# Automated validation of building models against legislation using Linked Data

## focussing on accessibility

Emma Nuyts

2021-2022

FACULTY OF ENGINEERING AND ARCHITECTURE

# Automated validation of building models against legislation using Linked Data

Emma Nuyts
Student number: 01704320

Supervisor: Prof. ir.-arch. Paulus Present
Counsellor: Ir.-arch. Jeroen Werbrouck

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

Academic year 2021-2022

GHENT UNIVERSITY

# Acknowledgements

Before diving into the world of the Semantic Web and the creation of a more accessible built environment, I want to thank some people.

Firstly, I want to thank my promotors, Paulus Present and Jeroen Werbrouck. Thank you for the discussions on the difference between the industry and the academic world, and the feedback this past year. I also want to thank you both for guiding me over the last couple of years, for teaching me the ins and outs of BIM and JavaScript.

I also want to thank some people on the sideline, for supporting me this last year. Jolien, for all the feminist literature as a break from the programming. Elin, for the daily support and much-needed breaks in the library. Melvin, for the unconditional support. My grandparents, for the eternal support by letter. My dad, for the constant guidance throughout these last years. Pieter, for everything. Without your postivity, this year would not have been the same. I'm already looking forward to our third thesis together.

I really enjoyed writing this thesis, since the alternation of programming and literature on the importance of an accessible (built) environment certainly broadened my understanding of both topics. I hope the few people who will read this, enjoy it just as much.

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

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

# Automated validation of building models against legislation using Linked Data

Emma Nuyts

Supervisor: Prof. ir.-arch. Paulus Present
Counsellor: Ir.-arch. Jeroen Werbrouck

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

Academic year 2021-2022

## Abstract

Manually checking the compliance of a building project with the legislation is a time-consuming and error-prone process. These drawbacks were confirmed by a study by Inter in 2019: out of the 147 reviewed buildings, only 9 were designed following the Belgian legislation on accessibility.

A solution for this complex manual checking procedure is Automated Compliance Checking (ACC). Existing commercial systems use either hard-coded requirements, which are almost as error-prone and time-consuming as manually checking the compliance, or focus on one specific software package.

This research will focus on open Building Information Management (BIM), by using Linked Building Data (LBD), relying on ontologies like Building Topology Ontology (BOT), PROPS, and OPM. To define the requirements from the legislation, the Shapes Constraint Language (SHACL), a Semantic Web technology, is used. This language allows defining constraints using the Resource Description Framework (RDF). One set of rules is both human- and machine-readable since a syntax like the Terse RDF Triple Language (Turtle) can be used. Another perk of this language is that the compliance checking can be combined with a Model View Definition (MVD), ensuring that all the necessary data is present in the building project.

These objectives will be combined in a proof of concept application. A user can upload an LBD graph of a building project, which is then evaluated against the SHACL rulebook. The output is an interface of the validation report, where full transparency for the user is crucial.

**Keywords:** Automated Compliance Checking, Linked Data, SHACL

# Extended abstract

## Current situation

Building Information Management (BIM) has been widely implemented in the Architecture Engineering and Construction (AEC) industry in the last decade but is often not used to its full extent. While this model contains both the geometry and extra information like non-physical objects or semantic information about the building elements, compliance with the building code is still often manually calculated. This procedure involves retrieving the needed data from the model and then separately calculating the compliance. This process is not only time-consuming but also error-prone. That the latter is still a significant problem, is seen in a Belgian study by Inter (2019); out of 147 reviewed building projects, only 9 were designed complying with the building code on accessibility. Furthermore, the checking procedures in the urban planning department appear inadequate due to time pressure or a lack of interest in accessibility.

Fortunately, there is increasingly more research on the topic of automating the compliance checking, by combining a data set of the legislation with a dataset of the building project.

## Research objectives

While existing commercial systems often focus on one specific software package, this research will investigate the potential of open Web standards, applied to rule checking in the built environment. Moreover, it will situate the conversion chain required to move from current BIM authoring tools, used in the industry towards an open system for compliance checking. The research will use Linked Building Data (LBD) since both Industry Foundation Classes (IFC) and ifcOWL have some weaknesses when used in the context of the Semantic Web. This building graph will rely on ontologies like the Building Topology Ontology (BOT), the Ontology for Property Management (OPM), and PROPS (Bonduel, Oraskari, Pauwels, Vergauwen, & Klein, 2018).

Furthermore, the research will evaluate the suitability of sentence structures in governmental documents for transformation into machine-readable rules, with as a use-case the regulation on accessibility. These rules will be defined using the Shapes Constraint Language (SHACL). Ideally, this language also incorporates some form of additional Model View Definition (MVD), to ensure the model is of enough quality, and all the needed information is present in the building graph, thus preventing some form of 'false compliance' due to missing building data (Oraskari, Senthilvel, & Beetz, 2021). Moreover, the language should be able to handle inference statements (if…then…) and should be implementable in a web-based tool, where the rules are easy to access and update. Additionally, the set of rules should be both human- and machine-readable, meaning there is no discrepancy between what is checked, and what the user thinks is being checked. In short, SHACL will be evaluated for use in compliance checking.

These objectives will be combined in a proof of concept application.[i] Here, a user can upload an LBD graph of a building project, which is then evaluated against the defined SHACL constraints. The output is a user interface of the validation report, of which the textual file can be downloaded. In this validation report, the transparency for the user is a key point: the report contains exactly which building element does not comply with which constraint. It also includes the identifier of the non-complying building element, meaning it is easy to find and alter in the native modeling software.

While this specific thesis will focus on accessibility, due to the starting point with the evaluation of Inter, the method should be applicable to more kinds of building code, and ideally all of them.

[i] The code can be found at https://github.com/NuytsE

## Outline

In **chapter 1**, the current situation on compliance checking is discussed, mainly focussing on the inadequate checking procedures at the urban planning department and the physically inaccessible design of buildings by architects. This is followed by the scope of the thesis, its research objectives, and the structure of the chapters.

**Chapter 2** is an introduction to the concept of Automated Compliance Checking and the two data sources it combines. On one hand, the legislative system in Belgium is reviewed, including the timing of compliance checking in a traditional architectural workflow. On the other hand, a brief introduction to BIM and the role of buildingSMART in the standardization of these processes is given.

Since this thesis uses Linked Data, an introduction to this topic is given in **chapter 3**. It provides an explanation of the Resource Description Framework (RDF) and its semantic extensions RDF Schema (RDFS) and the Web Ontology Language (OWL). Furthermore, the SPARQL Protocol and RDF Query Language (SPARQL) and the Shapes Constraint Language (SHACL) are discussed. These languages are respectively used for querying RDF graphs and defining constraints on the data graph. Lastly, the implementation of Linked Data in the AEC industry is presented by discussing Linked Building Data (LBD) and the different approaches for representing geometry.

Some background on the methodology is explained in **chapter 4**, again divided into a section on the normative and the building data. Firstly, some methods for the processing of legislation are reviewed, after which different strategies for the creation of machine-readable rules are discussed. Lastly, the rules will be stored in a way that ensures full transparency to the user. For the building data, the focus lies on the limitation of BIM authoring tools and defining what a sufficient quality of the BIM model is, to be able to use the building data in compliance checking.

In **chapter 5**, the methodology of the research is discussed. The normative knowledge is first structured, after which some additional properties on building elements are defined, to be able to define the constraints for accessibility. A rulebook is created using the Shapes Constraint Language (SHACL). The rulebook first defines a Model View Definition (MVD) to ensure all the necessary building data is present. The different types of constraints are defined afterward, and the method for making a combination of those types is explained, ensuring the different requirements can be defined in the rulebook.

A proof of concept application is proposed in **chapter 6**, which starts with explaining the conversion workflow from IFC to LBD, with an intermediate step using ifcOWL. Furthermore, the interface of the tool is shown by checking the compliance of a student project and the architecture of the application is explained by describing the different steps of the compliance checking procedure and the used modules.

Lastly, **chapter 7** starts with an evaluation of SHACL to define constraints on building data for compliance checking. Furthermore, the proof of concept application is evaluated. Moreover, the developed methodology is evaluated in the entirety of the building legislation, since this thesis focuses on accessibility. Lastly, the scope of this particular research is passed, by discussing possible future work on this topic.

## Conclusion

The developed methodology consisted of three main decisions: using RASE to structure the normative data, using LBD to create the building graph, and using SHACL to define the constraints. While the RASE markup proved to be an effective way to manually structure the legislation, it is quite laborious and the efficiency is highly dependent on sentence structures. Representing the building data in an LBD graph was successful since this graph is easy to query and understand. Some extra properties had to be derived from the ifcOWL graph to be able to check the compliance, but this does only prove the modular aspect of LBD. Furthermore, SHACL constraints can easily be defined on the semantic data in an LBD graph, although further research is needed to evaluate geometrical constraints. Moreover, the combination of an MVD to check if all the necessary data is present in the graph and checking the compliance appears to be interesting in the context of the Semantic Web. In short, the developed methodology can be used to check any type of prescriptive building legislation, as long as the data is defined in the building graph.

## References

Bonduel, M., Oraskari, J., Pauwels, P., Vergauwen, M., & Klein, R. (2018). The IFC to Linked Building Data Converter - Current Status. *LDAC 2018 Linked Data in Architecture and Construction: Proceedings of the 6th Linked Data in Architecture and Construction Workshop*, (pp. 34-43). London, United Kingdom.

Inter. (2019). *Evaluatieonderzoek Vlaamse toegankelijkheidsverordening.*

Oraskari, J., Senthilvel, M., & Beetz, J. (2021). SHACL is for LBD what mvdXML is for IFC. *Proceedings of the Conference CIB W78*, (pp. 693-702). Luxembourg.

# Uitgebreid abstract

## Huidige situatie

*Building Information Management* (BIM) is in de afgelopen 10 jaar steeds meer geïmplementeerd in de *Architecture Engineering and Construction* (AEC) industrie, maar wordt vaak niet ten volle benut. Hoewel dit model zowel geometrie als extra informatie zoals niet-fysieke objecten of semantische informatie bevat over de bouwelementen, wordt de toetsing aan de bouwnormen nog vaak manueel berekend. Deze methode bestaat eruit om eerst de nodige data uit het model te halen, en dan apart de overeenstemming met de wetgeving te berekenen. Dit proces kost niet enkel veel tijd, het is ook foutgevoelig. Dat dit laatste nog steeds een aanzienlijk probleem is, is te zien in een Belgische studie door Inter (2019); van de 147 onderzochte gebouwprojecten, voldeden er slechts 9 aan de toegankelijkheidsverordening. Bovendien blijken ook de controles door de stedenbouwkundige dienst ontoereikend, door een gebrek aan tijd of interesse voor toegankelijkheid.

Gelukkig is er steeds meer onderzoek op het automatiseren van deze controle, door een data set van de wetgeving te combineren met een data set van het gebouwproject.

## Onderzoeksdoelen

Hoewel bestaande commerciële systemen vaak focussen op een specifiek softwarepakket, zal deze thesis het potentieel van open Web standaarden onderzoeken, toegepast op regelcontrole in de gebouwde omgeving. Bovendien zal het de conversieketen die nodig is om van BIM software in de industrie naar een open systeem voor controle van wetgeving te gaan situeren. Het onderzoek zal gebruik maken *van Linked Building Data* (LBD) aangezien zowel *Industry Foundation Classes* (IFC) als ifcOWL enkele nadelen kennen wanneer ze gebruikt worden in de context van het Semantisch Web. De graaf van de gebouwdata zal steunen op ontologieën als de *Building Topology Ontology* (BOT), de *Ontology for Property Management* (OPM) en PROPS (Bonduel, Oraskari, Pauwels, Vergauwen, & Klein, 2018).

Bovendien zal dit onderzoek de bruikbaarheid van zinsstructuren in overheidsdocumenten voor de omzetting naar machine-leesbare regels evalueren, meer specifiek de toegankelijkheidsverordening. De regels zullen gedefinieerd worden met behulp van de *Shapes Constraint Language* (SHACL). Optimaal zou deze taal ook een bijkomende *Model View Definition* (MVD) kunnen definiëren, om te verzekeren dat het model van voldoende kwaliteit is en alle nodige informatie aanwezig is in de gebouwdata, waardoor 'valse conformiteit' door ontbrekende data wordt voorkomen (Oraskari, Senthilvel, & Beetz, 2021). Daarnaast moet de taal ook *inference* verklaringen (als…dan…) aankunnen en zou deze implementeerbaar moeten zijn in een web-gebaseerde applicatie, waar de regels makkelijk toegankelijk en te updaten zijn. Bijkomend zou eenzelfde set van regels zowel voor de mens als voor de machine leesbaar moeten zijn, zodat er geen discrepantie is tussen wat er gecontroleerd wordt en wat de gebruiker denkt dat er gecontroleerd wordt. Kortom zal SHACL geëvalueerd worden om de naleving op bouwnormen te verifiëren.

Deze onderzoeksdoelen zullen gecombineerd worden in een *proof of concept* applicatie.[i] Hierin kan een gebruiker een LBD graaf van een bouwproject opladen, welke dan geëvalueerd wordt tegen de gedefinieerde SHACL voorschriften. De output is een gebruikersinterface van het rapport, waarvan het tekstuele bestand gedownload kan worden. In dit rapport is de transparantie voor de gebruiker cruciaal: het bevat exact welk element niet voldoet aan welk voorschrift.

[i] De code is te vinden op https://github.com/NuytsE

Bovendien bevat het ook het identificatienummer van het niet-conforme bouwelement, zodat het makkelijk op te zoeken en te veranderen is in de modelleersoftware.

Hoewel deze thesis specifiek zal focussen op toegankelijkheid, door het uitgangspunt met de evaluatie van Inter, zou de methode toepasbaar moeten zijn op meer soorten wetgeving, en idealiter op ze allemaal.

## Overzicht

In **hoofdstuk 1** wordt de huidige situatie van de naleving van bouwnormen besproken, waarin de focus vooral ligt op de ontoereikende controles van de stedenbouwkundige dient en de fysiek ontoegankelijke ontwerpen van architecten. Hierna wordt het onderzoeksveld van de thesis afgebakend, en de onderzoeksdoelen en de structuur ervan besproken.

**Hoofdstuk 2** is een introductie tot het concept van *Automated Compliance Checking* (ACC) en de twee data sets die hierin gecombineerd worden. Aan de ene kant wordt het wetgevingsstelsel in België beoordeeld, waaronder ook het tijdstip van de controle in een traditionele architecturale workflow. Aan de andere kant wordt er een korte introductie tot BIM en de rol van buildingSMART in de standaardisatie van deze processen gegeven.

Aangezien deze thesis gebruik maakt van *Linked Data*, wordt er een introductie tot dit onderwerp gegeven in **hoofdstuk 3**. Het verstrekt een toelichting van het *Resource Description Framework* (RDF) en zijn semantische uitbreidingen *RDF Schema* (RDFS) en de *Web Ontology Language* (OWL). Bovendien wordt het *SPARQL Protocol and RDF Query Language* (SPARQL) en de *Shapes Constraint Language* (SHACL) uitgelegd. Deze talen worden respectievelijk gebruikt voor het *queryen* van RDF grafen en voor het opleggen van restricties op de graaf. Ten slotte wordt de implementatie van *Linked Data* in de AEC industrie besproken, door het uitleggen van *Linked Building Data* (LBD) en de verschillende benaderingen om geometrie te beschrijven.

Wat achtergrond over de methodologie wordt gegeven in **hoofdstuk 4**, opnieuw opgedeeld in een deel over normatieve en gebouwdata. Eerst worden enkele methodes voor het verwerken van wetgeving besproken, waarna verschillende strategieën voor het creëren van machine-leesbare regels worden gegeven. Ten slotte moeten deze regels op een manier opgeslagen worden die volledige transparantie ten opzichte van de gebruiker garandeert. Voor de gebouwdata ligt de focus op de limieten van BIM software en het definiëren wat een voldoende kwaliteit van een BIM model is, om de gebouwdata te kunnen gebruiken voor de controle.

In **hoofdstuk 5** wordt de methodologie van het onderzoek besproken. De normatieve data wordt eerst gestructureerd, waarna enkele aanvullende eigenschappen van bouwelementen worden gedefinieerd, om de voorwaarden van de toegankelijkheidsverordening te kunnen definiëren. Er wordt een regelboek gemaakt met behulp van de *Shapes Constraint Language* (SHACL). Dit regelboek bepaalt eerst een *Model View Definition* (MVD) om er zeker van te zijn dat alle nodige gebouwdata aanwezig. De verschillende types van regels worden hierna besproken, en ook de manier om combinaties van deze types te maken, om alle voorschriften te kunnen bepalen.

Een *proof of concept* applicatie wordt voorgesteld in **hoofdstuk 6**, welke start met een verklaring van de conversie van IFC naar LBD, met een tussenstap via ifcOWL. Bovendien wordt de interface van de tool getoond door het controleren van een studentenproject en wordt de architectuur van de applicatie uitgelegd door het beschrijven van de verschillende stappen van de procedure en de gebruikte modules.

Ten slotte start **hoofdstuk 7** met een evaluatie van SHACL om de voorschriften op de gebouwdata te bepalen. Bovendien wordt de *proof of concept* applicatie beoordeeld. Verder wordt de ontwikkelde methodologie geëvalueerd in het geheel van de bouwwetgeving, aangezien deze thesis focuste op toegankelijkheid. Ten slotte wordt aan het onderzoeksveld van deze thesis voorbijgegaan, door mogelijk verder onderzoek over dit onderwerp te bespreken.

## Conclusie

De ontwikkelde methodologie bestond uit drie voorname beslissingen: het gebruik van RASE voor het structureren van de normatieve data, het gebruik van LBD voor de gebouwdata en het gebruik van SHACL om de voorschriften te bepalen. Hoewel de RASE markup een effectieve manier bleek om wetgeving manueel te structureren, is het vrij arbeidsintensief en is de efficiëntie ervan sterk afhankelijk van de zinsopbouw. Het gebruik van LBD om de gebouwdata te representeren was succesvol, aangezien deze graaf makkelijk te *queryen* en te begrijpen is. Enkele extra eigenschappen werden uit ifcOWL afgeleid om alle voorschriften te kunnen controleren, maar dit bewijst enkel het werkend modulair karakter van LBD. Bovendien kunnen de SHACL-restricties makkelijk gedefinieerd worden op de semantische data in een LBD graaf, hoewel verder onderzoek nodig is voor de evaluatie van geometrische eisen. Overigens is de combinatie van een MVD om te controleren of alle nodige data aanwezig is en het controleren van de voorschriften interessant in de context van het Semantisch Web. Kortom kan de ontwikkelde methodologie gebruikt worden om ieder type van prescriptieve wetgeving te controleren, zolang de nodige informatie gedefinieerd is in de gebouwdata.

## Referenties


Bonduel, M., Oraskari, J., Pauwels, P., Vergauwen, M., & Klein, R. (2018). The IFC to Linked Building Data Converter - Current Status. *LDAC 2018 Linked Data in Architecture and Construction: Proceedings of the 6th Linked Data in Architecture and Construction Workshop*, (pp. 34-43). London, United Kingdom.

Inter. (2019). *Evaluatieonderzoek Vlaamse toegankelijkheidsverordening.*

Oraskari, J., Senthilvel, M., & Beetz, J. (2021). SHACL is for LBD what mvdXML is for IFC. *Proceedings of the Conference CIB W78*, (pp. 693-702). Luxembourg.

# Contents

# List of figures

# List of abbreviations

| | |
|---|---|
| ACC | Automated Compliance Checking |
| AEC | Architecture, Engineering, and Construction |
| AI | Artificial Intelligence |
| ASCII | American Standard Code for Information Interchange |
| BBRI | Belgian Building Research Institute |
| BCM | Building Compliance Model |
| BEO | Building Element Ontology |
| BIM | Building Information Management |
| BOT | Building Topology Ontology |
| CAD | Computer-Aided Design |
| CG | Community Group |
| CNL | Controlled Natural Language |
| CSV | Comma-Separated Values |
| CWA | Closed World Assumption |
| DRL | Drools Rule Language |
| ER | Exchange Requirements |
| FOG | File Ontology for Geometry |
| GUID | Globally Unique Identifier |
| HTTP | HyperText Transfer Protocol |
| IDM | Information Delivery Manual |
| IFC | Industry Foundation Classes |
| IRI | Internationalized Resource Identifier |
| JSON | JavaScript Object Notation |
| JSON-LD | JavaScript Object Notation for Linked Data |
| LBD | Linked Building Data |
| MVD | Model View Definition |
| NLP | Natural Language Processing |
| OMG | Ontology for Managing Geometry |
| OPM | Ontology for Property Management |
| OWA | Open World Assumption |
| OWL | Web Ontology Language |
| RASE | Requirement Applicability Selection Exception |
| RDF | Resource Description Framework |
| RDFa | Resource Description Framework in Attributes |
| RDFS | Resource Description Framework Schema |
| RIBA | Royal Institute of British Architects |
| SHACL | Shapes Constraint Language |
| ShEx | Shape Expressions |
| SPARQL | SPARQL Query Language for RDF |
| SWRL | Semantic Web Rule Language |
| Turtle | Terse RDF Triple Language |
| UD | Universal Design |
| UI | User Interface |
| UNA | Unique Name Assumption |
| URI | Universal Resource Identifier |
| URL | Uniform Resource Locators |
| W3C | World Wide Web Consortium |
| XML | eXtensible Markup Language |
| XSD | XML Schema Definition |

# List of prefixes

@prefix beo: < https://pi.pauwel.be/voc/buildingelement# > .
@prefix bot: < https://w3id.org/bot# > .
@prefix inst: < http://example.org/ > .
@prefix omg: < https://w3id.org/omg# > .
@prefix opm: < https://w3id.org/opm# > .
@prefix owl: < http://www.w3.org/2002/07/owl# > .
@prefix props: < https://w3id.org/props# > .
@prefix rdf: < http://www.w3.org/1999/02/22-rdf-syntax-ns# > .
@prefix rdfs: < http://www.w3.org/2000/01/rdf-schema# > .
@prefix schema: < http://schema.org/ > .
@prefix sh: < http://www.w3.org/ns/shacl#> .
@prefix xsd: < http://www.w3.org/2001/XMLSchema# > .

# 1 Introduction

## 1.1 Current situation

Nowadays, both designers and people in the urban planning department spend a lot of time checking if building projects comply with national standards. Automatization of this process will make sure that architects can work more efficiently on their designs but will also decrease the risk of inadequate checking procedures in the urban planning department.

A Belgian study from 2019 shows that the latter is still a point of improvement in Belgium: out of 147 reviewed building permits, only 44 were classified correctly regarding accessibility. However, only 9 of those designs fully satisfied the requirements, meaning only 6% of the inspected public buildings would be accessible in theory. Even more striking is that out of those 147 designs, none actually complied with the building code on accessibility after the execution of the building (Inter, 2019).

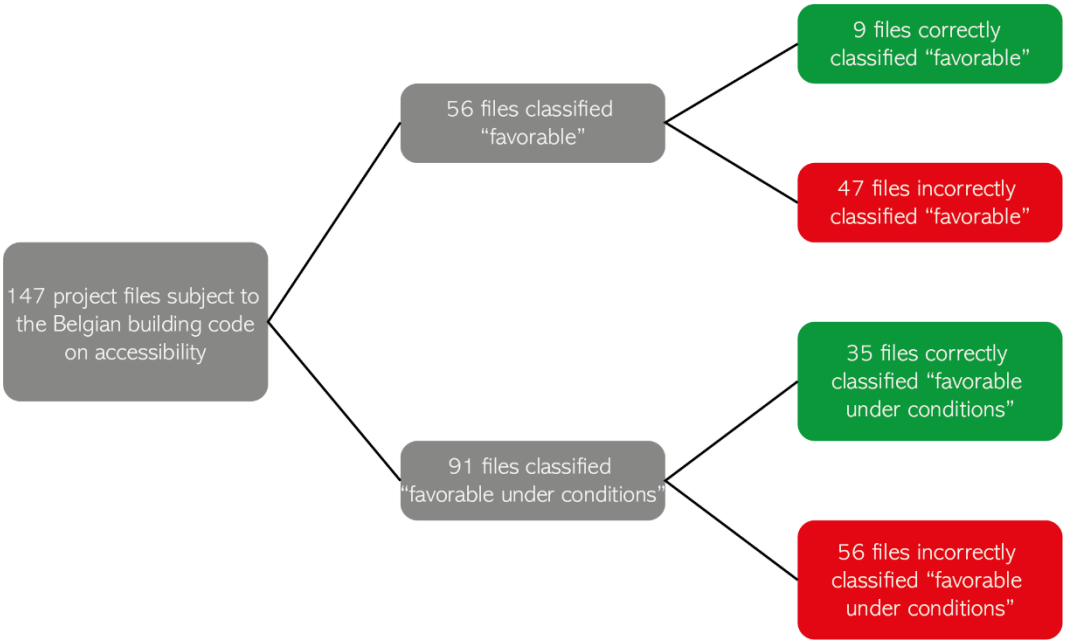


*Figure 1 - Only 44 out of 147 projects were classified correctly, and only 9 were designed complying to the building code on accessibility (Inter, 2019)*

The questioned architects indicate they do know the national standard on accessibility and only 1.02% answer they find it unnecessary to design an accessible project. This does however not relate to the 147 inaccessible building projects. This may be because an inclusive design is often seen as a burden, since hallways and doors need to be wider, and slopes can take up quite some space if not integrated properly. Hence why incorporating immediate feedback into the design process would be an improvement.

These remarkable percentages are also noticed by Belgian politicians, who specify three strategies for handling the problem: educating architects, educating officials in the urban planning department, and incorporating mandatory advice from an independent organization like Inter (Vaes, Votquenne, Baert, & Arnoudt, 2022). However, automating (a part) of the compliance checking process would also lead to fewer mistakes and a better understanding of the regulation.

Creating an accessible built environment is not only crucial for people who make use of a wheelchair permanently. Every one of us will come to a point in our lives where we make use of a wheelchair or rollator, take a buggy with us, or have other mobility issues.

## 1.2 Scope of the thesis

Building Information Management[1] (BIM) has widely been implemented in the Architecture Engineering and Construction (AEC) industry in the last decade but is often not used to its full extent. While this model contains both the geometry and extra information like non-physical objects or semantic information about the building elements, compliance with the building code is still often manually calculated. This procedure involves retrieving the needed information from the model and then separately calculating the compliance. Not only the calculation is error-prone, but mistakes can even be made in the copying of the needed values. This process can be made more efficient, with the aid of Automated Compliance Checking (ACC). While existing commercial systems often focus on one specific software package, this research will investigate the potential of open Web standards, applied to rule checking in the built environment. Moreover, it will situate the conversion chain required to move from current BIM tools, used in the industry towards an open system for compliance checking. The research will use the Linked Building Data (LBD) data model since both Industry Foundation Classes (IFC) and ifcOWL have some weaknesses when used in the context of the Semantic Web. Furthermore, the research will seek a Semantic Web technology that can check compliance. Ideally, this language also incorporates some form of additional Model View Definition (MVD), to ensure the model is of enough quality, and all the needed information is present in the building graph, thus preventing some form of 'false compliance' due to missing building data. Moreover, the language should be able to handle inference statements (if…then…) and should be implementable in a web-based tool, where the rules are easy to access and update. Additionally, the set of rules should be both human- and machine-readable, meaning there is no discrepancy between what is checked, and what the user thinks is being checked.

While this specific thesis will focus on accessibility, due to the starting point with the evaluation of Inter, the method should be applicable to more kinds of building code, and ideally all of them.

Apart from the defined objectives, the four key requirements of a language for automated code checking as defined by Greenwood, Lockley, Malsane & Matthews (2010) can be used to evaluate the suitability:

- The computer programmed rules must be easily understood by the regulation authors;
- The lifecycle of the rule base must be independent of software and schema updates;
- All development must comply with Open Standards;
- Consideration must be given to the industry processes of model authoring.

## 1.3 Structure of the thesis

**Chapter 2** is an introduction to the concept of ACC and the two data sources it combines. On one hand, the legislative system in Belgium is reviewed, including the timing of compliance checking in a traditional architectural workflow. On the other hand, a brief introduction to BIM is given, and the role of buildingSMART in the standardization of these processes.

Since this thesis uses Linked Data, an introduction to this topic is given in **chapter 3**. Lastly, the implementation of Linked Data in the AEC industry is presented.

Some background on the methodology is explained in **chapter 4**, again divided into a section on the normative and the building data. Firstly, some methods for the processing of legislation are reviewed, after which different strategies for the creation of machine-readable rules are discussed. Lastly, the rules will be stored in a way that ensures full transparency to the user. For the building data, the focus lies on the limitation of BIM authoring tools and defining what a sufficient quality of the BIM model is.

[1] 'BIM' is an ambiguous term: some use it as Building Information Model (a product), while others define it as Building Information Management (a process). In this thesis, the latter definition will be used, since the process of handling information is more relevant in this context.

In **chapter 5**, the methodology of the research is discussed. The normative knowledge is first structured, after which a rulebook using the Shapes Constraint Language (SHACL) is created, by defining different types of constraints and explaining how a combination of those constraints can be made.

A proof of concept application is proposed in **chapter 6**, starting with the conversion workflow from IFC to LBD. The architecture of the web-based tool is explained, along with a demonstration using a student project.

Lastly, **chapter 7** starts with an evaluation of SHACL to define constraints on building data for compliance checking. Furthermore, the implementation and its positioning in the entirety of the building legislation are discussed. Lastly, the scope of this particular thesis is passed, by discussing possible future work on this topic.

# 2 Automated Compliance Checking

## 2.1 Definition

The AEC industry has gradually evolved from using 2D Computer-Aided Design (CAD) visualization techniques to implementing BIM models. Even though this kind of model centrally stores all the available building information, the checking of compliance against the national building code is still mostly done manually by both architects and people of the urban planning department. This is an error-prone and time-consuming process, which can be automated using ACC. To do this, a data set of the normative data originating from the building legislation and the building data have to be linked.

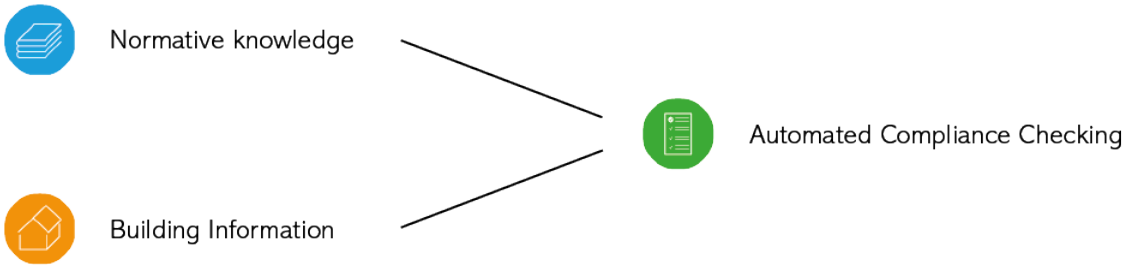


*Figure 2 - Concept of Automated Compliance Checking*

## 2.2 Normative knowledge

Building legislation consists of many different parts, to ensure a safe and coherent built environment. There are written regulations for energy efficiency, fire safety, accessibility… But all of them have one thing in common: the legislation is written in a human-readable format, and seldom in a machine-readable way.

However, even in the legislation written for humans, there should be a differentiation between *readable* and *understandable*. The '*Evaluatieonderzoek Vlaamse Toegankelijkheidsverordening*' (Inter, 2019) investigated the ease of understanding and implementation of the Belgian legislation on accessibility, with a survey for architects and people in the urban planning department. This yielded some remarkable results: 60% of the urban planners declare that their knowledge about the legislation is mediocre to bad, and only 47% deem the norms on accessibility clear. Moreover, only 32% of the questioned architects think it is (very) simple to design an accessible building.

This also leads to inadequate checking procedures in the urban planning department: 28% admits only looking at specific files. And out of the files that were checked, 24% indicate they only check if a note on accessibility was added (so not checking the contents) or skims over it rather quickly.

The difficulty to apply the legislation is not due to complex juridical jargon since the building legislation is specifically written for architects with minimal legal knowledge. This is however a problem in for example tax law, as Curtotti & McCreath (2013) describes. The problem with the building legislation on accessibility stems from complex descriptions, which are never accompanied by a figure. An excellent example can be found in (the translated) article 24 of the regulation on accessibility:

> *For manually operated doors, the outer edge of the free and level turning area on the pull side of the door must touch the door's turning plane, and the outer edge of the free and level turning area on the push side of the door must touch the closed-door plane.*

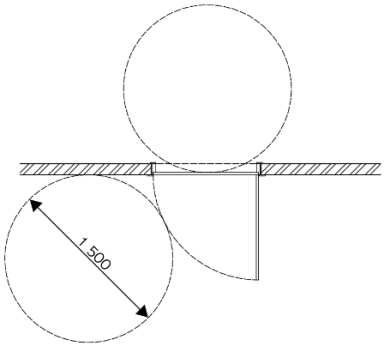


*Figure 3 - Positioning of turning circles in relation to a door, following article 24*

A simple figure would immediately improve the understanding of this complex sentence.

There are ongoing projects to rewrite the Belgian legislation on accessibility, focussing on three key principles: integral accessibility, the chain of accessibility, and Universal Design (UD). Integral accessibility implies that the legislation should focus on more than just physical accessibility, a building should be easy to navigate for all visitors. The chain of accessibility means that every building element should comply since this chain is as strong as its weakest link. Lastly, Universal Design evaluates the specific needs of every user, meaning no additional aids are needed for specific audiences. Moreover, accessibility goes further than the built environment; also, websites, apps, and documents should be accessible, by including psychosocial, informative, and communicative aspects (Vande Reyde, et al., 2020).

This concept of integral accessibility is not only important for the psychological well-being of users but also for their safety since there is a discrepancy between the legislation on fire safety and accessibility in Belgium. The minimal door width for evacuation is only 80 cm (Royal Decree of December 7, 2016), while the door width should be at least 90 cm to ensure accessibility. Creating a non-integral accessible building could thus mean that the designer only creates one entrance of 90 cm, but dozens of scenarios could be imagined where that door cannot be used in case of an emergency.

Apart from some pre-checks, the building only gets thoroughly checked at the moment of the building permit, a tool for compliance checking would thus mean fewer changes after most of the design is already completed. The ability to check the compliance earlier on in the design process would mean a smaller cost of the design changes, while the effect of the changes on the design is minimal, as shown in Figure 4. This concept is easily illustrated with the proper integration of a ramp, which is quite easy when drawn early on in the design process, but always takes up too much space in an almost-completed design.

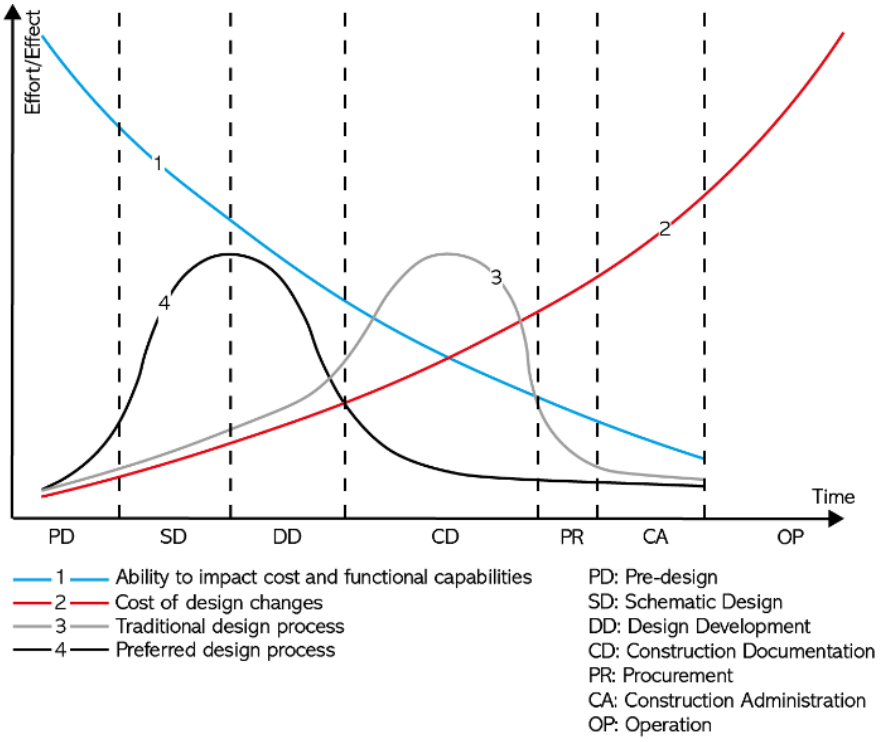


*Figure 4 - The MacLeamy curve describes the opposite course of the ability to make changes and their respective cost (Pauwels & Petrova, 2020)*

Furthermore, the building legislation often gets updated, due to new technological innovations or research. For example, the mentioned Royal Decree on fire safety is revised every 3,5 years on average. This means that it is difficult for architects to keep up with every update of every building code.

## 2.3 Building Information Management

The use of BIM models has drastically increased over the last decade since it not only provides a 3D visualization of the architects' vision but also includes all the building information. This approach ensures a better collaboration through the coordination of information, an accurate simulation with accurate information, and less reworking, remodeling, and manual data input.

buildingSMART differentiates between two kinds of BIM: open BIM and closed BIM. The latter means that native file formats are exchanged, meaning the visibility of the building information is dependent on the chosen software license of one of the stakeholders. When implementing open BIM, all the stakeholders can view the building information, since open and neutral file formats like Industry Foundation Classes (IFC) are used in the information exchange. buildingSMART thus defines open BIM as *'a universal approach to the collaborative design, realization, and operation of buildings based on open standards and workflows'*.[2]

Apart from the IFC standard for data, buildingSMART also defines standards for processes, for which the Information Delivery Manual (IDM) is used. It defines what information is required during collaboration and when it needs to be exchanged, by defining Exchange Requirements (ER). To ensure these can be fulfilled, the software requirements are specified in the Model View Definition (MVD) (Pauwels & Petrova, 2020).

The evolution of the implementation of BIM is shown in Figure 5, defining four levels numbered from zero to three. Level 0 is seen as the mere beginning of BIM. Level 1 is already a big step further in terms of digitalization; 3D software is used for visualizations and as a central storage point for information regarding performance and materials. However, this level indicates a form of 'Lonely BIM', meaning the model is not used collaboratively between stakeholders. Level 2 is often targeted, to ensure an efficient and thorough information exchange between team members. Level 3 (the last level) is described by RIBA (Royal Institute of British Architects) as 'the holy grail' since it uses a purely web-based and data-based approach (2012).

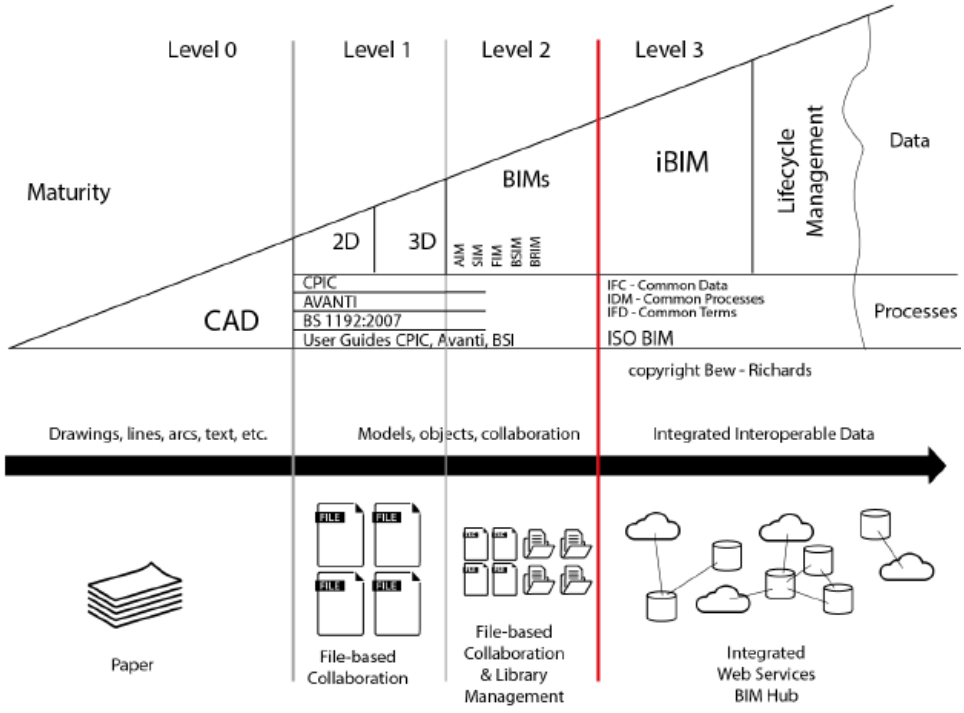


*Figure 5 - The Bew-Richards maturity model defines four levels (Pauwels & Petrova, 2020)*

[2] https://www.buildingsmart.org/about/openbim (last accessed 05/04/2022)

## 2.4 Existing commercial projects

There is ongoing research on Automated Compliance Checking, but publicly available tools mostly have two downsides: they either use some form of hard-coded requirements or they use native file formats, thus enabling closed BIM (Dimyadi & Amor, 2013).

The use of hard-coded requirements is not desired since it takes a long time to manually code all the different constraints. Experience in New Zealand has shown it takes an expert one full day of coding to translate only one page of written building code, making this process time-consuming and not cost-efficient (Amor & Dimyadi, 2021). Moreover, when the building legislation is updated, all those requirements have to get updated manually. This process is almost as error-prone and time-consuming for the developer, as it is to manually check the building code by the designer.

Secondly, the use of closed BIM means only people with the targeted software package can make use of the tool. This method thus excludes designers based on their used software, also meaning that the developers put a lot of time and effort into a tool that cannot be used globally.

Some examples of existing ACC tools that use either hard-coded requirements or focus on one specific software package are Autodesk model checker, Xinaps, Upcodes AI, and SMART review (Beach, Hippolyte, & Rezgui, 2020).

## 2.5 Conclusion

The aforementioned points confirm the need for a methodology for ACC, with the aim to give architects feedback on their design in each phase of the building project. The methodology will focus on using open BIM for the building information, meaning the proposed method can be used regardless of the software package used by the designers. For the building legislation, some form of data structuring and storage will be used. This way, the applied constraints can be easily updated to the most recent versions of the building code.

# 3 Linked Data

## 3.1 Introduction to Linked Data

To be able to provide a vendor-neutral method for compliance checking, using a normative dataset, this thesis will make use of Linked Data. This chapter will provide a brief introduction to key concepts like RDF (3.1.2), RDFS and OWL (3.1.3), SPARQL (3.1.4), and SHACL (3.1.5).

This chapter will however not provide a complete overview of all concepts of Linked Data and the Semantic Web. It will merely discuss and explain the concepts needed for application in compliance checking. The main information comes from The Web of Data (Hogan, 2020) and the website of the World Wide Web Consortium (W3C). When relevant, the web pages are included in a footnote.

### 3.1.1 Definition

In 1990, Tim Berners-Lee intended the use of the World Wide Web (from now on: the Web) as one central data storage point for use in CERN. But the Web evolved from a static and informative Web 1.0 to an interactive experience, denoted as Web 2.0. While this version of the Web mainly focussed on human user experience, a new development has taken place with Web 3.0. This most recent version entails an integrated web experience where machines can understand data in a similar way as humans (Rudman & Bruwer, 2016). The scope of this web is to enable computers to efficiently do more useful work and to develop systems that can support trusted interactions over the network.[3]

This Web 3.0 is often called the Semantic Web, of which the concept of Linked Data lies at the base. The W3C defines Linked Data as *'the collection of interrelated datasets on the Web'*, which is used for large-scale integration of, and reasoning on, data on the Web.[4] This is achieved by using Uniform Resource Identifiers (URI) to name data. By using Hypertext Transfer Protocol (HTTP) URIs, people can then look up this data, and useful information about the data is provided using the W3C standards.[5] A crucial concept is that the data includes links to other URIs so that everything on the Web is linked, as described by Tim Berner-Lee in the five-star open data scheme.


1★ make your stuff available on the Web (whatever format) under an open license

2★ make it available as structured data (e.g. Excel instead of an image scan of a table)

3★ make it available in a non-proprietary open format (e.g. CSV instead of Excel)

4★ use URIs to denote things, so that people can point at your stuff

5★ link your data to other data to provide context

*Figure 6 - Five-star open data scheme (https://5stardata.info/en (last accessed 05/04/2022))*

[3] https://www.w3.org/standards/semanticweb/ (last accessed 05/04/2022)
[4] https://www.w3.org/standards/semanticweb/data (last accessed 05/04/2022)
[5] https://www.w3.org/DesignIssues/LinkedData.html (last accessed 05/04/2022)

### 3.1.2 Resource Description Framework

#### 3.1.2.1 Definition

To achieve full interoperability, the machine-readable description is standardized in the Resource Description Framework (RDF). The implementation of one standard data format has the benefit of not having to convert heterogeneous file formats, like CSV, XML, or JSON. This graph-structured data model was designed with the Semantic Web in mind, making it the foundation of the web.[6]

#### 3.1.2.2 RDF terms

RDF terms are used to refer to the resources that RDFs describe, existing out of Internationalized Resource Identifiers (IRIs), literals, and blank nodes.

IRIs are used to identify generic resources, whereas Uniform Resource Locators (URLs) can only be used to identify and denote the location of documents. RDF terms were first based on URIs, but since these only allow a subset of American Standard Code for Information Interchange (ASCII) characters, IRIs were implemented.

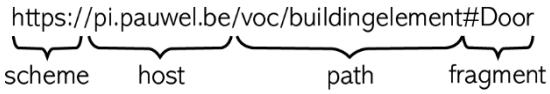


*Figure 7 - An Internationalized Resource Identifier (IRI) consists of a scheme, host, path, and fragment*

It is important to note that RDF allows multiple IRIs to refer to the same source; it does not enforce a Unique Name Assumption (UNA). However, if RDF only consisted of IRIs, it would not contain human-readable information like titles, descriptions, dates, etc. For this reason, literals were implemented.

Since identifying a resource with globally unique IRIs or literals can be restrictive, unknown values can be represented by blank nodes.

#### 3.1.2.3 Triples

While RDF terms identify resources, RDF triples are used to describe resources; to make statements about them. Triples are constructed in the same way as one of the simplest forms of sentence structure in the natural language: subject-verb-object sentences. Moreover, the verb can be generalized as a predicate. A triple is graphically represented with an arrow, from the subject to the object and is considered as the core unit of RDF.

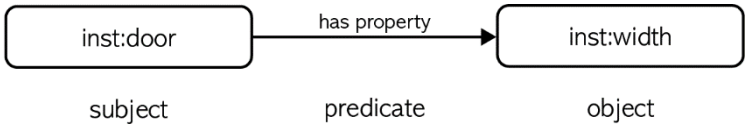


*Figure 8 - Triple defining that a 'door' has a 'width' property*

Each position in the triple is occupied with a single RDF term, where IRIs can appear in any position, literals can only appear as objects, and blank nodes can only appear in the subject or object position.[7]

[6] https://www.w3.org/TR/rdf11-concepts/ (last accessed 20/03/2022)

[7] Since the Semantic Web focusses on internationally available data, using language tags is recommended. However, for this thesis, they will be omitted since no other languages than English are used. After every literal, there thus should be an '@en' language tag for completeness.

### 3.1.2.4 RDF syntaxes

To be able to write down RDF in files for storage, parsing, and processing purposes, RDF graph syntaxes are needed. Some examples of those syntaxes include RDF/XML, N-triples, Turtle, RDFa, and JSON-LD.

While The Terse RDF Triple Language (Turtle) is not the quickest to parse out of these syntaxes, it is the most human-friendly and concise. It allows users to write down RDF graphs in a compact and natural text form, with some abbreviations for common usage patterns.

Some examples of this more concise form of writing include that the IRI `<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>` is denoted with an '`a`' in the predicate position, that predicate lists can be defined by a separation with a semicolon, and that object lists can be defined by a separation with a comma.[8]

Another perk of this syntax is the introduction of prefixes, to be able to shorten a reoccurring part of an IRI. The used prefixes in this thesis can be found in the List of prefixes, and will not be repeated in the figures.

```
inst:wall_1
   a                    beo:Wall , bot:Element ;
   bot:hostsElement     inst:door_2, inst:window_1 .

inst:door_2
   a                    beo:Door , bot:Element ;
   props:width          "0.90"^^xsd:double .
   props:height         "2.10"^^xsd:double .

inst:window_1
   a                    beo:Window , bot:Element ;
   props:width          "1.10"^^xsd:double .
   props:height         "0.90"^^xsd:double .
```

*Figure 9 - Turtle syntax defining a wall, a window, and a door*

### 3.1.2.5 RDF Graphs

The core unit in RDF is a triple, where a set of triples is then defined as an RDF graph. An RDF graph is thus by definition unordered and unindexed. The main perk is that the data is now displayed graph-structured, meaning it is a generic and flexible data model. New edges can thus be defined from any node to any other node with any edge.

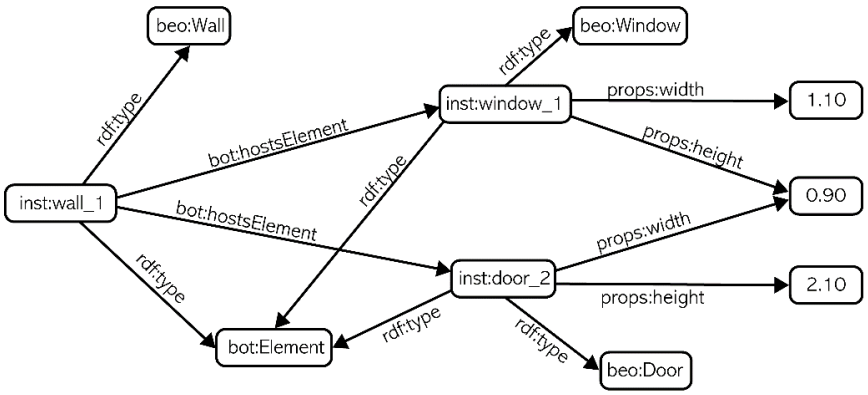


*Figure 10 - RDF graph defining a wall, a window, and a door*

[8] https://www.w3.org/TR/turtle/ (last accessed 20/03/2022)

Properties (denoted with lowercase) are relationships that hold between pairs of resources, often used in the predicate position. Classes (denoted with uppercase) on the other hand, are groups of resources with some conceptual similarities; they group resources of the same type.

RDF makes it possible to merge multiple datasets in one, which is difficult to achieve with for example a CSV file. The ability to merge is important for this context since therefore RDF is considered to be the foundation of the Semantic Web.

### 3.1.3 RDF Schema and Web Ontology Language

Some semantic extensions for RDF were proposed, to define relations between terms in the vocabulary or to define patterns that capture common-sense knowledge.

Two main standards for defining semantics over RDF data are the RDF Schema (RDFS) and the Web Ontology Language (OWL).

#### 3.1.3.1 RDF Schema

RDFS is a lightweight semantic extension of the RDF vocabulary, providing ways to describe groups of related resources and the relationships between them.

The RDFS vocabulary defines dozens of relationships between resources, but the three key instances for defining relationships between classes are `rdfs:subClassOf`, `rdfs:domain`, and `rdfs:range`.

For defining a relation between different classes, `rdfs:subClassOf` is used. For example, the Building Element Ontology (BEO) defines a 'Quarter Turn Stair' as a subclass of the class 'Stair'.[9] These subclasses allow a further specification of the superclass, making an instance of a subclass always an instance of the corresponding superclass (but not the other way around).

To state that the values of a property are instances of a class, `rdfs:range` is used. While `rdfs:domain` is used to state that a resource that has a property, is an instance of a class.[10]

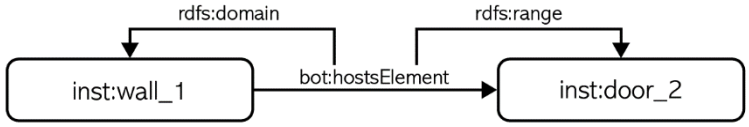


*Figure 11 - Use of rdfs:domain and rdfs:range*

#### 3.1.3.2 Web Ontology Language

While RDFS suffices for simple deductions, more complex definitions are supported by OWL. It aims to extend the core of the RDFS vocabulary with a wide range of new and well-defined terms, allowing to define more in a machine-readable way.[11] OWL can be used to define cardinality or type restrictions, and class expressions. RDF graphs constructed with OWL concepts are called OWL ontologies (Pauwels, Zhang, & Lee, 2017).

[9] https://pi.pauwel.be/voc/buildingelement/index-en.html (last accessed 07/04/2022)
[10] https://www.w3.org/TR/rdf-schema/ (last accessed 05/04/2022)
[11] https://www.w3.org/TR/owl-ref/ (last accessed 07/04/2022)

### 3.1.3.3 TBOX and ABOX

When talking about ontologies, a difference is made between the concepts and the instances. The concept of the ontology layer is called TBOX (Terminological), while ABOX (Assertion) is used to define instances of TBOX concepts. As shown in Figure 12, all the ABOX instances in this thesis will be denoted with an underscore and a (random) instance number after the name of the instance.

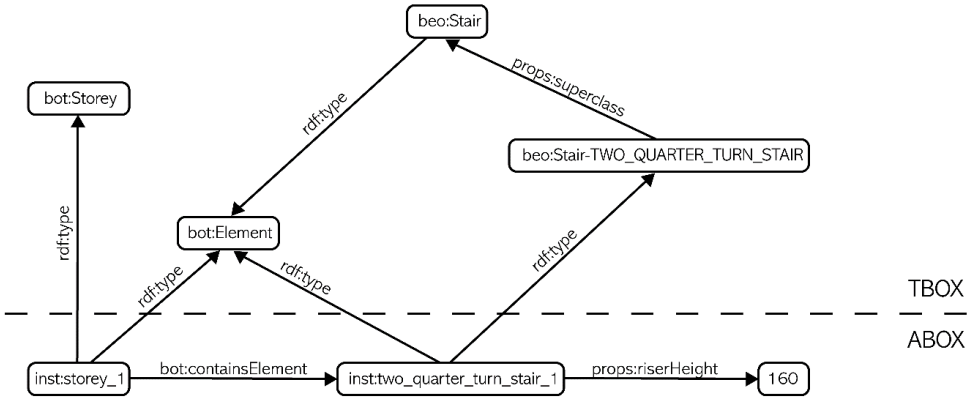


*Figure 12 - ABOX instances are defined using TBOX concepts*

## 3.1.4 SPARQL Protocol and RDF Query Language

Since machines struggle to interpret complex questions posed in the natural language, queries allow users to express questions in a machine-readable format. W3C recommends the SPARQL Protocol and RDF Query Language (SPARQL) in the context of the Semantic Web.

SPARQL allows defining the subject, predicate, or object of a triple as a variable. SPARQL also supports the use of prefixes, meaning the same notation as in the triples can be used. The result of a query is a solution sequence, meaning that there may be zero, one, or multiple results.

SPARQL supports four query forms: SELECT, CONSTRUCT, ASK and DESCRIBE. A SELECT query returns the variables and their bindings directly, while a CONSTRUCT query returns an RDF graph of the variables. An ASK query is used to test whether a query pattern has a solution or not, and a DESCRIBE query will result in an RDF graph containing RDF data about the resources. SPARQL also allows the use of a FILTER in a query pattern, to indicate a constraint on the solutions.[12]

In Figure 13, an example of a SELECT query is shown. Due to the structure of the queried graph, which will be explained in section 3.2.2, there are additional variables (like `?doorwidth` and `?doorheight`) needed. However, these are not returned in the query result shown in Figure 14, since they are not among the selected variables.

```
SELECT ?door ?overallWidthIfcDoor ?overallHeightIfcDoor
WHERE {
?door         rdf:type                      beo:Door .
?door         props:overallWidthIfcDoor     ?doorwidth .
?doorwidth    schema:value                  ?overallWidthIfcDoor .
?door         props:overallHeightIfcDoor    ?doorheight .
?doorheight   schema:value                  ?overallHeightIfcDoor .
}
```

*Figure 13 - SPARQL query to retrieve all doors with their respective width and height*

[12] https://www.w3.org/TR/sparql11-query/ (last accessed 21/03/2022)

The example query results in a table of the requested variables and their bindings since a SELECT query form was used to retrieve all doors with their corresponding width and height property.

| door | overallWidthIfcDoor | overallHeightIfcDoor |
|---|---|---|
| inst:door_76b | 0.813 | 2.420 |
| inst:door_b4f | 0.762 | 2.032 |
| inst:door_6b8 | 1.250 | 2.010 |
| inst:door_761 | 0.813 | 2.420 |

*Figure 14 - Excerpt of the result of the SPARQL query*

Moreover, SPARQL cannot only retrieve defined data in the graph but can also update it. This can be done by using an INSERT, DELETE, LOAD, or CLEAR query.[13]

### 3.1.5 Shapes Constraints Language

While the Semantic Web first started as a place where every resource could be linked to an optional number of other resources, known as an Open World Assumption (OWA), the need for data validation quickly arose. Especially when using Semantic Web standards in rule checking since a graph can never be considered complete. Since RDFS and OWL cannot enforce 'needed' elements of a graph, the Shapes Constraints Language (SHACL) was recommended by the W3C as of 2017. It is one of the few Semantic Web technologies that actually involves a Closed World Assumption (CWA) (Knublauch, 2017).

SHACL is a language for validating RDF graphs against a set of conditions. SHACL has two dialects: SHACL-Core and SHACL-SPARQL. The Core provides a vocabulary for defining constraints in a high-level declarative manner, while SHACL-SPARQL extends this first dialect to allow the use of queries to specify additional constraints.

The SHACL standard distinguishes two classes of an RDF graph: the data graph (the RDF graph to be validated) and the shapes graph (the RDF graph describing the validation constraints). Two types of shapes can be defined in a shapes graph: a `sh:NodeShape` and a `sh:PropertyShape`. The NodeShape targets nodes in the data graph and defines constraints directly on the node at its targets, while a PropertyShape defines constraints on each targeted node, reachable from it via a specified path.

SHACL allows declaring three types of severities when creating the shapes graph: `sh:Info`, `sh:Warning`, and `sh:Violation`. For every shape, `sh:Violation` is the default if `sh:severity` is unspecified. Defining the severities ensures a nuanced and fully human-readable validation report as the result of the validation process.[14]

```
schema:DoorShape
   // apply the shape to a focus node
   a                          sh:NodeShape ;
   // target all nodes with class 'Door'
   sh:targetClass             beo:Door ;
   // target a property of each 'Door'
   sh:property [
   // target the height predicate
    sh:path                   props:overallHeightIfcDoor ;
   // each 'Door' should have at least one height property
    sh:minCount               1 ;
```

[13] https://www.w3.org/TR/sparql11-update/ (last accessed 10/05/2022)
[14] https://www.w3.org/TR/shacl/ (last accessed 08/05/2022)

```
// each 'Door' should have at most one height property
 sh:maxCount             1 ;
// if not, return the violation message
 sh:message              "Each door should have exactly one height property." ;
] .
```

*Figure 15 - Shapes graph defining that all doors should have exactly one value as a height*

The SHACL-SPARQL language, an extension of SHACL-Core, allows expressing restrictions, based on a SELECT query. The language also allows defining constraint components, to declare high-level reusable components.

SHACL also supports some Advanced Features to extend both SHACL-Core and SHACL-SPARQL. They allow defining custom targets, annotation properties, SHACL functions, node expressions, expression constraints, and SHACL rules.[15]

Another option for data validation besides SHACL is the Shape Expressions (ShEx) language. However, for the purpose of compliance checking, SHACL is the more fitting option. Since SHACL allows Scopes, Filters, and constraints, offers mechanisms for inference and is defined as an RDF vocabulary, of which the importance will be explained in section 4.2.2 (Labra Gayo, Prud'hommeaux, Solbrig, & Boneva, 2016). However, ShEx supports recursion, meaning it is possible to define and validate cyclic data structures. There is ongoing research on this subject, creating 'core constraint components' on which recursive constraints can be applied (Corman, Reutter, & Savkovic, 2018).

## 3.2 Linked Data in the AEC industry

### 3.2.1 Linked Building Data

The brief introduction to BIM in section 2.3 of this thesis, concluded with the open and neutral file-format IFC as the 'Holy Grail' of file formats. However, this filetype has some downsides when used in the context of the Semantic Web: (1) the used EXPRESS and XSD languages lack methods for defining formal semantics, making it difficult for reasoning and querying purposes, (2) it is complex to link the building data stored in an IFC file to related data on the web and (3) the IFC schema is large and rather complex, making it a challenge to implement it correctly (Bonduel, Oraskari, Pauwels, Vergauwen, & Klein, 2018).

For this reason, some new methods were developed with the Semantic Web in mind, like ifcOWL and Linked Building Data (LBD). ifcOWL was developed as an ontology lying close to the original IFC schema but made available in OWL. However, since this ontology lies so close to the original schema and contains so much information, it has some negative consequences: many of the EXPRESS-specific semantic constructs are maintained and result in complex and counterintuitive constructs in OWL and RDF, and the ABOX graphs are at least as complex and large as the original IFC model (Pauwels & Roxin, 2016). Since the AEC industry needed a simplified method for handling IFC models, LBD was introduced. For this, the IFC file is first converted to ifcOWL, after which the ifcOWL classes are mapped to the related Building Topology Ontology (BOT) classes. By taking this intermediate conversion step, it becomes possible for the convertor to query the RDF patterns directly. As a case study, the duplex house (distributed by buildingSMART) was converted to both ifcOWL and LBD, to quantify the difference between the two graphs. The comparison reveals that the LBD graph contains 83% fewer triples than the ifcOWL ABOX graph, confirming that the latter is more difficult to query and less human-readable since it contains so much information and provides less overview. Moreover, IFC attributes are defined by specific property paths in LBD, making it easier to query and validate (Bonduel, et al., 2018).

[15] https://www.w3.org/TR/shacl-af/ (last accessed 20/04/2022)

The LBD Community Group (CG) of the W3C developed the lightweight and extensible BOT ontology, providing a high-level description of buildings including stories and spaces, building elements, and a 3D mesh geometry of these spaces and elements. The scope of BOT is to explicitly define necessary relationships between the sub-components of a building (Rasmussen, Lefrançois, Schneider, & Pauwels, 2020).

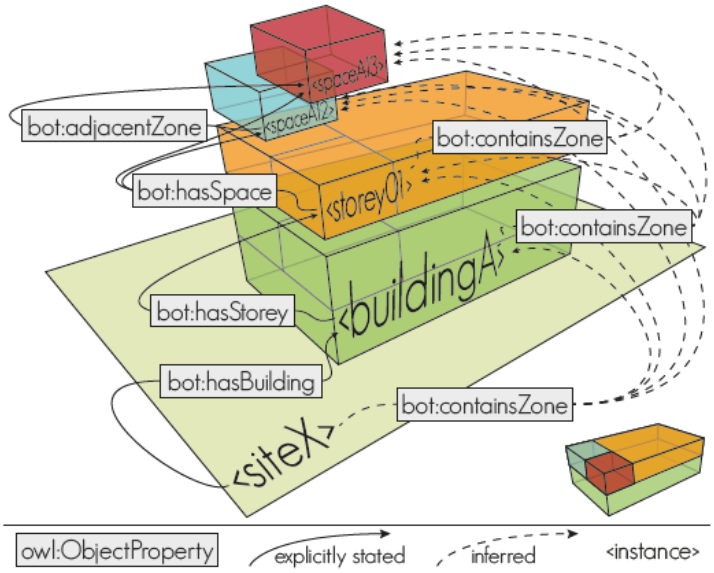


*Figure 16 - Definition of instances and their relations in BOT (Rasmussen, et al., 2020)*

### 3.2.2 Levels of modeling complexity

The LBD CG proposes three levels of modeling complexity for properties, which are used in ontologies such as PROPS and OPM (Rasmussen, Lefrançois, Bonduel, Hviid, & Karlshoj, 2018). The first is used for assigning properties to building elements, while the second is used for the management of these properties.

An overview of these complexities is given in Figure 17. The first level uses one triple to link the building element and the property value, while the second level defines a property, and then assigns the value in a separate triple. The third level uses a property and an evaluation, after which the property value is assigned (Bonduel, et al., 2018). The third level is often used for assigning metadata, or the change of properties during the design process. Since this is less important for this implementation, this thesis will make use of Level 2.

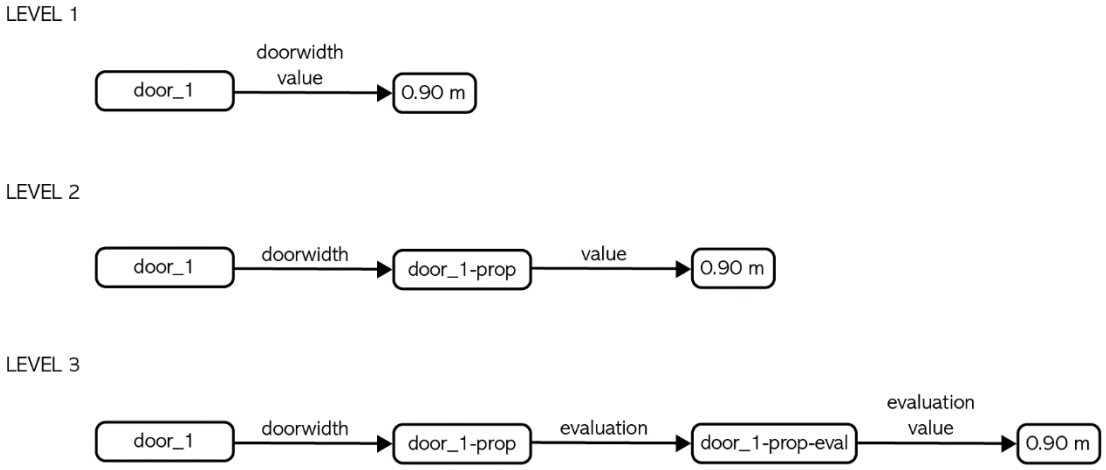


*Figure 17 - Three levels of modeling complexity*

### 3.2.3 Representation of geometry

The exchange of modular building data instead of a file-based exchange would allow a more flexible workflow. For this, geometric and non-geometric data sets would have to be linked. Wagner, Bonduel, Pauwels & Rüppel (2020) define four approaches for representing building geometry in a Semantic Web context. The approaches differentiate in three parts of the geometry description: how the geometry is referred to (linking), how the geometry is stored (storage), and whether the content is structured using Semantic Web specifications or not (content).

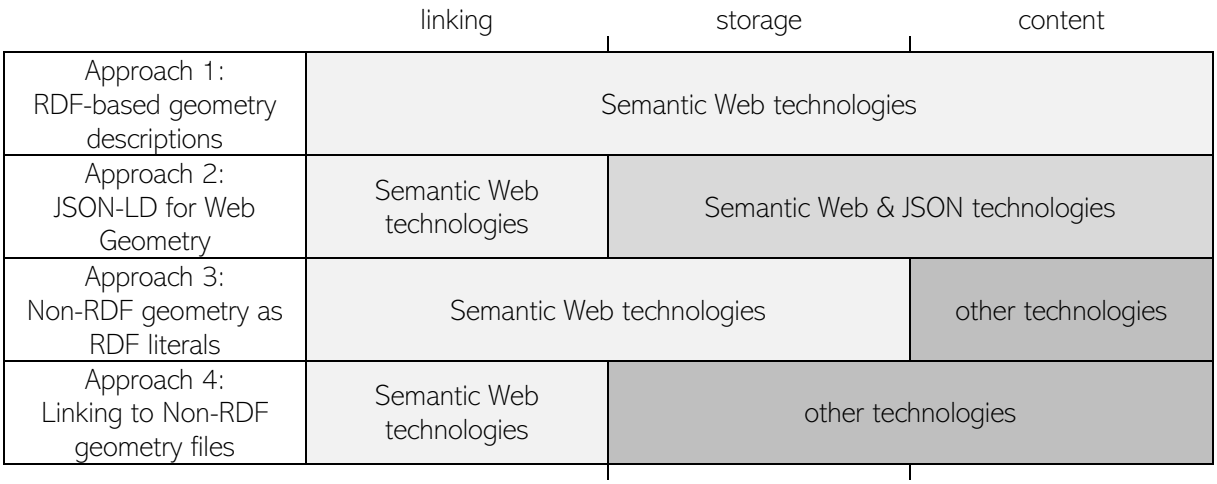


*Figure 18 - Overview of approaches for representing building geometry (Wagner, et al., 2020)*

The first approach is implemented the closest to the Semantic Web, by using its specifications for all three parts and is the most relevant for this thesis. The geometry scheme has to be defined in RDF, for example using OntoBREP, OntoSTEP, or GEOM. The link to the geometry description can be realized with the Ontology for Managing Geometry (OMG). This ontology provides methods for linking geometric and non-geometric descriptions. Furthermore, it provides properties to relate RDF and non-RDF geometry descriptions to a node. To define sub-properties of the generic OMG properties, the Ontology for Geometry File (FOG) can be used.

The second approach also uses Semantic Web standards for all three parts, but the last two are more restricted since the storage and structure are both compliant with both RDF and JSON. Approach three uses a Semantic Web approach for linking and storing the geometry but uses a different technology for the content structure. The fourth approach only uses RDF to link the geometry descriptions and depends on other technologies for the structure and storage of the content. This approach thus has the least accordance with the Semantic Web.

## 3.3 Conclusion

Linked Data is used for integration of, and reasoning on the Web, where machine-readable descriptions are made using RDF. The core unit of RDF are triples, which can consist of IRIs, literals, and blank nodes. Multiple triples can be combined, leading to an RDF graph. However, to be able to write RDFs down in files for storage, RDF graph syntaxes are needed. Out of those syntaxes, Turtle is the most human-friendly and concise. Further semantic extensions of RDF are RDFS and OWL, which can define relations between terms in the vocabulary or define patterns that capture common-sense knowledge. In these ontologies, there is a differentiation between concepts (TBOX) and instances of those concepts (ABOX). In the context of the Semantic Web, SPARQL is used for expressing machine-readable questions and SHACL is used to define constraints on the data graph, resulting in a validation report. The chapter ended with a discussion on the use of Linked Data in the AEC industry, elaborating on the differences between ifcOWL and LBD, the different levels of modeling complexity, and approaches to represent geometry in building graphs.

# 4 Preliminary research

## 4.1 Introduction

The compliance checking will be done by combining the legislative and the building data. The normative data first has to be processed, after which it can be transformed into machine-readable rules. For the building data, some limitations of BIM authoring tools will be mentioned, after which different methods for obtaining a qualitative model will be described.

For all these steps, different methods exist. This chapter will give an overview of these strategies, after which a combination will be chosen and implemented.

## 4.2 Normative knowledge

### 4.2.1 Processing the normative data

The normative data first has to be processed, to be able to use it for compliance checking. For this, a few methods exist: like using Natural Language Processing (NLP), using a manual mark-up language, or proposing a new methodology or framework.

Natural Language Processing (NLP) is the science of making computers understand the human language. This technique facilitates text analysis and processing, which can be used for extracting regulatory data from the building legislation. However, the understanding of both simple and complex sentences is essential to achieve full automation in the analysis of the normative knowledge (Zhang & El-Gohary, 2017). While this process may seem the most efficient (after the Artificial Intelligence (AI) is developed), some form of control mechanism will still be needed when using NLP based on AI with a learning component. Since the results are not deterministic, the translations can change with time and increased learning of the AI (Amor & Dimyadi, 2021).

The second option is using a mark-up language, like RASE of LegalRuleML.
A promising methodology for use in the AEC industry is the semantic mark-up Requirement Applicabilities Selection and Exceptions (RASE). This concept was specifically developed to transform normative documents into a set of well-defined rules which can be implemented in model checking software. It uses four tags with corresponding colors[16] to differentiate sentences in legislation, as shown in Figure 19. The uppercase tags are used for clauses, while lowercase tags are used for phrases. The biggest limitation is that the mark-up is done manually, meaning it is still laborious and error-prone (Hjelseth & Nisbet, 2011).

<R>Standard NS 11001-1, Clause: 5.2 Dimensioning an <a>access route</a> to a building
<R> The <a>access route</a> for <s>pedestrians</s> <s>wheelchair users</s> shall <r>not be steeper than 1:20</r>. <E>For <a>distances of less than 3 metres</a>, it may be steeper, but <r>not more than 1:12</r>. </E> </R>
<R>The <a>access route</a> shall have <r>clear width of a minimum of 1,8 m</r> and <r>obstacles shall be placed so that they do not reduce that width </r>. <r>Maximum cross fall shall be 2 %. </r> </R>
<R>The <a>access route</a> shall have <r>a horizontal landing at the start and end of the incline<r>, plus <r>a horizontal landing for every 0,6 m of incline </r>. <r>The landing shall be a minimum of 1,6 m deep. </r> </R>
<R><r>Minimum clear height shall be 2,25 m </r>for the full width of the defined walking zone of the entire <a>access route</a> including crossing points. </R> </R>

*Figure 19 - Manual RASE mark-up on building legislation (Hjelseth & Nisbet, 2011)*

[16] The original colors proposed by Hjelseth & Nisbeth were changed in this thesis, to ensure the visual accessibility.

Another mark-up language developed for normative knowledge is LegalRuleML, used to structure legal text in a machine-readable format. It was developed to (1) close the gap between natural language text description and semantic norm modeling, (2) provide an XML standard for modeling normative rules that satisfy legal domain requirements, and (3) extend the Linked Open Data approach to modeling from raw data to legal concepts and rules. However, LegalRuleML was not specifically developed for application in the AEC industry. Research has shown it does have enough criteria for this kind of application, but a methodology for mapping the mark-up to the existing representation in BIM models has yet to be found (Dimyadi, Governatori, & Amor, 2017).

The last possibility is by proposing a new methodology or framework, meaning updates of the building legislation can be written in a (more) machine-readable way, making the conversion process to an RDF graph less time-consuming and error-prone. This can be done by the implementation of a Controlled Natural Language (CNL), which uses a restricted set of grammar rules and vocabulary. These restrictions reduce the ambiguity and complexity of natural language and improve human readability and machine-processibility. However, a good balance between both is crucial, and more research has to be conducted (Xue, Poteet, Kao, Mott, & Braines, 2013).

### 4.2.2 Creating machine-readable rules

After processing the normative knowledge, these requirements need to be converted into machine-readable rules. To do so, Pauwels & Zhang (2015) define three strategies: using hard-coded requirements, rule-checking by querying, or using a dedicated rule language.

Hard-coded requirements are not desired, as discussed in section 2.4.

The second option is rule-checking by querying. In this method, only the building model is represented in RDF, and not the rule information. While already being more efficient than the first strategy, this still requires a lot of implementation work, since the queries still need to be fired one by one. Examples of a SPARQL implementation for compliance checking can be found in Xu & Cai (2020) and Guo, Onstein & La Rosa (2017).

The third option is using a dedicated rule language, like Semantic Web Rule Language (SWRL), N3Logic, Drools Rule Language (DRL), etc. An extensive overview of these languages can be found in Van Breedam (2017).

However, another method could be applied, by using SHACL. As already depicted in the overview on Linked Data, RDF allows a human- and machine-readable way of storing the normative data. Moreover, by directly storing the legislation in SHACL, which is an RDF, it can be used directly to check the compliance of the building data. SHACL was originally developed to check the completeness of Linked Data, like to be able to define that a defined building should have at least one story (Werbrouck, Senthilvel, Beetz, & Pauwels, 2019).

The use of a Linked Data technique like SHACL is also useful for cross-domain implementations. For example, to be able to check the energy performance, a link with data sources on the weather would be useful (Hu, et al., 2021).

There is ongoing research on the automation of SHACL constraints, e.g., the Astrea-KG incentive (Cimmino, Fernández-Izquierdo, & García-Castro, 2020). However, since there is little research on the use of SHACL for compliance-checking in the AEC industry, this is done manually in this thesis to be able to review the SHACL language for this purpose.

### 4.2.3 Storing the rules

The normative data should be stored in a white-box representation, meaning the user can access the requirements, ensuring an overview of the data. The opposite, a black box, is not desired, since the standards are not accessible to the end-user. Unfortunately, the latter is still the dominant mode of development of ACC tools for commercial systems to date.

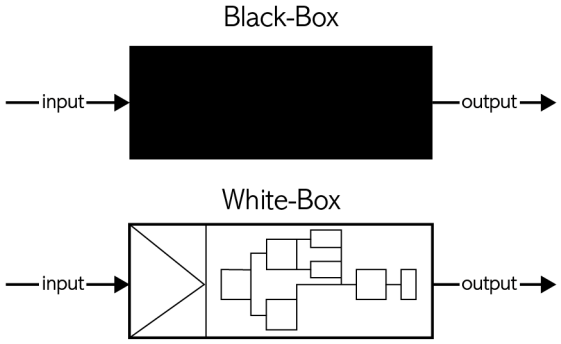


*Figure 20 - Black- and White-Box storage (Preidel & Borrmann, 2015)*

## 4.3 Building data

### 4.3.1 Limitations of BIM authoring tools

BIM modeling software typically does not support the modeling of directional aids (for people with limited vision) or alarms with a strobe light (for people who are hard of hearing). While the LBD graph can be extended, there is no ontology available to define these kinds of elements with enough detail to check the compliance. The defined constraints in this thesis will thus focus on physical accessibility, but the methodology can be used to check the compliance of each building element when defined in the data graph.

### 4.3.2 Qualitative building model

In the context of this thesis, some assumptions about the quality of the building model are made, since both on a national and an international level, the attention of designers is drawn to the perks of having a qualitative BIM model. International standards include the Information Delivery Manual (IDM) of buildingSMART,[17] while one of the main national incentives is taken by the Belgian Building Research Institute (BBRI)[18] in the form of the Modeling Guidelines.[19]

The quality of BIM models is also discussed in literature: Zhang, Teizer, Lee, Eastman & Venugopal (2013) state that every object needs to have a name, type, attributes and relations, an ID, a date, and an author. To ensure the building model contains all the necessary information, Model View Definitions (MVDs) can be defined. An MVD defined specifically for compliance checking results in a Building Compliance Model (BCM). More detailed MVDs can be defined for specific compliance checks, like a Fire Compliance Model (FCM) for example (Dimyadi, Pauwels, & Amor, 2016).

[17] https://technical.buildingsmart.org/standards/information-delivery-manual (last accessed 05/04/2022)
[18] In Dutch known as *'Wetenschappelijk en Technisch Centrum voor het Bouwbedrijf'* (WTCB)
[19] https://www.bimportal.be/nl/projecten/tc/publicaties-resultaten/bim-modelleerrichtlijnen (last accessed 21/05/2022)

Since this thesis mainly focuses on the semantics captured in the building graph, the threshold for a qualitative model is quite insignificant to be able to evaluate the accessibility of a building. The only assumptions are that the semantics match the geometry (e.g., the door width in the semantics is identical to the width as seen in the properties panel of the modeling software) and that all building elements are used as intended. The latter means that for example doors with a big glass surface are not defined as windows, although the error may not be apparent in the 3D viewer of the native software.

## 4.4 Conclusion

The methodology will make use of the RASE mark-up language to structure the normative data. Using this normative data, the requirements will be defined using SHACL, since it can capture all the constraints on the semantics needed for the compliance checking of accessibility. The SHACL constraints will validate the data in the building graph. Applying the constraints on the building data will then generate a validation report, creating feedback for the designer.

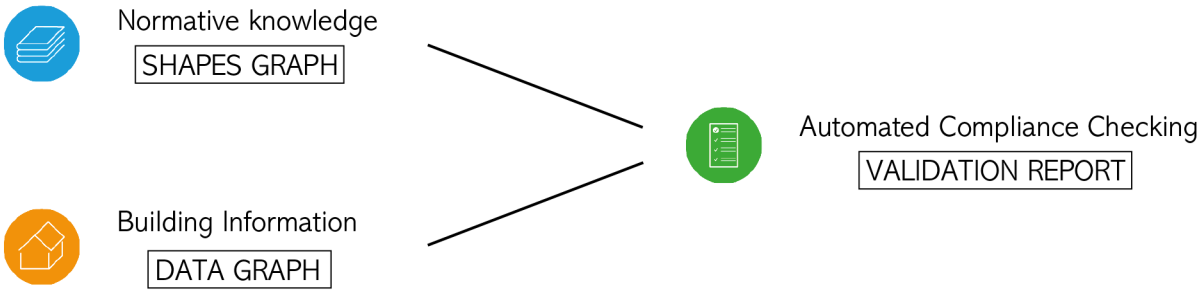


*Figure 21 - Evaluating the data graph against the shapes graph leads to a validation report*

# 5 Methodology

## 5.1 RASE mark-up on the legislation

While proposing a new framework to write (updates of) the legislation or using NLP would be the most thorough solutions, both lie either beyond the scope of the thesis or beyond the author's skillset. That's why the manual RASE markup language is used to structure the third chapter of the *'Decision by the Flemish Government to adopt a regional urban development regulation on accessibility'* of 05/06/2009.[20] This particular chapter dictates requirements for specific building elements. The result of this method can be found in Appendix A: RASE mark-up on the regulation on accessibility. Purely informational paragraphs, like the definition of the riser height of a stair, are greyed-out.

While the RASE mark-up language is an easy way to structure normative data, it is quite complex to consistently use the tags in the same manner. This is also due to sentence structures; the ideal scenario would be that a building element is an Applicability, on which Requirements are defined. This would mean that you could extract a list of defined requirements on the same building element.

For example:

> *<R> For <a> manually operated doors </a>, <r> the outer edge of the free and level turning area on the pull side of the door should touch the turning plane of the door </r>, and <r> the outer edge of the free and level turning area on the push side of the door should touch the closed-door plane </r>. </R>*
>
> *<R> In the case of a <a> manually operated door </a>, <r> a free and level wall and floor width of at least 50 cm must be provided adjacent to the crank side after finishing </r>. </R>*

These two clauses would lead to the table shown in Figure 22, where multiple requirements are defined on one building element, which functions as the applicability.

| Applicability | Requirement |
|---|---|
| manually operated door | the outer edge of the free and level turning area on the pull side of the door should touch the turning plane of the door |
| | the outer edge of the free and level turning area on the push side of the door should touch the closed-door plane |
| | a free and level wall and floor width of at least 50 cm must be provided adjacent to the crank side after finishing |

*Figure 22 - Table of requirements for a manually operated door*

However, the Applicability is often not this clearly distinct from the Requirement, as seen in the rewritten version of the first article. This phrase would lead to a less comprehensible table of requirements.

> *<R> The <a> outer edge of the free and level turning area on the pull side of a manually operated door </a> should <r> touch the turning plane of the door </r>, and <r> the outer edge of the free and level turning area on the push side of the door should touch the closed-door plane </r>. </R>*

Another difficulty is the cross-reference to other articles in the regulation since the article then should be incorporated in the table to be able to apply the cross-reference.

> *In the case of operations on changing rooms or changing cubicles accessible to the public, at least four percent of the total number of changing rooms or cubicles after the operations must comply with the provisions of article 12 and articles 22 to 26.*

The RASE language is thus not ideal to extract data from normative knowledge, but it is a useful method for manually structuring the regulation.

[20] Original title: *'Besluit van de Vlaamse Regering tot vaststelling van een gewestelijke stedenbouwkundige verordening betreffende toegankelijkheid'.* The full regulation can be found at https://www.toegankelijkgebouw.be/Regelgeving/Downloads/tabid/328/Default.aspx (last accessed 24/05/2022)

## 5.2 Dummydata

As already described in section 3.2, the LBD graph is converted from ifcOWL. While ifcOWL does contain all the geometric information of a building model, it is hard to query due to its extensivity. This is in contrast with LBD, which is much easier to query. However, the conversion[21] does typically not contain all the necessary information to apply the requirements regarding accessibility.

Since LBD is a modular data type, this section will create some added properties on the building elements and add them to the building graph. However, it is first investigated how these properties are defined in ifcOWL, to conform their presence. For this, a simple BIM model is converted to an ifcOWL graph, on which the shown figures are based.

### 5.2.1 Dimensions of a space

The definition of a space in an LBD graph does include some geometric information like the volume and perimeter, apart from properties like name and identifiers. However, to be able to check the compliance of a building project, the height and width of a space are needed.

A space is defined as an area, extruded over a certain depth, as shown in Figure 23. This figure does not contain the full graph, only the direct path from the space to the dimensions is shown. The orientation of the space, or in this case the difference between the x-dimension and the y-dimension, does not matter for this implementation. The space will comply if they are both larger than the requirement for accessibility.

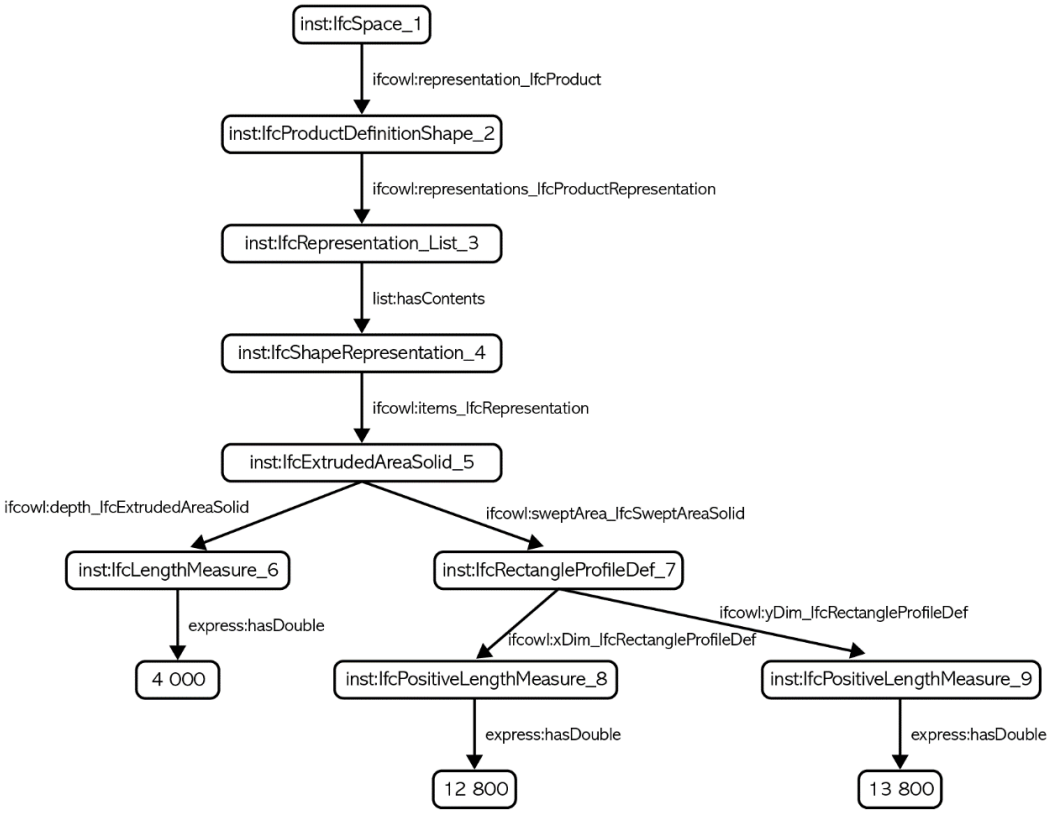


*Figure 23 - Definition of the dimensions of a rectangular space in ifcOWL*

The dummydata used in the SHACL rulebook has the predicates `props:heightIfcSpace` and `props:widthIfcSpace`.

[21] Using the IFCtoLBD converter at https://github.com/jyrkioraskari/IFCtoLBD (last accessed 31/05/2022)

### 5.2.2 Dimensions of a rampflight

A rampflight is defined without any measurements. The LBD graph only consists of identifiers and literals representing the names given in the native modeling software.

In ifcOWL, a rampflight is defined as a polyline, extruded over a certain depth. The points of this polyline are defined using cartesian coordinates, where the predicate **`hasContents`** denotes the x-coordinate, and the predicate **`hasNext`** of the same **`CartesianPoint`**, points to the corresponding y-coordinate.

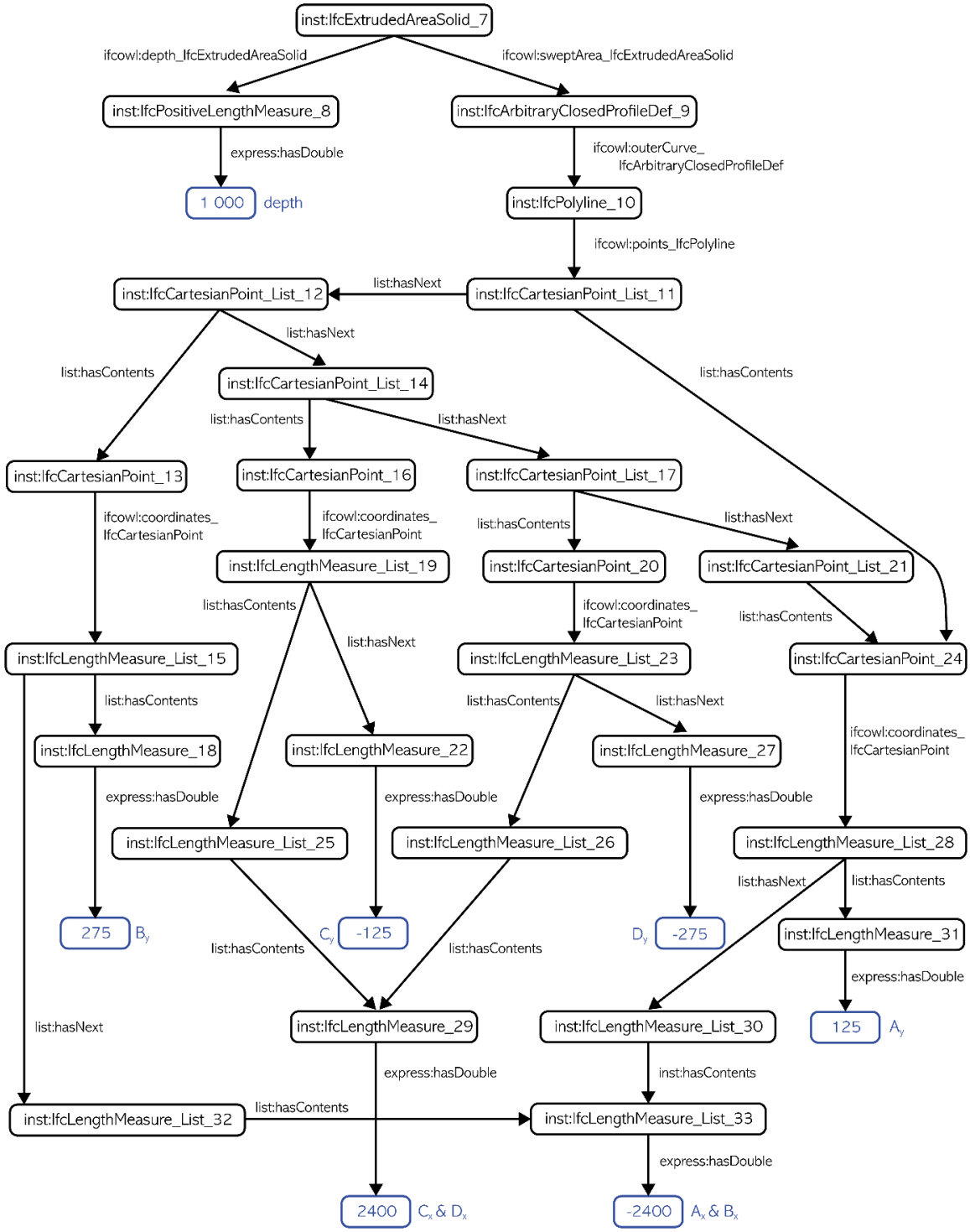


*Figure 24 - Definition of a rampflight in ifcOWL*

When plotting these points, we see that ifcOWL draws a volume on top of the actual rampflight, since the drawn measurements were 400 by 4800. [22] This means that the length of the rampflight can be calculated using the x-coordinates of a point on the left, and a point on the right. Similarly, the height of the rampflight can be calculated using the y-coordinate of the two points at the top, or the two points at the bottom.

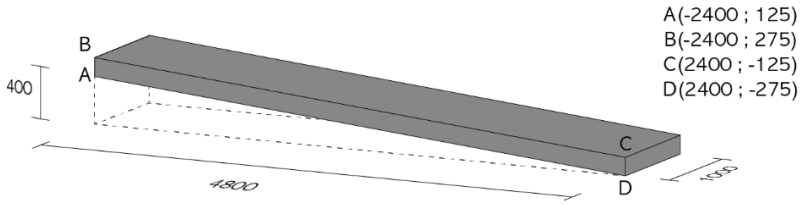


*Figure 25 - Graphical representation of a rampflight, as defined in ifcOWL*

The dummydata used in the SHACL rulebook has the predicates `props:heightIfcRampFlight`, `props:lengthIfcRampFlight,` and `props:widthIfcRampFlight`.

### 5.2.3 Dimensions of a landing

A landing, a subclass of a slab, also has no measurements in the LBD graph. Similar to a rampflight, the information only consists of identifiers and literals representing the names given in the native modeling software.

In ifcOWL, a slab is also defined as an area, extruded over a certain depth. Since the cross-section is rectangular most of the time, the derivation of the dimensions is the same as for a space.

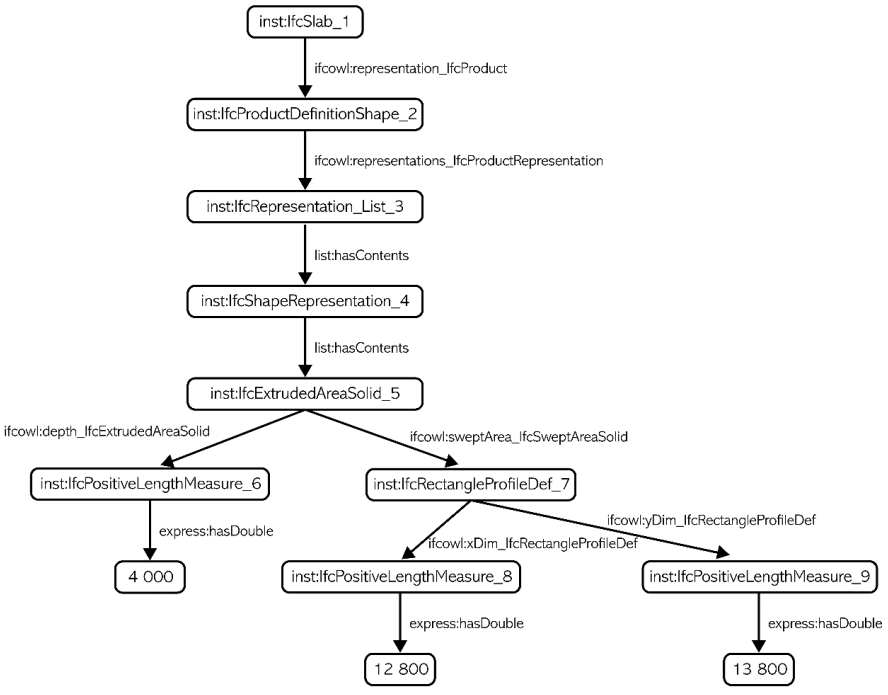


*Figure 26 - Definition of a slab in ifcOWL*

The dummydata used in the SHACL rulebook has the predicate `props:widthIfcLanding` and `props:depthIfcLanding`.

[22] This corresponds with the definition of a rampflight by buildingSMART: https://standards.buildingsmart.org/IFC/RELEASE/IFC4/ADD2_TC1/HTML/link/ifcrampflight.htm (last accessed 12/05/2022)

### 5.2.4 Insertion of the data

Different methods exist for combining the derived dummydata with the LBD data graph. The triples can be inserted into the LBD data graph, or both graphs can be merged.

The first method uses a SPARQL INSERT query, as shown in Figure 27 to complete the definition of a slab in LBD with a width and a depth, following the chosen Level 2 modeling complexity.

```
INSERT DATA
{
   slab_4                  props:widthIfcLanding          inst:widthIfcLanding_4 ;
                           props:depthIfcLanding          inst:depthIfcLanding_4 .

   inst:widthIfcLanding_4  schema:value                   "12 800"^^xsd:double .
   inst:depthIfcLanding_4  schema:value                   "13 800"^^xsd:double .
}
```

*Figure 27 - SPARQL insert query to complete the slab data*

A second method consists of the merging of the dummydata graph with the LBD graph. These can be combined by taking the set union of the triples in both graphs, for example in an RDF triplestore (Hogan, 2020). The graphs can be aligned by either the IRIs in the subject position, or by the Globally Unique Identifier (GUID) each IFC object has.[23]

[23] https://technical.buildingsmart.org/resources/ifcimplementationguidance/ifc-guid/ (last accessed 24/05/2022)

## 5.3 Rulebook

### 5.3.1 Introduction

The normative data, structured using RASE, is now converted into machine-readable rules using SHACL. The defined constraints are based on the converted building graph, which is complemented with derived dummydata to define all the necessary properties of spaces, ramps, and landings. The rules will however only be applied to the semantics contained in the data graph and do not incorporate the geometric descriptions.

Due to the nature of the legislation on accessibility and the specific individual needs of users, the default severity (`sh:Violation`) is used for every constraint, meaning nothing is classified as purely informational or a warning. A ranking of the severity of the requirements would be arbitrary and irrelevant since a fully accessible building should be the standard. Due to this objective, the field of application of the legislation is omitted.[24]

To fully understand the constraints defined on the building data, it is crucial to understand the structure of the graph first, using the example in Figure 28. A building element is defined with `rdf:type` (denoted as '`a`' in Turtle) and several properties. Normally, the sequence of letters and numbers after the underscore will be the same for the building element and its properties, but this implementation does not rely on the sequence.

The chosen Level 2 structure (as explained in section 3.2.2) means that two paths will have to be defined in the shapes graph: the first one to get the object of the '`props`' predicate, and the second one to get the object of the '`schema:value`' predicate. This can be done using sequence paths, in the form of `(props:overallHeightIfcDoor schema:value)`.

In an informal note,[25] W3C discourages this use of sequence paths. However, due to the structure of LBD (where the second path will always be `schema:value`) and the fact that this reduces the length of a simple example from 13 to 8 lines in this implementation, they are used in the declaration of parameters of functions for conciseness and readability.

```
inst:door_2
   a                                  beo:Door, bot:Element ;
   props:batid                        inst:batid_2 ;
   props:globalIdIfcRoot              inst:globalIdIfcRoot_2 ;
   props:nameIfcRoot                  inst:nameIfcRoot_2 ;
   props:objectTypeIfcObject          inst:objectTypeIfcObject_2 ;
   props:overallHeightIfcDoor         inst:overallHeightIfcDoor_2 ;
   props:overallWidthIfcDoor          inst:overallWidthIfcDoor_2 .

inst:batid_2                  schema:value      "159734" .
inst:globalIdIfcRoot_2        schema:value      "2OBrcmyk58NupXoVOHUuXp" .
inst:nameIfcRoot_2            schema:value      "M_Single-Flush:0864 x 2032mm:159734" .
inst:objectTypeIfcObject_2    schema:value      "0864 x 2032mm" .
inst:overallHeightIfcDoor_2   schema:value      "2032"^^xsd:double .
inst:overallWidthIfcDoor_2    schema:value      "864"^^xsd:double .
```

*Figure 28 - Example definition of a door using LBD*

[24] The field of application can be checked by using the quickscan at www.toegankelijkgebouw.be
[25] https://www.w3.org/TR/shacl/#property-path-sequence (last accessed 08/05/2022)

It is important to note that this implementation will not rely on the `props:nameIfcRoot` or the `props:objectTypeIfcObject` to check the compliance. While these user-defined tags normally should correspond to the properties of the building element, there is still a chance the model is less qualitative and contains wrong information. Moreover, buildingSMART defines the additional '`HandicapAccessible`' property on 158 building elements.[26] However, since the value of this property is still user-defined, this implementation will not rely on it.

The full rulebook can be found in Appendix B: rulebook.ttl.

### 5.3.2 Model View Definition

To ensure all the necessary data is available in the building graph and to prevent 'false compliance', it is defined that every used path in the rulebook should just have one value, by setting the `sh:minCount` and `sh:maxCount` both to one. This way, the SHACL rulebook functions simultaneously as an MVD. To be able to differentiate between the legislation requirements and the MVD, this `sh:message` starts with 'MVD', while the compliance constraints message starts with the article number from the legislation.

```
schema:DoorMVD
   // apply the shape to a focus node
   a                sh:NodeShape ;
   // target all nodes with class 'Door'
   sh:targetClass   beo:Door ;
   // target a property of each 'Door'
   sh:property [
     // target the height predicate
     sh:path        props:overallHeightIfcDoor ;
     // each 'Door' should have at least one height property
     sh:minCount    1 ;
     // each 'Door' should have at most one height property
     sh:maxCount    1 ;
     // if not, return the violation message
     sh:message     "MVD - each door should have exactly one 'overallHeightIfcDoor'." ;
   ] .
```

*Figure 29 - Shape defining that each space should have exactly one height property*

While the MVD could be checked in a separate shapes file, both are combined in this implementation. However, the building constraints are defined in separate shapes, to keep both clear. The MVD is declared first in Appendix B: rulebook.ttl, up until line 90.

### 5.3.3 Types of constraints

Different types of constraints are applied to the data graph to investigate accessibility. These constraints can be categorized as value, relational and mathematical constraints. Furthermore, some inference rules and logical constraint components will be applied to the constraints to be able to define all the different requirements.

SHACL allows more types of constraints than these four, but for example string-based constraints are not relevant for this thesis, since user-defined labels or names from the modeling software are omitted.

[26] https://search.bsdd.buildingsmart.org/Property/Index/116507 (last accessed 15/05/2022)

#### 5.3.3.1 Value constraints

SHACL defines four value constraints, which allow checking if a value is smaller/bigger than (or equal to) a defined limit value.

This method is used for constraints on values already defined in the building graph, meaning no additional mathematical operations are needed to check the compliance of the building element. In Figure 30, an example of a value constraint is given by defining that all doors must have a minimum height of 2090 mm.

```
schema:Door
   // apply the shape to a focus node
   a                sh:NodeShape ;
   // target all nodes with class 'Door'
   sh:targetClass   beo:Door ;
   // target a property of each 'Door'
   sh:property [
     // target the height predicate
     sh:path        props:overallHeightIfcDoor ;
     // name the object of this height predicate path
     sh:property    schema:DoorHeight ;
   ] .

// the named object is now a subject
schema:DoorHeight
   // target a property of the focus node
   a                sh:PropertyShape ;
   // target the value predicate
   sh:path          schema:value ;
   // the object should be more than 2090 mm
   sh:minInclusive  "2090"^^xsd:double ;
   // if not, return the violation message
   sh:message       "Art. 22 §1 - The door height must be at least 2090 mm." .
```

*Figure 30 - Example of a value constraint*

The following requirements are defined in the SHACL rulebook with only value constraints.

| Legislation | |
|---|---|
| Art. 12 §2 | The clearance height (after finishing) must be at least 2300 mm. |
| Art. 14 | The width of the walkway (after finishing) must be at least 1500 mm. |
| Art. 19 §2 | The landing of a ramp must be at least 1200 by 1500 mm. |
| Art. 20 §1 | The width of a staircase (after finishing) must be at least 1000 mm. |
| Art. 20 §2 | A staircase must have an intermediate landing after at most 17 treads. |
| Art. 20 §3 | The riser height must be at most 180 mm. |
| Art. 20 §3 | The tread length must be at least 230 mm. |
| Art. 22 §1 | The door height (after finishing) must be at least 2090 mm. |
| Art. 22 §2 | The door width (after finishing) must be at least 900 mm. |

*Figure 31 - The requirements based on value constraints*

The IFC4 ADD2 TC1 EXPRESS schema[27] defines the dimensions of a space as the dimensions between the finishing on opposite sides, hence why the requirement after finishing is used when defining the constraints.[28]

Since a stairflight is defined as *'an assembly of building components in a single "run" of stair steps (not interrupted by a landing)'*, the constraint can be defined that each stairflight can consist of at most 17 steps. Furthermore, buildingSMART states that all riser heights and tread lengths should be equal for all steps of a stair flight. This requirement is also defined in the legislation on accessibility but is not considered since the IFC schema already assumes this.[29]

For winders, the walking line is defined, and the tread length is then measured at the walking line. The building graph only includes the minimal tread at this offset, meaning that the compliance of this smallest tread is enough.[30]

For doors, the requirement for the width after finishing is used instead of the dimensions of the casing, since the attribute definitions of buildingSMART state that the overall width is the door opening width, and not the total width of the door lining. The corresponding definition is used for the door height.[31]

#### 5.3.3.2 Relational constraints

Relational constraints are defined to check if a building element is present in relation to another building element in the project. For this `sh:qualifiedValueShape` is used, to check that a specified number of nodes conforms to the targeted shape. This specified number is defined with `sh:qualifiedMinCount`.

An example of a purely relational constraint is that all stairs should have a railing on both sides.

```
schema:Stair
   // apply the shape to a focus node
   a                          sh:NodeShape ;
   // target all nodes with class 'Stair'
   sh:targetClass             beo:Stair ;
   // target a property of each 'Stair'
   sh:property [
     // target the subelement predicate
     sh:path                  bot:hasSubElement ;
     // in the object, target the class 'Railing'
     sh:qualifiedValueShape   [ sh:class beo:Railing ] ;
     // 2 elements of this class should be present
     sh:qualifiedMinCount     2 ;
     // if not, return the violation message
     sh:message               "Art. 20 §4 - All stairs must have a railing on both sides." ;
   ] .
```

*Figure 32 - Example of a relational constraint*

[27] This schema was published in October 2017 and is the official buildingSMART standard at the time of writing.
[28] https://standards.buildingsmart.org/IFC/RELEASE/IFC4/ADD2_TC1/HTML/link/ifcspace.htm (last accessed 30/04/2022)
[29] https://standards.buildingsmart.org/IFC/RELEASE/IFC4/ADD2_TC1/HTML/schema/ifcsharedbldgelements/lexical/ifcstairflight.htm (last accessed 30/04/2022)
[30] https://standards.buildingsmart.org/IFC/RELEASE/IFC4/ADD2_TC1/HTML/link/pset_stairflightcommon.htm (last accessed 07/05/2022)
[31] https://standards.buildingsmart.org/IFC/RELEASE/IFC4/ADD2_TC1/HTML/schema/ifcsharedbldgelements/lexical/ifcdoor.htm (last accessed 30/04/2022)

This requirement is the only one based on purely relational constraints. Since the assumption of a qualitative building model is made, requiring two railings on one stairflight implies they are positioned on opposite sides of the stairflight.

#### 5.3.3.3 Mathematical constraints

Mathematical functions in SHACL can be defined using SPARQL queries, thus making use of the extensions of the SHACL-Core, named SHACL-SPARQL. A SHACL function always consists of a declaration of the parameters and a return type, as shown in Figure 33. The function can then be used in a Node- or PropertyShape, using `sh:expression`. Since the function definitions in this rulebook use no optional values, the MVD does not include the parameters of the function.

```
schema:stairFormula
   // define a function
   a                  sh:SPARQLFunction ;
   // name the function
   rdfs:comment       “stair formula” ;
   // declare the first parameter
   sh:parameter [
     // first parameter is op1
     sh:path          inst:op1 ;
     // op1 is a decimal
     sh:datatype      xsd:double ;
     // op1 corresponds to the riser height
     sh:description   “riser height” ;
   ] ;
   // declare the second parameter
   sh:parameter [
     // second parameter is op2
     sh:path          inst:op2 ;
     // op2 is a decimal
     sh:datatype      xsd:double ;
     // op2 corresponds to the tread length
     sh:description   “tread length” ;
   ] ;
   // the returned value is a decimal
   sh:returnType      xsd:double ;
   // start a SPARQL select query
   sh:select          “””
     // calculate the stair formula and return the result
     SELECT ( ( 2 * $op1 + $op2) AS ?result)
     WHERE {
     }
    “””  .
```

*Figure 33 - Example of a mathematical constraint*

The stair formula is the only requirement defined with only mathematical constraints.

#### 5.3.3.4 Inference

To be able to combine the constraints, some inference statements, like 'if…then…' have to be created. In SHACL, this can be mainly done using `sh:node`, which enables to define that each value node conforms to a given node shape. To properly define the inference rules, some logical constraint components like `sh:not`, `sh:or`, `sh:and`, `sh:xone` are also needed.

```
schema:Slopes
   // apply the shape to a focus node
   a                sh:NodeShape ;
   // target all nodes with class 'Rampflight'
   sh:targetClass   beo:RampFlight ;
   // both constraints should be fulfilled
   sh:and (
     [
       // first constraint: the slope is less than 10%
       sh:expression [
         inst:lessThan (
           [ inst:slope (
             [ sh:path (props:heightIfcRampFlight schema:value) ]
             [ sh:path (props:lengthIfcRampFlight schema:value) ] ) ]
           10
         ) ;
       ]
     ]
     // if the slope is less than 10%, check the next constraint
     sh:node schema:SlopeConstraint
   )
   // if not, return the violation message
   sh:message       "Art. 19 §1 - The slope of a rampflight can be at most 10% when the
                    difference in height is less than 100 mm."
   ] .
```

*Figure 34 - Excerpt of the slope definition, defining that a ramp with a slope of less than 10% needs to conform to the SlopeConstraint too*

Important to note is that `sh:or` specifies that each targeted node conforms to at least one of the provided shapes. This means that for constraints like Art 19 §5: *'The width of a rampflight (after finishing) must be at least 1200 mm, if the slope is more than 4%.'*, the structure of the constraints is as follows:

```
// data should conform to at least one constraint
sh:or (
  // data should conform to both of the following shapes
  sh:and (
    // if the slope is more than 4%
    [ slope is more than 4% ]
    // the width constraint should be fulfilled
    [ to additional constraint for width ]
  )
  // else, the slope should be less than 4% (and no additional constraint is defined)
  [ slope is less than 4% ]
)
```

*Figure 35 - Structure of logical constraint components*

#### 5.3.3.5 Combination of constraints

Using these three types of constraints in combination with inference rules, and logical constraint components, all the requirements on the semantics are defined in the SHACL rulebook.

| Legislation | |
|---|---|
| Art. 19 §1 | The slope of a rampflight can be at most 10% when the difference in height is less than 100 mm. |
| Art. 19 §1 | The slope of a rampflight can be at most 8.3% when the difference in height lies between 100 and 250 mm. |
| Art. 19 §1 | The slope of a rampflight can be at most 6.25% when the difference in height lies between 250 and 500 mm. |
| Art. 19 §1 | The slope of a rampflight can be at most 5% when the difference in height is more than 500 mm. |
| Art. 19 §4 | A ramp must have an intermediate landing if the difference in height is more than 500 mm. |
| Art. 19 §5 | The width of a rampflight (after finishing) must be at least 1200 mm, if the slope is more than 4%. |

*Figure 36 - The requirements based on a combination of constraints*

## 5.4 Conclusion

The first step of the methodology was to structure the normative data using the RASE markup language. While this is a good way to manually decompose complex sentences, the sentence structures would have to be more straightforward to be able to automate the extraction of the different components.

The second step was to add some properties to the definition of spaces, rampflights, and landings in the building graph, needed to check the compliance with the regulation on accessibility.

The last step was to create a rulebook in Turtle format, using SHACL. This rulebook first defines an MVD to ensure all the necessary data is present in the building graph. It then uses a combination of three types of constraints (value, relational and mathematical), inference statements (if…then…), and logical constraint components (and, or) to define different requirements for the accessibility of the building project.

# 6 Proof of concept

## 6.1 Introduction

Using the SHACL rulebook, or shapes graph, the building, or data graph, can be validated. To do so, a proof of concept application is created, which is divided into a frontend and a backend, and both can be found on https://github.com/NuytsE.

## 6.2 Conversion workflows

To be able to check the compliance of the building design, the IFC file should be converted to an RDF graph. To accomplish this, different converters exist, like IFCtoLBD.[32] This converter first transforms the file to a (temporary) ifcOWL graph, which is then converted to an LBD ABOX graph. The converter follows the graph traversal, using path templates. The newly created instances get the right LBD OWL classes by using LBD modular ontologies like BOT, PRODUCT, and PROPS. If a higher modeling complexity is chosen by the user, new instance nodes for the properties are created (Bonduel, et al., 2018).

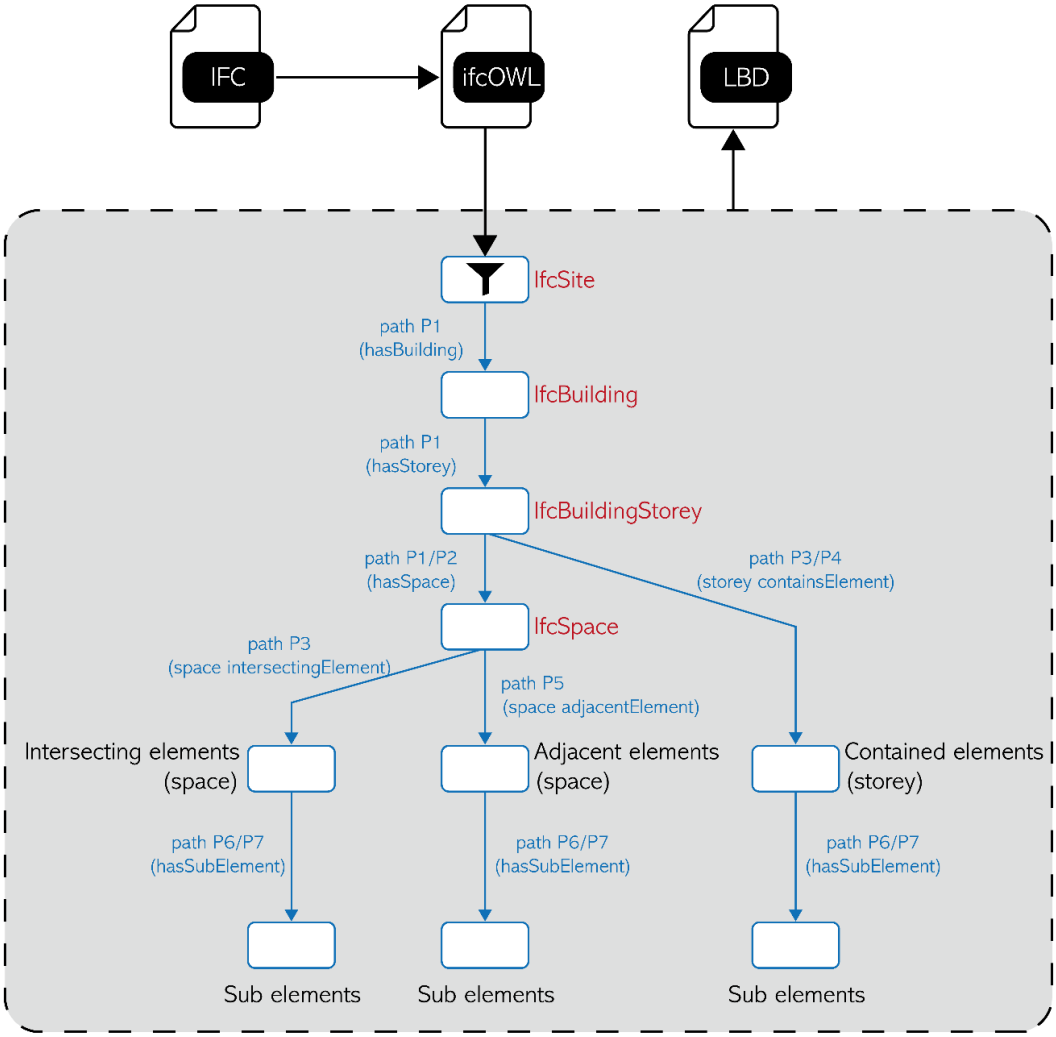


*Figure 37 - The conversion workflow from IFC to LBD, reworked from (Bonduel, et al., 2018)*

[32] https://github.com/jyrkioraskari/IFCtoLBD (last accessed 31/05/2022)

## 6.3 Implementation

For the demonstration of the tool, a student project of a library will be used. This model was however changed for this demonstration since the original design was fully physically accessible. It was changed in a way that each constraint is shown exactly once in the validation report.


*Figure 38 - Render of the student project*

The converted building graph can be uploaded on the User Interface (UI). The used graph is written in Turtle syntax, explaining the .ttl filetype. The file is not stored in a public server, but in a separate folder in the project directory of the backend. Since access control lies beyond the scope of this thesis, this choice was made to ensure the details of the building project are not online available, maintaining the property rights and safety. Hence why a (not-executed) student project was chosen for the demonstration since this is publicly available on GitHub.[33]

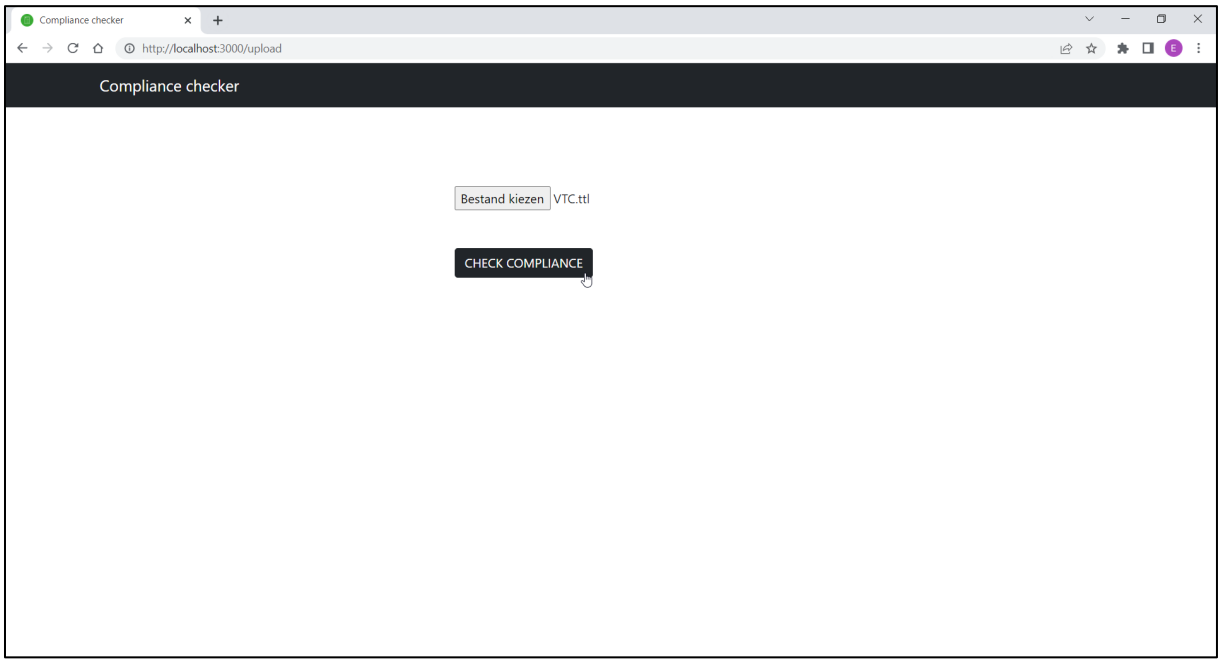


*Figure 39 - User Interface of the upload page*

[33] https://github.com/NuytsE/ACC-back/blob/main/files/LBDFile.ttl

Then, this LBD graph is checked against the created SHACL rulebook, meaning both the completeness of the data and the accessibility of the project are checked. Since there is no JavaScript implementation of SHACL-SPARQL (needed for the mathematical constraints), at the time of writing, pySHACL[34] is used. The python command is executed in the application using the 'child_process' module. Since pySHACL returns a purely textual report, there were some string methods needed to create a UI of the validation report. [35] This report can be downloaded locally. In the report, the transparency to the user is key: apart from the message containing the building legislation, both the source shape and the focus node are returned, meaning it is easily checked exactly which building element does not comply with said shape. Moreover, the identifier of the element is returned, ensuring that the building element can be easily found and altered in the native modeling software. This identifier is not incorporated in the validation report of pySHACL, but is retrieved by querying for the non-conforming element in the building graph.

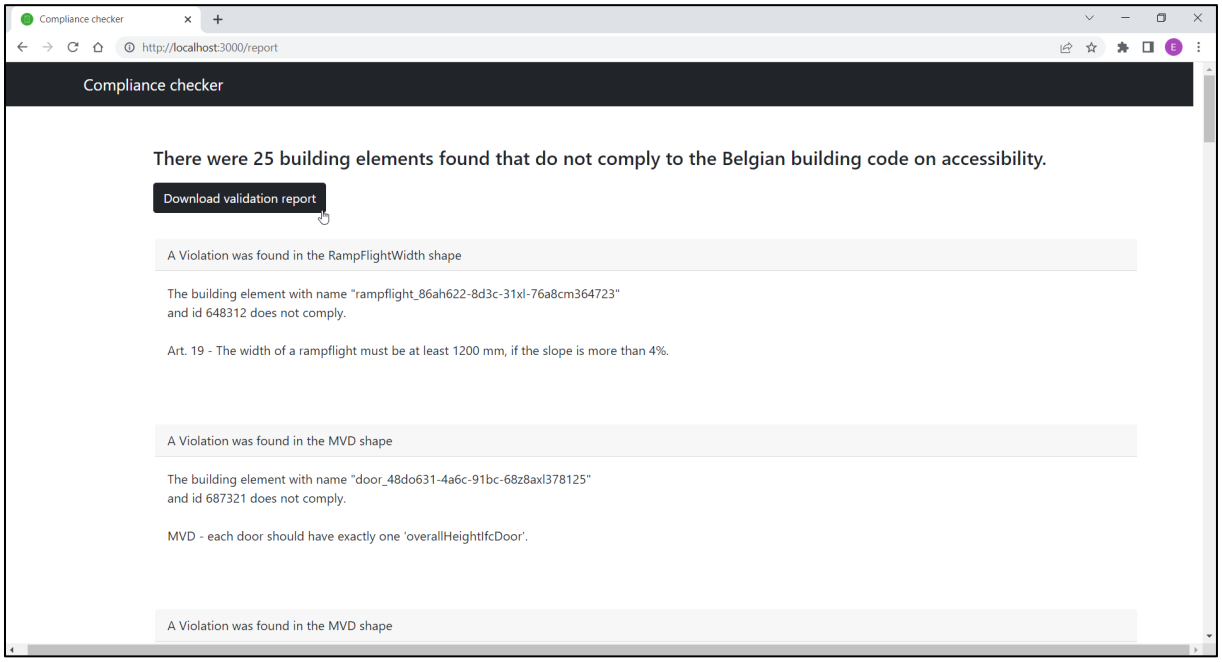


*Figure 40 - User Interface of the non-conforming validation report*

After altering the remarks in the BIM authoring tool, the model can be converted to an RDF and uploaded again, leading to this conforming validation report.

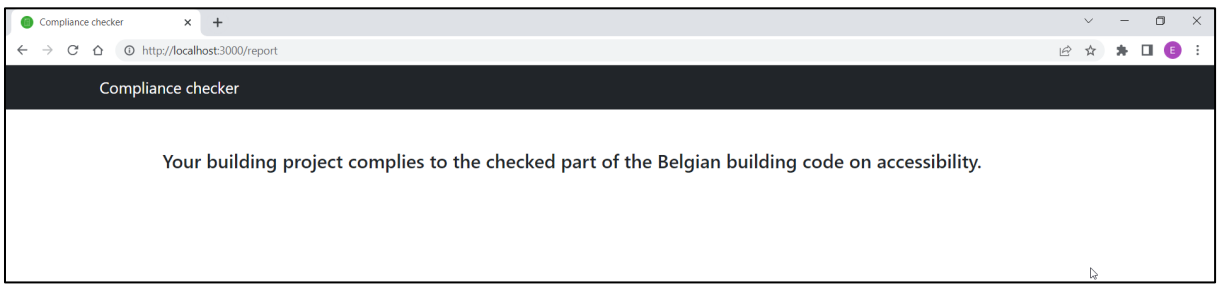


*Figure 41 - User Interface of the conforming validation report*

[34] https://github.com/RDFLib/pySHACL (last accessed 15/05/2022)
[35] The full report can be found at https://github.com/NuytsE/ACC-back/blob/main/files/report

## 6.4 Conclusion

Although this specific implementation focuses on accessibility, the rulebook can be easily extended for other legislation, since it is stored in a separate file on the backend of the tool. This also means the rules can be updated to the most recent version of the building code, the user can define some extra requirements or alter existing requirements (for example for renovations, or to change the units of the checked legislation). Of course, the created validation report is only feedback on the accessibility of the design, meaning the designer will have to manually alter the project in the native modeling software. The tool can also be used on as-built models, evaluating the design of already built projects. However, accessibility also requires effort during the life cycle of the building. Creating a narrower space than the one checked in the tool due to additional interior elements will certainly harm the accessibility since the checked model is only a digital twin of the actual building.

# 7 Evaluation and conclusion

## 7.1 Evaluation of SHACL for use in compliance checking

To evaluate the suitability of SHACL for use in compliance checking, the four key requirements for automated code checking as defined by Greenwood et al. (2010) can be used:

- The computer programmed rules must be easily understood by the regulation authors;
- The lifecycle of the rule base must be independent of software and schema updates;
- All development must comply with Open Standards;
- Consideration must be given to the industry processes of model authoring.

While the first requirement is quite subjective, the Turtle syntax is generally seen as the most human-readable syntax for RDF (Hogan, 2020). However, the question arises if regulation authors without any knowledge of programming could understand this. This relates to the differentiation between *readable* and *understandable*, as explained in section 2.2. But in this case, there is a nuance: while the regulation is specifically written to be understandable by architects (and is apparently not), this rulebook is mostly written to automatically check compliance. The user will have some understanding of what is being checked, but a full understanding of the underlying structures is less necessary. The RDF shapes can however be converted into a human-friendly interface if needed. The second requirement is achieved by implementing LBD, which is independent of software packages since it is derived from the open and neutral IFC schema. The third requirement is fulfilled by using Semantic Web technology like SHACL, which is a W3C recommendation. The last requirement is partially avoided by implementing an MVD in the constraints in the rulebook. This does not improve the quality of the building model but does check if all the necessary data is present before applying the constraints.

However, SHACL only allows the compliance checking of prescriptive legislation. While the regulation on accessibility only consists of prescriptive requirements, other types of building legislation use performance-based regulation, like fire safety or energy efficiency.

A disadvantage of SHACL is that the constraints are defined in one unit of measurement, and there is no conversion possible in the language. This means that either the building data and the constraints need to be modeled in the same unit, or some implementation work is needed to convert the units in one of the graphs.

## 7.2 Evaluation of the implementation

Unfortunately, there is no implementation of SHACL-SPARQL in JavaScript at the time of writing. The module 'rdf-validate-shacl'[36] exists, but this only supports SHACL-Core constraints. The perk of this module is that the validation report is a JavaScript Object, meaning some built-in methods like `result.conforms` or `result.focusNode` can be used to extract the results separately and map them in a visual report on the frontend.

While the python module pySHACL does implement SHACL-SPARQL, and the Command Line Use can be run in JavaScript with an extra module, the resulting validation report is a purely textual file. This means that some string methods were needed to transform them into a JavaScript Object and successfully create a UI of the report.

An advantage of this react-application is that the rulebook is stored in a separate file. This means that when updating or adding the constraints, the architecture of the application does not have to be changed, only the contents of the rulebook.ttl change.

[36] https://github.com/zazuko/rdf-validate-shacl (last accessed 17/05/2022)

## 7.3 Evaluation in the entirety of the building legislation

While this thesis focuses on the regulation of accessibility, the method can be used on each kind of prescriptive building code, as long as the necessary data is (made) present in the building graph. For example for fire safety, the fire resistance of building elements, the compartmentation area, the presence of sprinklers or smoke- and heat extraction, the number of exits… can be checked. The method can also be applied to acoustical codes, daylight regulation, or energy efficiency for example.

This research studied the definition of constraints on semantic data in a building graph, although geometrical constraints are also needed in terms of creating an accessible built environment. While the evaluation of Inter (2019) does not name all the requirements that are overlooked by architects or people in the urban planning department, the layout of sanitary blocks is one. Other named requirements like insufficient door widths or slopes that are too steep are however checked with this methodology.

## 7.4 Future work

This research is certainly not the full solution for Automated Compliance Checking of every building code. More work can be done on the geometrical requirements since this thesis focuses on the semantics defined in the building graph. For example, the width of a walkway is purely defined on the dimensions of a space and does not consider the presence of obstructing elements (like radiators or fire hose reels), also turning circles cannot be checked using the proposed methodology. Furthermore, the derived dummydata would ideally be incorporated into LBD automatically, where this research used manual SPARQL UPDATE queries. Lastly, automation of the SHACL constraints would be a big improvement but is, unfortunately, beyond the scope of this research due to the author's skillset.

## 7.5 Conclusion

The developed methodology consisted of three main decisions: using RASE to structure the normative data, using LBD to create the building graph, and using SHACL to define the constraints. While the RASE markup proved to be an effective way to manually structure the legislation, it is quite laborious and the efficiency is highly dependent on sentence structures. Representing the building data in an LBD graph was successful since this graph is easy to query and understand. Some extra data had to be derived from the ifcOWL graph to be able to check the compliance, but this does only prove the modular aspect of LBD. Furthermore, SHACL constraints can easily be defined on the semantic data in an LBD graph, although further research is needed to evaluate geometrical constraints. Moreover, the combination of an MVD to check if all the necessary data is present in the graph and checking the compliance appears to be interesting in the context of the Semantic Web. In short, the developed methodology can be used to check any type of prescriptive building legislation, as long as the data is defined in the building graph.

While the proposed methodology cannot (yet) incorporate all the requirements of the regulation on accessibility, it is hopefully a starting point for the creation of a more accessible and inclusive environment.

# Bibliography


Amor, R., & Dimyadi, J. (2021, March). The promise of automated compliance checking. *Developments in the Built Environment, 5*. doi: https://doi.org/10.1016/j.dibe.2020.100039

Beach, T. H., Hippolyte, J.-L., & Rezgui, Y. (2020, October). Towards the adoption of automated regulatory compliance checking in the built environment. *Automation in Construction*. doi: https://doi.org/10.1016/j.autcon.2020.103285

Bonduel, M., Oraskari, J., Pauwels, P., Vergauwen, M., & Klein, R. (2018). The IFC to Linked Building Data Converter - Current Status. *Proceedings of the 6th Linked Data in Architecture and Construction Workshop*, (pp. 34-43). London, United Kingdom.

Cimmino, A., Fernández-Izquierdo, A., & García-Castro, R. (2020). Astrea: Automatic generation of SHACL shapes from ontologies. *European Semantic Web Conference* (pp. 497-513). Heraklion, Greece: Springer. doi: https://doi.org/10.1007/978-3-030-49461-2_29

Corman, J., Reutter, J. L., & Savkovic, O. (2018). Semantics and validation of recursive SHACL. *International Semantic Web Conference* (pp. 318-336). Monterey: Springer. doi: https://doi.org/10.1007/9783-030-00671-6_19

Curtotti, M., & McCreath, E. (2013). A right to access implies a right to know: an open online platform for research on the readability of law. *Open Access to Law, 1*.

Dimyadi, J., & Amor, R. (2013, May). Automated building code compliance checking - where is it at? *19th International CIB World Building Congress*. doi: http://dx.doi.org/10.13140/2.1.4920.4161

Dimyadi, J., Governatori, G., & Amor, R. (2017). Evaluating LegalDocML and LegalRuleML as a Standard for Sharing Normative Information in the AEC/FM Domain. *Proceedings of the Joint Conference on Computing in Construction (JC3)*, *1.* Heraklion, Greece. doi: http://dx.doi.org/10.24928/JC32017/0012

Dimyadi, J., Pauwels, P., & Amor, R. (2016, November). Modelling and accessing regulatory knowledge for computer-assisted compliance audit. *Journal of Information Technology in Construction*, 317-336. Retrieved from https://www.itcon.org/2016/21

Greenwood, D., Lockley, S., Malsane, S., & Matthews, J. (2010). Automated compliance checking using Building Information Models. *The construction, building and real estate research conference of the Royal institution of Chartered Surveyors.* Paris, France: RICS.

Guo, D., Onstein, E., & La Rosa, A. D. (2017, August). A semantic approach for automated rule compliance checking in construction industry. *IEEE Access*. doi: http://dx.doi.org/10.1109/ACCESS.2021.3108226

Hjelseth, E., & Nisbet, N. (2011). Capturing normative constraints by use of the semantic mark-up RASE methodology. *Proceedings of the CIB W78-W102.* Sophia Antipolis, France.

Hogan, A. (2020). *The Web of Data.* Springer. doi: https://doi.org/10.1007/978-3-030-51580-5

Hu, S., Wang, J., Hoare, C., Li, Y., Pauwels, P., & O'Donnell, J. (2021, April). Building energy performance assessment using Linked Data and cross-domain semantic reasoning. *Automation in Construction, 124*. doi: https://doi.org/10.1016/j.autcon.2021.103580

Inter. (2019). *Evaluatieonderzoek Vlaamse toegankelijkheidsverordening.*

Knublauch, H. (2017, August 17). *SHACL and OWL compared*. Retrieved from https://spinrdf.org/shacl-and-owl.html

Labra Gayo, J. E., Prud'hommeaux, E., Solbrig, H., & Boneva, I. (2016). *ShEx vs SHACL.*

Oraskari, J., Senthilvel, M., & Beetz, J. (2021). SHACL is for LBD what mvdXML is for IFC. *Proceedings of the Conference CIB W78*, (pp. 693-702). Esch-sur-Alzette, Luxembourg.

Pauwels, P., & Petrova, E. (2020). *Information in Construction.*

Pauwels, P., & Roxin, A. (2016). SimpleBIM: From full ifcOWL graphs to simplified building graphs. *European Conference on Product and Process Modelling.* Limasol, Cyprus.

Pauwels, P., & Zhang, S. (2015). Semantic rule-checking for regulation compliance checking: an overview of strategies and approaches. *Proceedings of the 32nd CIB W78 Conference*, (pp. 619-628). Eindhoven, Netherlands.

Pauwels, P., Zhang, S., & Lee, Y.-C. (2017, January). Semantic web technologies in AEC industry: a literature overview. *Automation in Construction, 73*, 145-165. doi: https://doi.org/10.1016/j.autcon.2016.10.003

Preidel, C., & Borrmann, A. (2015). Automated code compliance checking based on a visual language and Building Information Modeling. *International Symposium on Automation and Robotics in Construction and Mining.* Oulu, Finland. doi: http://dx.doi.org/10.13140/RG.2.1.1542.2805

Rasmussen, M. H., Lefrançois, M., Bonduel, M., Hviid, C. A., & Karlshoj, J. (2018). OPM: an ontology for describing properties that evolve over time. *Proceedings of the 6th Linked Data in Architecture and Construction Workshop. 2159*, pp. 24-33. London, United Kingdom: CEUR Workshop Proceedings.

Rasmussen, M. H., Lefrançois, M., Schneider, G. F., & Pauwels, P. (2020, November). BOT: the Building Topology Ontology of the W3C Linked Building Data Group. *Semantic Web Journal*. doi: http://dx.doi.org/10.3233/SW-200385

RIBA. (2012). *BIM Overlay to the RIBA Outline Plan of Work.* London, United Kingdom: RIBA Publishing.

Royal Decree of December 7. (2016, December 7). 30.

Rudman, R., & Bruwer, R. (2016, February). Defining Web 3.0: opportunities and challenges. *The Electronic Library*, 132-154. doi: http://dx.doi.org/10.1108/EL-08-2014-0140

Vaes, S., Votquenne, D., Baert, D., & Arnoudt, R. (2022, may 19). *Hoe (on)toegankelijk is Vlaanderen? "Terzake" rijdt mee met twee rolstoelgebruikers*. Retrieved may 22, 2022, from VRT NWS: https://www.vrt.be/vrtnws/nl/2022/05/19/toegankelijkheid-terzake/

Van Breedam, V. (2017). *Onderzoek naar automatisering van de evaluatie van brandnormering in een BIM model.* Universiteit Gent.

Vande Reyde, M., Saeys, F., Van Cauter, C., De Vroe, G., Van Volcem, M., & Ongena, T. (2020). *Conceptnota voor nieuwe regelgeving.*

Wagner, A., Bonduel, M., Pauwels, P., & Rüppel, U. (2020, July). Representing construction-related geometry in a Semantic Web context: a review of approaches. *Automation in Construction, 115*. doi: https://doi.org/10.1016/j.autcon.2020.103130

Werbrouck, J., Senthilvel, M., Beetz, J., & Pauwels, P. (2019). A checking approach for distributed building data. *Forum Bauinformatik*, (pp. 173-181). Berlin, Germany. doi: https://doi.org/10.14279/depositonce-8763

Xu, X., & Cai, H. (2020, January). Semantic approach to compliance checking of underground utilities. *Automation in Construction, 109*. doi: https://doi.org/10.1016/j.autcon.2019.103006

Xue, P., Poteet, S., Kao, A., Mott, D., & Braines, D. (2013). Constructing controlled English for both human usage and machine processing. *Proceedings of the 7th International Rule Challenge, the HLT and the DC at RuleML2013*, (pp. 175-189). Seattle, United States of America.

Zhang, J., & El-Gohary, N. M. (2017, January). Integrating semantic NLP and logic reasoning into a unified system for fully-automated code checking. *Automation in Construction, 73*, 45-57. doi: https://doi.org/10.1016/j.autcon.2016.08.027

Zhang, S., Teizer, J., Lee, J.-K., Eastman, C. M., & Venugopal, M. (2013, January). Building Information Modeling (BIM) and safety: automatic safety checking of construction models and schedules. *Automation in Construction, 29*, 183-195. doi: https://doi.org/10.1016/j.autcon.2012.05.006

# Appendix A: RASE mark-up on the regulation on accessibility

**Besluit van de Vlaamse Regering tot vaststelling van een gewestelijke stedenbouwkundige verordening betreffende toegankelijkheid (5 juni 2009)**
Dit is de gecoördineerde versie van de toegankelijkheidsverordening. Deze is volledig in werking sinds september 2011.

**Hoofdstuk III. Normbepalingen**
**Afdeling I. Algemene bepalingen**
**Art. 11.**
<R> <a> Indien een aanvraag valt onder het toepassingsgebied van deze stedenbouwkundige verordening </a>, dan wordt in de stedenbouwkundige vergunning opgelegd dat de <r> normbepalingen van hoofdstuk III dienen te worden nageleefd </r>. </R>

**Art. 12.**
**§1.** <R> In <a> alle delen van een constructie </a> waarop dit besluit van toepassing is, moet voor een <r> vrije en vlakke draairuimte </r> worden gezorgd. </R>
**§2.** <R> Met behoud van de toepassing van artikel 22, alinea 1, moet in <a> alle delen van een constructie </a> waarop dit besluit van toepassing is, <r> een vrije doorgangshoogte, na afwerking, van minstens 2,30 meter </r> gegarandeerd worden. <E> Deze verplichting geldt niet als de realisatie ervan tot een constructieprobleem leidt op bovenliggende verdiepingen waar geen werken aan gepland waren. </E> </R>

**Art. 13.**
<R> Het <a> traject naar de delen van een constructie die de publiek toegankelijke functie vervullen </a>, moet met <r> gids- of geleidelijnen </r> aangeduid worden. </R>
[Met een geleidelijn wordt een speciaal voor de geleiding van blinden of slechtzienden aangebracht kunstmatig element bedoeld, dat voor blinden en slechtzienden bruikbaar is als oriëntatiepunt of als ononderbroken geleiding bij het lopen. Met een gidslijn wordt een natuurlijk in de ruimte aanwezig element bedoeld dat, hoewel het niet speciaal voor de geleiding van blinden of slechtzienden werd aangebracht, voor blinden en slechtzienden bruikbaar is als oriëntatiepunt of als ononderbroken geleiding bij het lopen.]

**Afdeling II. Bepalingen met betrekking tot looppaden naar constructies en naar de daarin gelegen vertrekken**
**Art. 14.**
<R> De <a> breedte van een looppad </a> dat zich niet <e> tussen binnenmuren </e> bevindt en dat niet valt onder de <e> bepalingen van het besluit van de Vlaamse Regering van 29 april 1997 houdende vaststelling van een algemene bouwverordening inzake wegen voor voetgangersverkeer </e>, bedraagt <r> minstens 150 cm </r>. </R>
<E>In afwijking van het eerste lid is een <a> versmalling </a> van een dergelijk looppad toegestaan in de volgende gevallen:
**1.** bij een versmalling die zich over <r> hoogstens 120 cm </r> uitstrekt: als ter hoogte van die versmalling een <r> doorgang van minstens 90 cm </r> gegarandeerd wordt;
**2.** bij een versmalling die zich over <r> meer dan 120 cm uitstrekt </r>: als ter hoogte van die versmalling een <r> doorgang van minstens 120 cm </r> gegarandeerd wordt en <a> minstens elke tien meter </a>, alsook aan het <a> begin en het einde van de versmalling </a> voor een <r> vrije en vlakke draairuimte </r> wordt gezorgd. </E>

**Art. 15.**
<R> De <a> breedte van een looppad, gemeten tussen onafgewerkte binnenmuren </a>, bedraagt <r> minstens 175 cm </r>, zodat na de afwerking van de wanden en met inbegrip van de ruimte voor plinten en leuningen een <r> vrije en vlakke doorgangsbreedte van minstens 150 cm </r> gegarandeerd wordt. </R>

<E> In de aanvraag kunnen afwijkingen op de ruwbouwmaten worden opgenomen als in het aanvraagdossier gemotiveerd aangetoond wordt dat na de afwerking van de wanden een <a> vrije en vlakke doorgangsbreedte, gemeten tussen de afgewerkte wanden met inbegrip van eventuele leuningen en plinten </a>, van <r> minstens 150 cm </r> gegarandeerd wordt. </E>
<E> In afwijking van lid 1 en 2 is een <a> versmalling van een dergelijk looppad </a> toegestaan in de volgende gevallen:
**1.** bij een versmalling die zich over <r> hoogstens 120 cm </r> uitstrekt: als de <a> breedte van het looppad, gemeten tussen de onafgewerkte binnenmuren </a>, <r> minstens 115 cm </r> bedraagt, zodat na afwerking van de wanden en met inbegrip van de ruimte voor plinten steeds een <r> vrije en vlakke doorgangsbreedte van minstens 90 cm </r> gegarandeerd wordt;
**2.** bij een versmalling die zich over <r> meer dan 120 cm </r> uitstrekt: als de <a> breedte van het looppad, gemeten tussen de onafgewerkte binnenmuren </a>, <r> minstens 145 cm </r> bedraagt, zodat na afwerking van de wanden en met inbegrip van de ruimte voor plinten een <r> vrije en vlakke doorgangsbreedte van minstens 120 cm </r> gegarandeerd wordt. In dit laatste geval moet <a> minstens elke tien meter </a>, alsook aan het <a> begin en het einde van de versmalling </a>, voor een <r> vrije en vlakke draairuimte </r> worden gezorgd. </E>

**Art. 16.**
<R> Een <a> looppad </a> mag alleen <r> hellen in de richting dwars op de normale looprichting om een normale afwatering te verzekeren </r>. </R>

**Art. 17.**
<R> Bij <a> handelingen waarbij brandhaspels, brandblusapparaten of andere uit de wand stekende constructies </a> geïnstalleerd worden, mogen die constructies het <r> ongestoord gebruik van het looppad niet in het gedrang brengen </r>. </R>
[Hiertoe kunnen die apparaten in een nis ingewerkt worden zodat ze niet buiten het afgewerkte muurvlak uitsteken.]

**Afdeling III. Bepalingen met betrekking tot niveauverschillen**
**Art. 18.**
<R> <a> Niveauverschillen </a> <r> tot en met 18 cm </r> moeten, zowel binnen als buiten, minstens met een <r> helling </r> overbrugd worden, <e> met uitzondering van niveauverschillen <r> tot twee cm </r> in <s> buitenruimtes </s> of niveauverschillen <r> tot twee cm </r> bij een <s> overgang tussen binnen- en buitenruimtes </s> </e>. </R>
<R> <a> Niveauverschillen </a> van <r> meer dan 18 cm </r> moeten overbrugd worden, ofwel met een <r> trap in combinatie met een helling </r>, ofwel met <r> een trap in combinatie met een lift </r>, ofwel met <r> een helling in combinatie met een lift </r>. </R>
<E> Ter uitvoering van artikel 33 en artikel 34, §1, kan in de aanvraag een afwijking van de verplichting tot het plaatsen van een <a> lift </a> worden opgenomen onder andere als in het <r> aanvraagdossier gemotiveerd aangetoond wordt dat de aanvraag over een gebouw, bestemd voor industrie en ambacht, gaat, dat een of meer ruimtes op de gelijkvloerse verdieping eenzelfde functie hebben als de ruimtes op een andere verdieping die door het ontbreken van de lift ontoegankelijk zijn, en dat de totale oppervlakte die door de afwijking ontoegankelijk blijft, beperkt is tot ten hoogste vijfentwintig procent van de totale publiek toegankelijke oppervlakte </r>. </E>
<E> Ter uitvoering van artikel 33 en artikel 34, §1, kan in de aanvraag ook een afwijking op de verplichting tot het plaatsen van een <a> lift </a> worden opgenomen als in het <r> aanvraagdossier aangetoond wordt dat de aanvraag over een gebouw bestemd voor toeristische verblijfsaccommodatie gaat en dat het gebouw na de handelingen twee of minder dan twee bouwlagen omvat en na de handelingen maximaal tien accommodaties beschikbaar zijn </r>. </E>

**Art. 19.**
**§1.** <R> Het <a> hellingspercentage </a> bedraagt <r> hoogstens </r>:
**1.** <r> tien procent </r> bij <a> niveauverschillen tot 10 cm </a>, <e> of in geval van <s> buitenruimtes </s> bij <a> niveauverschillen van 2 tot en met 10 cm </a> </e>;

**2.** <r> 8,3 procent </r> bij <a> niveauverschillen van 10 cm tot 25 cm </a>;

**3.** <r> 6,25 procent </r> bij <a> niveauverschillen van 25 cm tot 50 cm </a>;

**4.** <r> vijf procent </r> bij <a> niveauverschillen van 50 cm of groter </a>. </R>

**§2.** <R> Een <a> combinatie van hellingen </a> is toegestaan op voorwaarde dat gezorgd wordt voor een <r> tussenbordes van 120 cm op 150 cm <r> ter hoogte van de overgang. <E> Als een combinatie van hellingen gepaard gaat met een <s> verandering van richting </s>, is een <r> tussenbordes van 150 cm op 150 cm </r>, ter hoogte van de richtingsverandering, vereist. </E> </R>

**§3.** <R> Bij hellingen met een hellingspercentage van meer dan vier procent moet zowel bovenaan als onderaan voor een vrije en vlakke draairuimte worden gezorgd. Die draairuimte mag maximaal twee procent in één richting hellen met het oog op de afwatering. </R>

**§4.** <R> Bij een overbrugging van een <a> niveauverschil van meer dan 50 cm </a> of <a> een helling van meer dan 10 m met een hellingspercentage van meer dan vier procent </a>, moet voor een <r> tussenbordes van 120 cm op 150 cm </r> gezorgd worden. <E> Als de helling <s> verandert van richting </s>, is een <r> tussenbordes van 150 cm op 150 cm </r>, ter hoogte van de richtingsverandering, vereist. </E> </R>

**§5.** <R> De <a> breedte van een helling met een hellingspercentage van meer dan vier procent </a> bedraagt <r> minstens 145 cm </r>, zodat na de afwerking van de eventuele wanden en met inbegrip van de ruimte voor eventuele plinten en leuningen een <r> vrije en vlakke doorgangsbreedte van minstens 120 cm </r> gegarandeerd wordt. </R>

**§6.** <E> In de aanvraag kunnen afwijkingen worden opgenomen als in het aanvraagdossier gemotiveerd aangetoond wordt dat na de afwerking van de eventuele wanden een <r> vrije en vlakke doorgangsbreedte, gemeten tussen de afgewerkte wanden en met inbegrip van eventuele leuningen en plinten, van minstens 120 cm </r> gegarandeerd wordt. </E>

**§7.** <R> Als een <a> helling een niveauverschil van meer dan 10 cm overbrugt </a>, moet aan de <a> open zijkanten </a> van de helling en aan de eventuele <a> tussenbordessen </a> over de volledige lengte van de helling voor een <r> afrijdbeveiliging van minstens 5 cm </r> hoogte gezorgd worden. </R>

**§8.** (…)

**§9.** <R> Aan de zijkanten van een <a> helling die een niveauverschil van meer dan 25 cm overbrugt </a>, moet <r> aan beide zijden een leuning </r> aangebracht worden, die doorloopt over eventuele tussenbordessen. <a> Voor het begin en aan het einde van de helling </a> moet de leuning <r> minstens 40 cm horizontaal verderlopen </r>. Als de leuning in het ijle stopt, moet ze worden <r> afgerond naar de grond of naar de wand </r>. </R>

**Art. 20.**

**§1.** <R> Over de volledige lengte van de <a> trappen </a> en ter hoogte van eventuele <a> tussenbordessen </a> moet voor een <r> breedte van minstens 125 cm </r> gezorgd worden, telkens gemeten tussen de onafgewerkte binnenmuren, indien aanwezig, , zodat na afwerking van de eventuele wanden en tussen de leuningen, een <r> breedte van minstens 100 cm </r>, vrij van obstakels, gegarandeerd wordt, wat gebruikers toelaat zich zonder hinder langs de trap te verplaatsen. </R>

**§2.** <R> Na <r> ten hoogste 17 treden </r> moet voor een <r> tussenbordes van minstens 100 cm </r> diep gezorgd worden. </R>

**§3.** <R> Alle <a> treden </a> moeten over een zo <r> gelijkvormig mogelijke op- en aantrede </r> beschikken. </R>

[De aantrede is de horizontale afstand tussen twee opeenvolgende trapneuzen, gelegen op de looplijn van twee opeenvolgende traptreden. De optrede is de verticale afstand tussen de bovenzijde van twee opeenvolgende treden.]

<R> De <a> optrede </a> mag <r> hoogstens 18 cm </r> meten en de <a> aantrede </a> moet <r> minimaal 23 cm </r> meten. </R> <R> De <a> som van tweemaal de optrede en eenmaal de aantrede </a> van elke trede moet <r> tussen 57 cm en 63 cm bedragen of een veelvoud daarvan </r>. </R>

**§4.** <R> Aan beide zijden van de <a> trap </a> moet een <r> trapleuning </r> aangebracht worden, die doorloopt ter hoogte van eventuele tussenbordessen. <a> Voor het begin en aan het einde van de

trap </a> moet de trapleuning <r> minstens 40 cm horizontaal verderlopen </r>. Als de leuning in het ijle stopt, moet ze worden <r> afgerond naar de grond of naar de wand </r>. </R>
**§5.** <E> Bij gebouwen als vermeld in artikel 5, eerste en tweede lid, moet niet voldaan worden aan de bepalingen van dit artikel, als het gebouw over een lift beschikt, die voldoet aan de bepalingen van artikel 21. </E>

**Art. 21.**
**§1.** <R> Als een <a> lift </a> geïnstalleerd wordt, moet die in een <r> afgesloten koker </r> zitten of moet het om een <e> verticale plateaulift </e> gaan. <a> Liften in een afgesloten koker </a> moeten <r> minstens liften zijn van het type 2 </r> zoals omschreven in de EN 81-70. </R>
**§2.** <R> Voor een <a> lifttoegang </a> moet een <r> vrije en vlakke draairuimte </r> zijn. </R>
**§3.** <R> <a> Liften die in een afgesloten koker </a> geplaatst zijn, moeten <r> automatische deuren </r> hebben. De <a> vrije en vlakke doorgangsbreedte </a> van de liftdeur moet <r> minstens 90 cm </r> bedragen. </R>
**§4.** <R> Bij <a> verticale plateauliften </a> moet het <r> hefplateau minstens 100 cm breed en 140 cm diep </r> zijn. </R>
**§5.** <R> Over de volledige lengte van de <a> plateaulift </a>, alsook ter hoogte van de doorgangen van de deuren, moet een <r> vrije en vlakke doorgangsbreedte van minstens 90 cm </r> gegarandeerd worden. </R>

**Afdeling IV. Bepalingen met betrekking tot toegangen en deuropeningen**
**Art. 22.**
**§1.** <R> Voor <a> toegangen of deuropeningen </a> moet, na afwerking, een <r> vrije doorgangshoogte van minstens 2,09 meter </r> gegarandeerd worden. </R>
**§2.** <R> De <a> ruwbouwmaten van toegangen of deuropeningen </a> moeten <r> minstens 105 cm </r> breed zijn, zodat na afwerking een <r> vrije en vlakke doorgangsbreedte van minstens 90 cm </r> gegarandeerd wordt.
<E> Voor wat de <a> ruwbouwmaten betreft van toegangsdeuren als vermeld in artikel 5 alinea 1, 2 en 3 </a> kan een <r> minimale breedte van 100 cm </r> volstaan, mits na afwerking een <r> vrije en vlakke doorgangsbreedte van minstens 85 cm </r> gegarandeerd wordt. </E> </R>

**Art. 23.**
<R> Bij elke <a> draaideur </a>, moet er gezorgd worden voor een <r> alternatieve toegang of deur, die niet draait </r>. <E> Deze verplichting geldt niet bij <a> draaideuren die uitgerust zijn met mechanismen die het gebruik door personen met een handicap garanderen </a>. </E> </R>

**Art. 24.**
<R> <a> Vóór en achter elke toegang of deur </a> waarop dit besluit van toepassing is, met uitsluiting van <e> toegangen tot of deuren naar gesloten trappenhallen </e> , moet voor een <r> vrije en vlakke draairuimte </r> worden gezorgd, die <r> maximaal twee procent in één richting mag hellen </r> met het oog op de afwatering. </R>
[De vrije en vlakke draairuimtes van meerdere deuren mogen elkaar overlappen. Het draaivlak van een deur moet vlak zijn.]
<R> Bij <a> manueel te bedienen deuren </a> moet de <r> buitenste rand van de vrije en vlakke draairuimte aan de trekzijde van de deur, het draaivlak van de deur raken </r> en moet de <r> buitenste rand van de vrije en vlakke draairuimte aan de duwzijde van de deur, het gesloten deurvlak raken </r>. </R>

**Art. 25.**
<R> Bij een <a> manueel te bedienen deur </a>, met uitsluiting van <e> toegangen tot of deuren naar gesloten trappenhallen </e>, moet <r> naast de krukzijde voor een vrije en vlakke wand- en vloerbreedte </r> worden gezorgd, met een <r> ruwbouwmaat van minstens 45 cm </r>, zodat na de afwerking een <r> vrije en vlakke wand- en vloerbreedte van minstens 50 cm </r> gegarandeerd wordt. </R>

**Art. 26.**
<R> <a>Deuren die toegang verlenen tot aangepaste sanitaire voorzieningen, kleedruimtes of pashokjes </a>, moeten <r> naar buiten opendraaien </r>. </R>

**Afdeling V. Bepalingen met betrekking tot parkeerplaatsen**
**Art. 27.**
<R> Als een <a> constructie beschikt over één tot en met honderd eigen parkeerplaatsen </a>, moet <r> minstens zes procent van het totale aantal parkeerplaatsen </r> , en <r> minstens één parkeerplaats, een aangepaste parkeerplaats </r> zijn. Vanaf <a> vijf tot en met honderd eigen parkeerplaatsen </a>, moeten de aangepaste parkeerplaatsen ook <r> voorbehouden parkeerplaatsen </r> zijn. </R>
<R> Als een <a> constructie beschikt over meer dan honderd eigen parkeerplaatsen </a>, moet bovendien <s> per extra schijf van vijftig parkeerplaatsen </s>, telkens <r> één parkeerplaats een aangepaste en voorbehouden parkeerplaats </r> zijn. </R>
<E> Als het totale aantal nieuw aan te leggen parkeerplaatsen minder bedraagt dan zes procent van het totale aantal parkeerplaatsen, beperkt de verplichting van het eerste en het tweede lid zich tot de nieuw aan te leggen parkeerplaatsen. </E>
<R> Een <a> aangepaste parkeerplaats </a> voldoet aan de volgende normen:
**1.** ze bevindt zich <r> zo dicht mogelijk bij de toegankelijke ingang van de constructie of bij de voetgangersuitgang van de parkeervoorziening </r>;
**2.** bij <s> dwarsparkeren en schuinparkeren </s> bedraagt de <r> breedte van de aangepaste parkeerplaats minstens 350 cm </r> en bij <a> langsparkeren </a> bedraagt de <r> lengte van de aangepaste parkeerplaats minstens 600 cm </r>;
**3.** <r> het oppervlak van de aangepaste parkeerplaats helt niet meer dan twee procent </r>. </R>
[Om als voorbehouden parkeerplaats in aanmerking te komen, moet een parkeerplaats aan de bepalingen van de derde alinea voldoen en voorbehouden worden voor personen met een handicap conform het koninklijk besluit houdende algemeen reglement op de politie van het wegverkeer en van het gebruik van de openbare weg van 1 december 1975, en aangegeven volgens de bepalingen van het voormelde koninklijk besluit en het ministerieel besluit van 11 oktober 1976 waarbij de minimumafmetingen en de bijzondere plaatsingsvoorwaarden van de verkeerstekens worden bepaald.]

**Afdeling VI. Bepalingen met betrekking tot vaste inrichtingselementen**
**Art. 28.**
<R> Als een <a> vast inrichtingselement met het oog op het onthaal van het publiek </a> of een daarmee gelijkgestelde constructie wordt aangebracht, moet <r> vóór dat element voor een vrije en vlakke draairuimte </r> worden gezorgd. </R>
<R> Aan <a> alle elementen als vermeld in het eerste lid </a> moet een <r> verlaagd gedeelte </r> aangebracht worden. De <r> hoogte tot de bovenzijde van het verlaagde gedeelte bedraagt hoogstens 80 cm </r>. <r> Onder het verlaagde gedeelte moet een opening zijn van minstens 70 cm hoog, minstens 90 cm breed en minstens 60 cm diep </r>. </R>

**Art. 29.**
<R> Bij <a> handelingen aan binnen- of buitenruimtes met vaste inrichtingselementen die dienst doen als zitplaatsen voor toeschouwers of toehoorders </a>, moeten <r> minstens twee vrije ruimtes gereserveerd worden voor personen met een handicap <r> in <s> elke ruimte met minder dan vijftig zitplaatsen waar een voorstelling wordt aangeboden </s>. Elk van die vrije ruimtes moet <r> minstens 90 cm breed en minstens 140 cm diep </r> zijn en moet zich bevinden op een <r> vloer zonder niveauverschillen of hellingen </r>. Op het toegangspad naar die vrije ruimtes en eraan grenzend moet in een <r> vrije en vlakke draairuimte </r> voorzien worden. </R>
<R> <a>Vanaf vijftig zitplaatsen en voor elke extra groep van vijftig zitplaatsen </a> moet bijkomend voor <r> minstens één extra vrije ruimte </r> als vermeld in het eerste lid gezorgd worden. </R>
<E> Als het totale aantal nieuw te bouwen, te herbouwen, te verbouwen of uit te breiden zitplaatsen minder bedraagt dan de te reserveren vrije ruimtes voor personen met een handicap als vermeld in het eerste en tweede lid, beperkt die verplichting zich tot de nieuw te bouwen, te herbouwen, te verbouwen of uit te breiden zitplaatsen. </E>

**Afdeling VII. Bepalingen met betrekking tot het aangepast karakter van constructies of delen van constructies**

**Art. 29/1.**

<R> Bij <a> handelingen aan publiek toegankelijke kleedruimtes of pashokjes </a> moet <r> minstens vier procent van het totale aantal kleedruimtes of pashokjes na de handelingen aan de bepalingen van artikel 12 en van artikel 22 tot en met 26 voldoen </r>. Ongeacht het totale aantal kleedruimtes of pashokjes na de handelingen moet <r> minstens één kleedruimte of pashokje aan de bepalingen van artikel 12 en van artikel 22 tot en met 26 voldoen </r>. </R>

<R> Bij <a> aparte kleedruimtes of pashokjes </a>, die alleen voor vrouwen of alleen voor mannen bestemd zijn, moet telkens <r> minstens één kleedruimte of pashokje in elke ruimte voldoen aan de bepalingen van artikel 12 en artikel 22 tot en met 26 </r>, <e> tenzij de aangepaste kleedruimte of het aangepaste pashokje, bestemd voor zowel vrouwen als mannen, zich in een zone bevindt die niet gereserveerd is voor mannen dan wel vrouwen </e>. </R>

<E> Als het totale aantal nieuw te bouwen, te herbouwen, te verbouwen of uit te breiden kleedruimtes of pashokjes minder bedraagt dan vier procent van het totale aantal kleedruimtes of pashokjes, beperkt de verplichting van paragraaf 1 zich tot de nieuw te bouwen, te herbouwen, te verbouwen of uit te breiden kleedruimtes of pashokjes. </E>

**Art. 29/2.**

<R> Bij <a> handelingen aan publiek toegankelijke toiletten </a> moet <r> in elk sanitair blok minstens één toilet voldoen aan de bepalingen van artikel 12, 30, eerste lid en artikel 31, inzonderheid 1° en 2° </r>. </R>

<R> Bij <a> handelingen aan publiek toegankelijke doucheruimtes </a>, moet <r> in elk sanitair blok minstens één douche voldoen aan de bepalingen van artikel 12, 30, tweede en derde alinea, artikel 31, inzonderheid 1° en 3° en artikel 31/1 </r>. </R>

<R> Bij <a> aparte toiletten of doucheruimtes </a>, die alleen voor vrouwen of alleen voor mannen bestemd zijn, moet telkens <r> minstens één toilet of doucheruimte in elke zone voldoen aan de bepalingen van artikel 12 en artikel 30 tot en met 31/1 </r>, <e> tenzij het aangepast toilet of de aangepaste doucheruimte, bestemd voor zowel vrouwen als mannen, zich in een zone bevindt die niet gereserveerd is voor mannen dan wel vrouwen </e>. </R>

**Art. 30.**

<R> De <a> ruwbouwmaten van een aangepast toilet </a> moeten <r> minstens 1,70 meter op 2,25 meter </r> zijn, zodat na de afwerking van de wanden en met inbegrip van de ruimte voor plinten een <r> ruimte van minstens 1,65 meter op 2,20 meter </r> gegarandeerd wordt. Bij die minimale maten moet de deur in de korte zijde aangebracht worden. </R>

<R> De <a> ruwbouwmaten van een aangepaste doucheruimte </a>, al dan niet met wastafel, moeten <r> minstens 2,20 meter op 2,40 meter </r> zijn, zodat na de afwerking van de wanden en met inbegrip van de ruimte voor plinten een <r> ruimte van minstens 2,15 meter op 2,35 meter </r> gegarandeerd wordt. Bij die minimale maten moet de deur in de korte zijde aangebracht worden. </R>

<R> De <a> ruwbouwmaten van een aangepaste sanitaire voorziening met douche en toilet </a>, al dan niet met wastafel, moeten <r> minstens 2,40 meter op 2,45 meter </r> zijn, zodat na de afwerking van de wanden en met inbegrip van de ruimte voor plinten een <r> ruimte van minstens 2,35 meter op 2,40 meter </r> gegarandeerd wordt. Bij die minimale maten moet de deur in de korte zijde aangebracht worden. </R>

**Art. 31.**

<E> In de aanvraag kunnen <a> afwijkingen van de ruwbouwmaten </a>, vermeld in artikel 30, worden opgenomen als in het aanvraagdossier gemotiveerd aangetoond wordt dat na de afwerking van de sanitaire ruimte aan de volgende voorwaarden is voldaan:

**1.** <R> <a>ter hoogte van de wastafel, de toiletpot en de douchezone </a> is een <r> vrije en vlakke draairuimte </r>. De ruimte onder de aangepaste wastafel mag meegerekend worden voor de bepaling van de vrije en vlakke draairuimte; </R>

**2.** <R> in een <a> aangepast toilet </a>:

**a.** moet <a> voor de toiletpot </a> en na de afwerking en inrichting van de ruimte een <r> vrije afstand van minstens 120 cm </r> gegarandeerd zijn;

**b.** moet <a> minstens aan één zijde van de toiletpot </a> een <r> vrije transferzone van minstens 90 cm </r> zijn;

**c.** moet de <a> vrije doorgang tussen de toiletpot en de wastafel </a> <r> minstens 90 cm </r> breed zijn;

**d.** moet <a> de afstand van de voorzijde van de toiletpot tot tegen de achterliggende wand </a> <r> minstens 70 cm </r> bedragen;

**e.** moet <a> een wastafel </a> aangebracht zijn <r> waaronder een ruimte is van minstens 70 cm hoog, minstens 90 cm breed en minstens 60 cm diep </r>. <e> Als de wastafel in een inwendige hoek is geplaatst, moet <a> de afstand tussen de as van de wastafel en de inwendige hoek </a> <r> minstens 50 cm </r> bedragen </e>; </R>

**3.** <R> in een <a> aangepaste doucheruimte </a>:

**a.** moet <a> de vloer van de douchezone </a> <r> drempelloos aansluiten op de vloer van de doucheruimte </r> ;

**b.** mag <a> de vloer van de douchezone </a> <r> hoogstens twee procent hellen </r>;

**c.** moet <a> het vloeroppervlak van de douchezone </a> na de afwerking van de wanden <r> minstens 120 cm op 120 cm </r> bedragen;

**d.** moet <a> een douchezitje </a> van <r> minstens 45 cm diep en 40 cm breed </r> aanwezig zijn. <e> Als het douchezitje in een inwendige hoek is geplaatst, moet de <a> afstand tussen de as van het douchezitje en de inwendige hoek </a> <r> minstens 45 cm </r> bedragen </e>;
moet aan <a> minstens één zijde van het douchezitje </a> een <r> vrije transferzone van minstens 90 cm </r> zijn;

**e.** moet aan <a> de voorzijde van het douchezitje </a> een <r> vrije ruimte van minstens 120 cm </r> zijn;

**f.** moet de <a> douchekraan </a> aangebracht worden op een <r> afstand tussen 45 en 55 cm van de wand waartegen het douchezitje geplaatst is </r>. </R> </E>

**Art. 31/1.**

<R> Een <a> doucheruimte </a> is pas aangepast als het een <r> vlakke drempelloze ruimte </r> is en voorzien is van een <r> vrije en vlakke draairuimte </r>. </R>

<R> De <a> douchezone </a> moet bovendien een <r> slipvrij oppervlak </r> hebben, dat <r> gelijkloopt met de vloer </r> en dat <r> maximaal twee procent helt </r>. </R>

**Art. 32.**

<R> Het <a> toilet, de wastafel en de doucheruimte </a> die ter beschikking staan van de gebruiker van een aangepaste accommodatie, als vermeld in artikel 4, moeten aan <r> de bepalingen van dit hoofdstuk </r> voldoen. </R>

<R> Als de aangepaste accommodatie een <a> vakantiewoning </a> betreft, moet vlak bij de ingang van de vakantiewoning in een <r> aangepaste parkeerplaats </r> worden voorzien, overeenkomstig artikel 27. </R>

# Appendix B: rulebook.ttl

```
@prefix bot: <https://w3id.org/bot#> .
@prefix inst:  <http://example.org/> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix props: <https://w3id.org/props#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix schema:  <http://schema.org/> .
@prefix sh: <http://www.w3.org/ns/shacl#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

// Model View Definition to ensure all the necessary data is present
schema:SpaceMVD
  a sh:NodeShape ;
  sh:targetClass bot:Space ;
  sh:property [
    sh:path props:heightIfcSpace ;
    sh:minCount 1 ;
    sh:maxCount 1 ;
    sh:message "MVD - each space should have exactly one 'heightIfcSpace'." ;
  ] ;
  sh:property [
    sh:path props:widthIfcSpace ;
    sh:minCount 1 ;
    sh:maxCount 1 ;
    sh:message "MVD - each space should have exactly one 'widthIfcSpace'." ;
  ] .

schema:RampFlightMVD
  a sh:NodeShape ;
  sh:targetClass <https://pi.pauwel.be/voc/buildingelement#RampFlight> ;
  sh:property [
    sh:path props:widthIfcRampFlight ;
    sh:minCount 1 ;
    sh:maxCount 1 ;
    sh:message "MVD - each rampflight should have exactly one 'widthIfcRampFlight'." ;
  ] .

schema:SlabMVD
  a sh:NodeShape ;
  sh:targetClass <https://pi.pauwel.be/voc/buildingelement#Slab-LANDING> ;
  sh:property [
    sh:path props:widthIfcSlab ;
    sh:minCount 1 ;
    sh:maxCount 1 ;
    sh:message "MVD - each slab should have exactly one 'widthIfcSlab'." ;
  ] ;
  sh:property [
    sh:path props:depthIfcSlab ;
    sh:minCount 1 ;
    sh:maxCount 1 ;
    sh:message "MVD - each slab should have exactly one 'depthIfcSlab'." ;
  ] .

schema:StairFlightMVD
  a sh:NodeShape ;
  sh:targetClass <https://pi.pauwel.be/voc/buildingelement#StairFlight> ;
```

```
57      sh:property [
58        sh:path props:numberOfRiserIfcStairFlight ;
59        sh:minCount 1 ;
60        sh:maxCount 1 ;
61        sh:message "MVD - each stairflight should have exactly one
62        'numberOfRiserIfcStairFlight'." ;
63      ] .
64
65    schema:StairMVD
66      a sh:NodeShape ;
67      sh:targetClass <https://pi.pauwel.be/voc/buildingelement#Stair> ;
68      sh:property [
69        sh:path props:width ;
70        sh:minCount 1 ;
71        sh:maxCount 1 ;
72        sh:message "MVD - each stair should have exactly one 'width'." ;
73      ] .
74
75    schema:DoorMVD
76      a sh:NodeShape ;
77      sh:targetClass <https://pi.pauwel.be/voc/buildingelement#Door> ;
78      sh:property [
79        sh:path props:overallHeightIfcDoor ;
80        sh:minCount 1 ;
81        sh:maxCount 1 ;
82        sh:message "MVD - each door should have exactly one 'overallHeightIfcDoor'." ;
83      ] ;
84      sh:property [
85        sh:path props:overallWidthIfcDoor ;
86        sh:minCount 1 ;
87        sh:maxCount 1 ;
88        sh:message "MVD - each door should have exactly one 'overallWidthIfcDoor'." ;
89      ] .
90
91    //declaration of functions
92    inst:lessThan
93      a sh:SPARQLFunction ;
94      rdfs:comment "Returns True if op1 < op2." ;
95      sh:parameter [
96        sh:path inst:op1 ;
97        sh:datatype xsd:double ;
98        sh:description "The first int" ;
99      ] ;
100     sh:parameter [
101       sh:path inst:op2 ;
102       sh:datatype xsd:double ;
103       sh:description "The second int" ;
104     ] ;
105     sh:returnType xsd:boolean ;
106     sh:select """
107       SELECT ?result
108       WHERE {
109         BIND(IF(?op1 < ?op2, true, false) AS ?result) .
110       }
111       """ .
112
113   inst:stairFormula
114     a sh:SPARQLFunction ;
115     rdfs:comment "stair formula" ;
116     sh:parameter [
117       sh:path inst:op1 ;
```

```
      sh:datatype xsd:double ;
      sh:description "riserHeight" ;
    ] ;
    sh:parameter [
      sh:path inst:op2 ;
      sh:datatype xsd:double ;
      sh:description "treadLength" ;
    ] ;
    sh:returnType xsd:double ;
    sh:select """
      SELECT ( (2 * $op1 + $op2) AS ?result)
      WHERE {
      }
      """ .

inst:slope
  a sh:SPARQLFunction ;
  rdfs:comment "slope of a rampflight" ;
  sh:parameter [
    sh:path inst:op1 ;
    sh:datatype xsd:double ;
    sh:description "height rampflight" ;
  ] ;
  sh:parameter [
    sh:path inst:op2 ;
    sh:datatype xsd:double ;
    sh:description "length rampflight" ;
  ] ;
  sh:returnType xsd:double ;
  sh:select """
    SELECT ( ( ($op1 / $op2) * 100) AS ?result)
    WHERE {
    }
    """ .

// constraint definitions
schema:Space
  a sh:NodeShape ;
  sh:targetClass bot:Space ;
  sh:property [
    sh:path props:heightIfcSpace ;
    sh:property schema:SpaceHeight ;
  ] ;
  sh:property [
    sh:path props:widthIfcSpace ;
    sh:property schema:SpaceWidth ;
  ] .

schema:SpaceHeight
  a sh:PropertyShape ;
  sh:path schema:value ;
  sh:minInclusive "2300"^^xsd:double ;
  sh:message "Art. 12 - The clearance height must be at least 2300 mm." .

schema:SpaceWidth
  a sh:PropertyShape ;
  sh:path schema:value ;
  sh:minInclusive "1500"^^xsd:double ;
  sh:message "Art. 14 - The width of the walkway must be at least 1500 mm." .

schema:Slopes;
```

```
    a sh:NodeShape ;
    sh:targetClass <https://pi.pauwel.be/voc/buildingelement#RampFlight> ;
    sh:message "Art. 19 - The slope of a rampflight can be at most 10% when the difference in
    height is less than 100 mm. The slope of a rampflight can be at most 8.3% when the
    difference in height lies between 100 and 250 mm. The slope of a rampflight can be at
    most 6.25% when the difference in height lies between 250 and 500 mm. The slope of a
    rampflight can be at most 5% when the difference in height is more than 500 mm." ;
    sh:or (
      [
        sh:and (
          [
            sh:expression [
              inst:lessThan (
                [
                  inst:slope (
                    [ sh:path (props:heightIfcRampFlight schema:value) ]
                    [ sh:path (props:lengthIfcRampFlight schema:value) ]
                  )
                ]
                10
              )
            ]
          ]
          sh:node schema:SlopeConstraint10
        )
      ]
      [
        sh:and (
          [
            sh:expression [
              inst:lessThan (
                [
                  inst:slope (
                    [ sh:path (props:heightIfcRampFlight schema:value) ]
                    [ sh:path (props:lengthIfcRampFlight schema:value) ]
                  )
                ]
                8.3
              )
            ]
          ]
          sh:node schema:SlopeConstraint8
        )
      ]
      [
        sh:and (
          [
            sh:expression [
              inst:lessThan (
                [
                  inst:slope (
                    [ sh:path (props:heightIfcRampFlight schema:value) ]
                    [ sh:path (props:lengthIfcRampFlight schema:value) ]
                  )
                ]
                6.25
              )
            ]
          ]
          sh:node schema:SlopeConstraint6
        )
```

```
      ]
      [
        sh:and (
          [
            sh:expression [
              inst:lessThan (
                [
                  inst:slope (
                    [ sh:path (props:heightIfcRampFlight schema:value) ]
                    [ sh:path (props:lengthIfcRampFlight schema:value) ]
                  )
                ]
                5
              )
            ]
          ]
          sh:node schema:SlopeConstraint5
        )
      ]
    ) .

  schema:SlopeConstraint10
    a sh:NodeShape ;
    sh:property [
      sh:path (props:heightIfcRampFlight schema:value) ;
      sh:maxInclusive "100"^^xsd:double ;
    ] .

  schema:SlopeConstraint8
    a sh:NodeShape ;
    sh:property [
      sh:path (props:heightIfcRampFlight schema:value) ;
      sh:minInclusive "100"^^xsd:double ;
      sh:maxInclusive "250"^^xsd:double ;
    ] .

  schema:SlopeConstraint6
    a sh:NodeShape ;
    sh:property [
      sh:path (props:heightIfcRampFlight schema:value) ;
      sh:minInclusive "250"^^xsd:double ;
      sh:maxInclusive "500"^^xsd:double ;
    ] .

  schema:SlopeConstraint5
    a sh:NodeShape ;
    sh:property [
      sh:path (props:heightIfcRampFlight schema:value) ;
      sh:minInclusive "500"^^xsd:double ;
    ] .

  schema:RampLanding
    a sh:NodeShape ;
    sh:targetClass <https://pi.pauwel.be/voc/buildingelement#Slab-LANDING> ;
    sh:property [
      sh:path props:widthIfcSlab ;
      sh:property schema:RampLandingWidth ;
    ] ;
    sh:property [
      sh:path props:depthIfcSlab ;
      sh:property schema:RampLandingDepth ;
```

```
  ] .

schema:RampLandingWidth
  a sh:PropertyShape ;
  sh:path schema:value ;
  sh:minInclusive "1200"^^xsd:double ;
  sh:message "Art. 19 - The width of a landing of a ramp must be at least 1200 mm." .

schema:RampLandingDepth
  a sh:PropertyShape ;
  sh:path schema:value ;
  sh:minInclusive "1500"^^xsd:double ;
  sh:message "Art. 19 - The depth of a landing of a ramp must be at least 1500 mm." .

schema:RampFlight
  a sh:NodeShape ;
  sh:targetClass <https://pi.pauwel.be/voc/buildingelement#RampFlight> ;
  sh:property [
    sh:path props:heightIfcRampFlight ;
    sh:property schema:RampFlightLanding ;
  ] .

schema:RampFlightLanding
  a sh:PropertyShape ;
  sh:path schema:value ;
  sh:maxInclusive "500"^^xsd:double ;
  sh:message "Art. 19 - A ramp must have an intermediate landing if the difference in
  height is more than 500 mm." .

schema:RampFlightWidth
  a sh:NodeShape ;
  sh:targetClass <https://pi.pauwel.be/voc/buildingelement#RampFlight> ;
  sh:message "Art. 19 - The width of a rampflight must be at least 1200 mm, if the slope is
  more than 4%." ;
  sh:or (
    [
      sh:and (
        [
          sh:expression [
            inst:lessThan (
              4
              [
                inst:slope (
                  [ sh:path (props:heightIfcRampFlight schema:value) ]
                  [ sh:path (props:lengthIfcRampFlight schema:value) ]
                )
              ]
            )
          ]
        ]
        [
          sh:node schema:RampFlightWidthConstraint
        ]
      )
    ]
    [
      sh:expression [
        inst:lessThan (
          [
            inst:slope (
              [ sh:path (props:heightIfcRampFlight schema:value) ]
```

```
                [ sh:path (props:lengthIfcRampFlight schema:value) ]
              )
            ]
            4
          )
        ]
      ]
    ) .

  schema:RampFlightWidthConstraint
    a sh:NodeShape ;
    sh:property [
      sh:path (props:widthIfcRampFlight schema:value) ;
      sh:minInclusive "1200"^^xsd:double ;
    ] .

  schema:StairFlight
    a sh:NodeShape ;
    sh:targetClass <https://pi.pauwel.be/voc/buildingelement#StairFlight> ;
    sh:property [
      sh:path props:numberOfRiserIfcStairFlight ;
      sh:property schema:StairFlightLanding ;
    ] ;
    sh:property [
      sh:path props:riserHeightIfcStairFlight ;
      sh:property schema:StairFlightRiser ;
    ] ;
    sh:property [
      sh:path props:treadLengthIfcStairFlight ;
      sh:property schema:StairFlightTread ;
    ] .

  schema:StairFlightLanding
    a sh:PropertyShape ;
    sh:path schema:value ;
    sh:maxInclusive "17"^^xsd:double ;
    sh:message "Art. 20 - A staircase must have an intermediate landing after at most 17
    treads." .

  schema:StairFlightRiser
    a sh:PropertyShape ;
    sh:path schema:value ;
    sh:maxInclusive "180"^^xsd:double ;
    sh:message "Art. 20 - The riser height must be at most 180 mm." .

  schema:StairFlightTread
    a sh:PropertyShape ;
    sh:path schema:value ;
    sh:minInclusive "230"^^xsd:double ;
    sh:message "Art. 20 - The tread length must be at least 230 mm." .

  schema:TooEasy
    a sh:NodeShape ;
    sh:targetClass <https://pi.pauwel.be/voc/buildingelement#StairFlight> ;
    sh:expression [
      sh:message "Art. 20 - This stair is too easy." ;
      inst:lessThan (
        570
        [
          inst:stairFormula (
            [ sh:path (props:riserHeightIfcStairFlight schema:value) ]
```

```
            [ sh:path (props:treadLengthIfcStairFlight schema:value) ]
          )
        ]
      )
    ] .

  schema:TooSteep
    a sh:NodeShape ;
    sh:targetClass <https://pi.pauwel.be/voc/buildingelement#StairFlight> ;
    sh:expression [
      sh:message "Art. 20 - This stair is too steep." ;
      inst:lessThan (
        [
          inst:stairFormula (
            [ sh:path (props:riserHeightIfcStairFlight schema:value) ]
            [ sh:path (props:treadLengthIfcStairFlight schema:value) ]
          )
        ]
        630
      )
    ] .

  schema:Stair
    a sh:NodeShape ;
    sh:targetClass <https://pi.pauwel.be/voc/buildingelement#Stair> ;
    sh:property [
      sh:path bot:hasSubElement ;
      sh:qualifiedValueShape [ sh:class <https://pi.pauwel.be/voc/buildingelement#Railing> ];
      sh:qualifiedMinCount 2 ;
      sh:message "Art. 20 - All stairs must have a railing on both sides." ;
    ] ;
    sh:property [
      sh:path props:width ;
      sh:property schema:StairWidth ;
    ].

  schema:StairWidth
    a sh:PropertyShape ;
    sh:path schema:value ;
    sh:minInclusive "1000"^^xsd:double ;
    sh:message "Art. 20 - The width of a staircase must be at least 1000 mm." .

  schema:Door
    a sh:NodeShape ;
    sh:targetClass <https://pi.pauwel.be/voc/buildingelement#Door> ;
    sh:property [
      sh:path props:overallHeightIfcDoor ;
      sh:property schema:DoorHeight ;
    ] ;
    sh:property [
      sh:path props:overallWidthIfcDoor ;
      sh:property schema:DoorWidth ;
    ] .

  schema:DoorHeight
    a sh:PropertyShape ;
    sh:path schema:value ;
    sh:minInclusive "2090"^^xsd:double ;
    sh:message "Art. 22 - The door height must be at least 2090 mm." .

  schema:DoorWidth
```

```
484     a sh:PropertyShape ;
485     sh:path schema:value ;
486     sh:minInclusive "900"^^xsd:double ;
487     sh:message "Art. 22 - The door width must at least 900 mm." .
```

FACULTY OF ENGINEERING AND ARCHITECTURE

# Automated validation of building models against legislation using Linked Data

Emma Nuyts
Student number: 01704320

Supervisor: Prof. ir.-arch. Paulus Present
Counsellor: Ir.-arch. Jeroen Werbrouck

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

Academic year 2021-2022

GHENT UNIVERSITY