "mvdXML 1.1," http://www.buildingsmart-tech.org/specifications/mvd-overview/mvdxml-releases/mvdxml-1.1, accessed: 2018-06-09.

[24] OKSTRA, "Einführung," http://www.okstra.de/, accessed: 2018-06-09.

[25] W3C, "Semantic Web," https://www.w3.org/standards/semanticweb/, accessed: 2018-06-09.

[26] E. N. Zalta, "The Stanford Encyclopedia of Philosophy," 2013.

[27] J. Busse, B. Humm, C. Lübbert, F. Moelter, A. Reibold, M. Rewald, V. Schlüter, B. Seiler, E. Tegtmeier, and T. Zeh, "Was bedeutet eigentlich Ontologie?: Ein Begriff aus der Philosophie im Licht verschiedener Disziplinen," *Informatik-Spektrum*, vol. 37, no. 4, pp. 286–297, 2014.

[28] T. Berners-Lee, "The Semantic Web as a language of logic," https://www.w3.org/DesignIssues/Logic.html, accessed: 2018-06-09.

[29] W3C, "RDF - Semantic Web Standards," https://www.w3.org/2001/sw/wiki/RDF, accessed: 2018-06-09.

[30] OWL Working Group, "OWL - Semantic Web Standards," https://www.w3.org/OWL/, accessed: 2018-03-14.

[31] T. Berners-Lee, "Linked Data - Design Issues," https://www.w3.org/DesignIssues/LinkedData.html, accessed: 2018-06-09.

[32] Ontotext, "What are Linked Data and Linked Open Data?" https://ontotext.com/knowledgehub/fundamentals/linked-data-linked-open-data/, accessed: 2018-06-09.

[33] DBpedia, "About dBpedia," http://wiki.dbpedia.org/about, accessed: 2018-06-09.

[34] P. Pauwels, S. Törmä, J. Beetz, M. Weise, and T. Liebich, "Linked Data in Architecture and Construction," *Automation in Construction*, vol. 57, no. September, pp. 175–177, 2015.

[35] P. Pauwels, T. Krijnen, W. Terkaj, and J. Beetz, "Enhancing the ifcOWL ontology with an alternative representation for geometric data," *Automation in Construction*, vol. 80, pp. 77–94, 2017. [Online]. Available: http://dx.doi.org/10.1016/j.autcon.2017.03.001

[36] P. Pauwels, "ifcOWL – openBIMstandards.org," http://openbimstandards.org/standards/ifcowl/, accessed: 2018-06-09.

[37] Deutsche Norm, "Klassifikationssysteme DIN 32705," 1987.

[38] ISO 12006-2, "Building construction - Organization of information about construction works - Part 2: Framework for classification," 2015.

[39] OCCS Development Committee, "OmniClass^{TM} Introduction and User's Guide," http://www.omniclass.org/tables/OmniClass{_}Main{_}Intro{_}2006-03-28.pdf, accessed: 2018-06-09.

[40] A. Benevolenskiy, K. Roos, P. Katranuschkov, and R. J. Scherer, "Ontologiebasiertes Framework für Referenzprozesse und Prozesskonfiguration," in *Informationssysteme im Bauwesen 1*, ser. VDI-Buch. Springer Vieweg, Berlin, Heidelberg, 2014, pp. 273–287.

[41] Designing Buildings Wiki, "NBS," https://www.designingbuildings.co.uk/wiki/NBS, accessed: 2018-06-09.

[42] J. Gelder, "The principles of a classification system for BIM: Uniclass 2015," *49th International Conference of the Architectural Science Association 2015*, vol. 1, pp. 287–297, 2015.

[43] S. Delany, "Uniclass 2015," https://toolkit.thenbs.com/articles/classification#classificationtables, accessed: 2018-06-09.

[44] R. Haberfellner, O. L. d. Weck, E. Fricke, and S. Vössner, *Systems Engineering: Grundlagen und Anwendung*. Orell Füssli Verlage, 2015.

[45] G. Drews and N. Hillebrand, *Lexikon der Projektmanagement-Methoden*, ser. GPM Projektmanagement bei Haufe. Haufe-Mediengruppe, 2010. [Online]. Available: https://books.google.de/books?id=Z06ItH4vb0gC

[46] D. Lock, *Project Management*. Gower, 2013. [Online]. Available: https://books.google.de/books?id=9b3XCQAAQBAJ

[47] M. Rich, "Defining Project Breakdown Structures," http://community.vitechcorp.com/index.php/defining-project-breakdown-structures.aspx, accessed: 2018-06-09.

[48] buildingSMART, "Summary of IFC Releases," http://www.buildingsmart-tech.org/specifications/ifc-releases/summary, accessed: 2018-06-09.

[49] J. Amann and A. Borrmann, "Open BIM for Infrastructure - mit OKSTRA und IFC Alignment zur internationalen Standardisierung des Datenaustauschs," *Tagungsband zum 6. OKSTRA-Symposium*, no. April 2014, pp. 1–10, 2015.

[50] D.-G. Lee, J.-Y. Park, and S.-H. Song, "Bim-based construction information management framework for site information management," *Advances in Civil Engineering*, vol. 2018, p. 15, 2018.

[51] J. Park and H. Cai, "Wbs-based dynamic multi-dimensional bim database for total construction as-built documentation," *Automation in Construction*, vol. 77, pp. 15 – 23, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0926580517300663

[52] S.-L. Fan, C.-H. Wu, and C.-C. Hun, "Integration of cost and schedule using bim," *Journal of Applied Science and Engineering*, vol. 18, pp. 223–232, 2015.

[53] Mobi, "User Manual," https://mobi.inovexcorp.com/docs/, accessed: 2018-06-09.

[54] Open Geospatial Consortium, "GeoSPARQL - A Geographic Query Language for RDF Data — OGC," http://www.opengeospatial.org/standards/geosparql, accessed: 2018-06-09.

[55] Jonathan Markow, "Fedora Repository Home," https://wiki.duraspace.org/display/FF, accessed: 2018-06-09.

[56] C. Johnson, "Einführung in das PANDORA Linked Open Data Framework," https://www.researchgate.net/publication/304560904_Einfuhrung_in_das_PANDORA_Linked_Open_Data_Framework, accessed: 2018-06-09.

[57] S. Koch, *JavaScript: Einführung, Programmierung und Referenz - inklusive Ajax*, 5th ed., ser. iX-Edition. Heidelberg: dpunkt-Verl, 2009, oCLC: 391434094.

[58] E. Korotya, "5 Best JavaScript Frameworks in 2017," https://hackernoon.com/5-best-javascript-frameworks-in-2017-7a63b3870282, accessed: 2018-06-09.

[59] StrongLoop, "Express - Node.js web application framework," http://expressjs.com/, accessed: 2018-06-09.

[60] Stephan Augsten, "Was sind Docker-Container?" https://www.dev-insider.de/was-sind-docker-container-a-597762/, accessed: 2018-06-09.

Declaration of Authorship

With this statement I declare that I have independently completed this Masters thesis. The thoughts taken directly or indirectly from external sources are properly marked as such. The submitted code is entirely mine and any part which has not been written by me is properly marked.

This thesis was not previously submitted to another academic institution and has also not yet been published.

München, 10. July 2018

Barth, Alexander

Barth, Alexander