MSc thesis in Geomatics

Automatic building permits checks by means of 3D city models

Jialun Wu

2021

TU Delft

MSc thesis in Geomatics

Automatic building permits checks by means of 3D city models

Jialun Wu

September 2021

A thesis submitted to the Delft University of Technology in partial fulfillment of the requirements for the degree of Master of Science in Geomatics

The work in this thesis was carried out in the:

3D geoinformation group
Delft University of Technology

Supervisors:	Dr. Francesca Noardo
	Dr. Ken Arroyo Ohori
Co-reader:	Prof.dr. J.E. Stoter

Abstract

Most construction work requires building permit checks. Therefore, to promote sustainable and high-quality urban development, it is necessary to improve the building permit application process and accelerate the digital process of building permit checks.

Many studies about the digitalization of building permitting use the Building Information Modeling (BIM), mainly coded in its reference open standard Industry Foundation Classes (IFC) by A worldwide industry body driving the digital transformation of the built asset industry (buildingSMART), to check compliance. However, many regulations need to consider some building context that is not supported by the current buildingSMART. This will affect the permit checks of these building contexts. The use of 3D city models for building permit checks is an alternative to BIM. 3D city models are powerful tools for city-level analysis and they can do extensions on different City Objects which able to cover the building context in the regulation, therefore it has good applicability to problems related to urban planning.

At present, there are little researches on building permit inspection based on 3D city models, so this study aims to provide an approach to use 3D city models and its extensions to support digital building permit checks. CityJSON has advantages over the CityGML data format in the development process, so it was selected as the 3D city model used in this study.

The first step is to choose the regulations for building permit checks. This involves the interpretation of the regulations, formalization process. In this study, the use case research method was selected, and the parking standards for cars and bicycles in Rotterdam were selected as the research object. We built a logic-based mechanism to map the content of the law to the corresponding 3D model. Next, we built the CityJSON conceptual model, which included four types of urban objects and expanded them. Then, the data we obtained and the data model will be stored and extended in a CityJSON model and then used for calculation and analysis in the developed tool.

Finally, through the tools developed, we successfully carried out a parking legal building permit inspection on the new building test data and stored and visualized the results. By using the framework and tool developed in this research, we can replicated the process to different places and regulations.

Acknowledgements

I would like to express my heartfelt thanks to those who have been supporting and encouraging the development of this thesis.

First of all, my sincere gratitude goes to my mentors Francesca Noardo and Ken Arroyo Ohori. They gave me a lot of guidance and support during my thesis period, and they will patiently and kindly give me advice when I encounter difficulties. Although we are unable to communicate with each other face to face due to the epidemic, the weekly meetings and social apps have brought us closer. They are not only my mentor but also my good friend. I'm happy to meet them and have precious memories. I still remember meeting Francesca at the faculty last September. We discussed the content of the thesis together and happily took a selfie and hugged. I hope we will meet again in the future.

I would like to thank my co-reader Jantien Stoter and the delegate Huib Plomp, for their valuable comments in the phased meetings.

In addition, I would like to thank my friends, classmates of Geomatics, friends in China, they will listen to my troubles and help me. Special thanks to my boyfriend, his company and care give me a lot of strength. Thank him for always making good dishes for me.

Finally, I would like to thank my parents and family. Their unreserved support and love for me is the driving force for me to study abroad in the Netherland.

Contents

List of Figures

List of Tables

Acronyms

BIM	Building Information Modeling	v
GIS	geographical information system	26
GUI	graphical user interface	14
IFC	Industry Foundation Classes	v
buildingSMART	A worldwide industry body driving the digital transformation of the built asset industry	v
EUnet4DBP	European Network for Digital Building Permits	2
BAG	Basisregistratie Adressen en Gebouwen	4
OSM	OpenStreetMap	4
AEC	architecture, engineering, and construction	5
CAD	computer-aided design	6
BERA	Building Environment Rule and Analysis	6
VCCL	Visual Code Checking Language	6
ADE	Application domain extension	10
API	Application Programming Interface	10
LCC	Linear Cell Complex	11
LOD	Levels of Detail	21
QGIS	Quantum GIS	25
CLI	Command-Line Interface	26
GFA	gross floor area	27

1 Introduction

1.1 Background and motivation

Most construction work requires a building permit. This includes constructing, modifying, or demolishing buildings or changing their use [Eirinaki et al., 2018]. In the Netherlands, municipalities issue building permits, supervise construction work and inspect permit applications for new development projects under the Building Decree [Decree et al., 2012]. In actual situations, in the face of more complex construction work, permit applicants may need to spend a lot of time studying relevant laws and regulations, application procedures, and required documents. This process is complicated and inefficient [May, 2005]. For the government, the review of building permits requires the cooperation of multiple departments. The slow work process will cause unreasonable waste of resources, which increases the risk of inadequate building supervision and ultimately affects public safety [Agyeman et al., 2016]. Therefore, to promote the sustainable and high-quality development of the city, it is necessary to improve the process of building permit checks.

The traditional way for doing building permit checks is based on the manual work of experts to review building permits of documents and 2D plans presented in paper or PDF formats. This method consumes a lot of manpower and material resources and cannot efficiently obtain results.

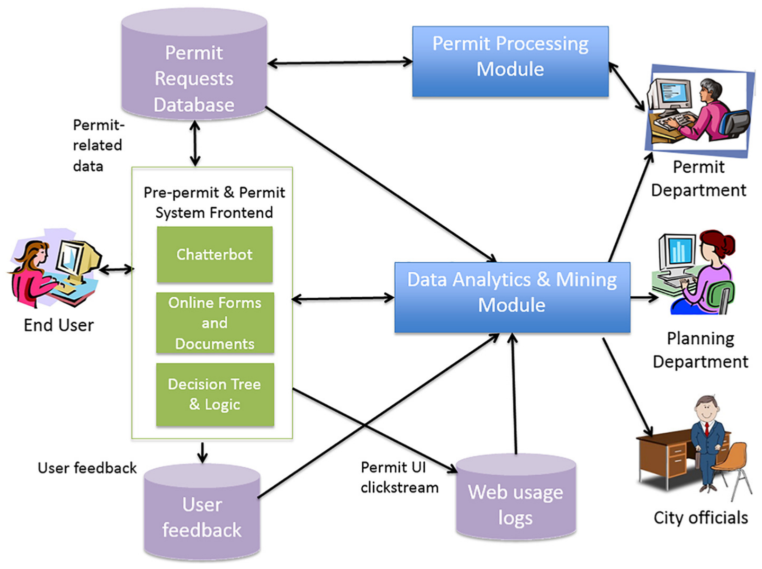
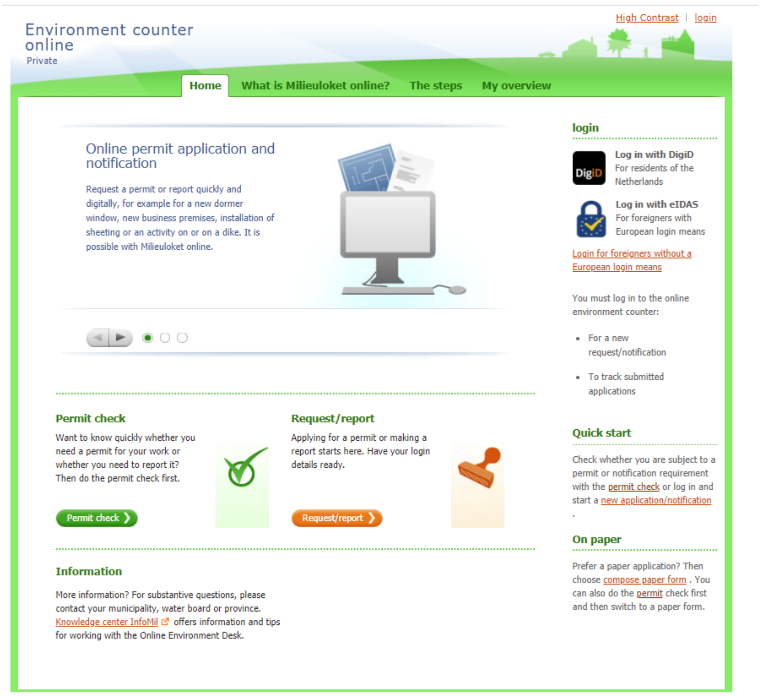


Figure 1.1: A Pre-Permit and Permit System Eirinaki et al. [2018]

With the development of computer technology, automated process based on computer technology with models has become a trend. The digital building permit is an application of 3D city data management. Currently, some cities provide online permitting platform for users to submit applications and inquiries online. For example, the platform of online application and notification of The Hague in the Netherlands which shows in figure 1.2. On this platform, you can quickly apply for permits or reports digitally, and you can also inquire whether permits are required for related construction work. Digitizing the building permit inspection process can improve work efficiency and provide more objective results. In figure 1.1, it shows a Pre-Permit and Permit System which simplifies the process required for building permits to benefit the end-user. With the advancement of technology and human cognition, regulations are also being updated at the same time. For a new building in some cases, in order to obtain a building permit, it must be verified whether it meets the new standards as expressed by regulations. However, different standards in various places and different interpretations of the regulations make the automated checking of building permits a very challenging issue [Eastman et al., 2009].

In the framework of automating the inspection of building permits, the ambitions/specification of European Network for Digital Building Permits (EUnet4DBP) aims to accelerate the digital transformation of the building permit process [EUnet4DBP, 2020]. Its vision is also the direction of the digital transformation of building permit inspection. One of the ambitions is to make regulation simple and machine-readable to achieve standardization of interpretation and eliminate unnecessary ambiguity [Barberis et al., 2019]. Machine-readable data is data that can be read and interpreted by a computer automatically. It must be structured data [Open Knowledge Foundation, 2019]. In an increasingly digitalized world, the storage and interpretation of laws and regulations are still based on human language. Because there is no unified standard, different people can have different interpretations of the same laws and regulations. This makes the administration and application of regulations complex and ambiguous.


Environment counter online
Private

High Contrast | login

Home | What is Milieuloket online? | The steps | My overview

Online permit application and notification

Request a permit or report quickly and digitally, for example for a new dormer window, new business premises, installation of sheeting or an activity on or on a dike. It is possible with Milieuloket online.

Permit check

Want to know quickly whether you need a permit for your work or whether you need to report it? Then do the permit check first.

Permit check ❯

Request/report

Applying for a permit or making a report starts here. Have your login details ready.

Request/report ❯

Information

More information? For substantive questions, please contact your municipality, water board or province. Knowledge center InfoMil offers information and tips for working with the Online Environment Desk.

login

Log in with DigiD
For residents of the Netherlands

Log in with eIDAS
For foreigners with European login means

Login for foreigners without a European login means

You must log in to the online environment counter:

- For a new request/notification
- To track submitted applications

Quick start

Check whether you are subject to a permit or notification requirement with the permit check or log in and start a new application/notification .

On paper

Prefer a paper application? Then choose compose paper form . You can also do the permit check first and then switch to a paper form.

Figure 1.2: An online application and notification platform, Municipality of The Hague

Another vision is to ensure the openness and reusability of data by using common storage and exchange formats. By doing this, it allows officials to visualize and check the buildings follow the open standards. Many studies about the digitalization of building permitting use the BIM, mainly coded in its reference open standard IFC by buildingSMART, to check compliance [Noardo et al., 2020c]. However, many regulations need to consider some building context that is not supported by the current buildingSMART. buildingSMART tries to solve this problem by leaving jurisdictions free to add local information requirements and implement their own checklists and rules which are still under construction now [1]. Therefore, we need an alternative way to do it. 3D city models are digital representations of 3D geometric entities found in urban environments, such as buildings, roads, rivers, bridges, vegetation, and city furniture [Billen et al., 2021]. In comparison to typical 2D maps, 3D city models can depict realistic city scenarios with texturing capability Wendel et al. [2017]. Urban visualization can be achieved with 3D city models by simulating the urban building environment and it can be used in many applications. Figure 1.3 shows various application of 3D city models. Therefore, 3D city models are powerful tools for city-level

[1]https://www.buildingSMART.org/regulatory-information-requirements-activity-proposal-out-for-consultation/

analysis and they can do extensions on different City Objects which able to cover the building context in the regulation, therefore it has good applicability to problems related to urban planning [Stoter et al., 2010].

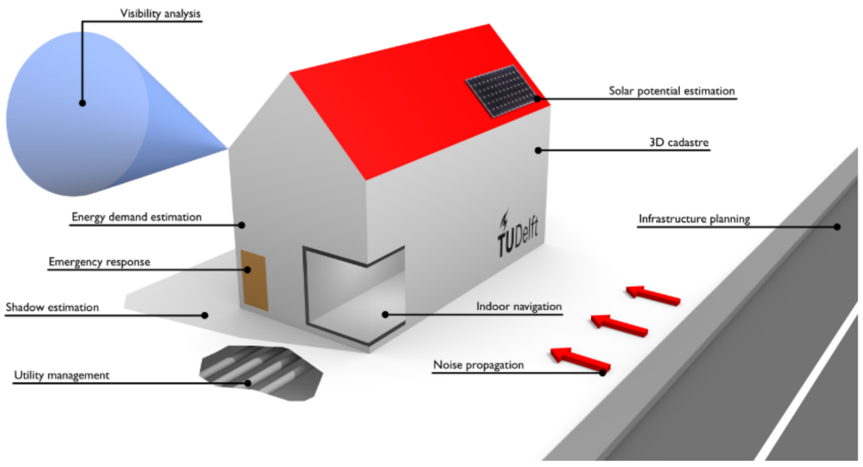


Figure 1.3: Various application of 3D city models [Biljecki et al., 2015]

This study aims to provide an approach to deals with the digitization and automation of the building permits checks under the guidance of EUnet4DBP. To automate this process, this study uses the 3D city model and uses its extensions to support automated building permit inspections. The purpose of final output results is to supporting government officials to check building permits through automatic tools in real situations, which will improve work efficiency, speed up the working process and save costs with better management. The results can also be used for reference to architectural design and developers, and to participate in the process of architectural design.

To achieve building permit checks using a 3D city model, several aspects need to be taken into account. Before conducting building permit checks, we need to choose the regulations for inspection. This involves the interpretation of the regulations, formalization process. Then, the data we obtained and the data model will be stored and extended in a 3D city model and then used for calculation and analysis in the developed tool. After getting the checking results, the final data visualization is also realized. These aspects together constitute the main content of this research, and they are integrated into a tool for automated inspection eventually. By using the framework and tool developed in this research, we can be replicated the process to different places and regulations.

1.2 Research objectives

This research aims to use CityJSON format to construct an urban building data model and develop its extensions to support automated building permit inspections. The data model store the building information which is needed in further calculation and inspection process. There can be different city objects according to different regulations. The main research question of this research is :

- How to do building permit checks automatically by developing a tool in form of the 3D city model?

To achieve the automation of building permits check using tools, the following sub-questions are derived:

- How to select and obtain the information needed by building permit checks?
- How to store and extend the information needed by building permit checks?

- How to test the 3D city model-based tool for the building permit regulation checking?

1.3 Research scope and challenges

The scope of this research is to use 3D city models to check the regulations for building permits issuing, rather than some commonly used building models such as BIM, IFC, because the 3D city model covers a larger spatial extent and can model a variety of City Objects in 3D.

Another difficulty in this research is to interpret the regulation and translate it into machine-readable language. In the process, this research will explain how to interpret and formalize specific regulations and the corresponding operations. However, human interpretation will produce misunderstandings and deviations, leading to different calculation results. For the ambiguous rules in the regulation, they need to be consulted the officials or the expert who makes the regulation.

On the other hand, the available datasets sources are relatively few, basically from the Basisregistratie Adressen en Gebouwen Basisregistratie Adressen en Gebouwen (BAG) data which provides information on the main registered address and buildings in the Netherlands and OpenStreetMap (OSM) data. They lack the necessary attribute information to check certain regulations, which makes the research unable to expand to more regulation at this stage. To solve this problem, the project will be limited to case studies in Rotterdam and start with a few regulations. In this study, we selected the parking regulation on the minimum number of parking spaces for bicycles and cars of Rotterdam as the target. Useful information related to building permit checks (such as different City Objects and their attributes) involved in the regulations will be stored in the 3D city model. However, this step may involve undefined City Objects and new attributes in the existing 3D city model. This can be solved by extending the existing model with the CityJSON extension.

1.4 Thesis outline

The paper consists of 6 parts and is structured as follows:

- Chapter 2 reviews the technologies and research related to this research. Most of the concept and definition is explained here. It focuses on the development of the automation of building permit check and the extension of the 3D city model;
- Chapter 3 gives the research methodology and the whole pipeline of how to carry out the building permit automated checking process. A detailed description of each step is elaborated with diagrams and formulas;
- Chapter 4 provide the use cases of this study to test the methodology;
- Chapter 5 described the conceptual model of CityJSON format;
- Chapter 6 focuses on the implementation process of the research, tools and data sets used, as well as the functions and commands used in the process of realizing building permit automation is present;
- Chapter 7 shows the main results of the research and provides relevant analysis and discussion;
- Chapter 8 gives the conclusion of the research and answers all research questions together with the future work.

2 Related work

The purpose of this chapter is to provide relevant knowledge and work related to building permit checks and 3D city models, which are the key technical points for completing this research. The chapter is structured as follows: Section 2.1 explains what a building permit is and the significance of digitalization of building permits. Some research related to building permits is also introduced. Section 2.2 focuses on the translation of regulations related to building permits for the purpose of the automatic building permit checks process. Section 2.3 provides a brief overview as well as two major open data models and sharing formats for storing and distributing virtual 3D city models. Finally, section 2.4 presents the research gap.

2.1 Digital building permit

Urban planning is based on the premise of development vision, scientific demonstration, and expert decision-making to plan the development of urban economic structure, spatial structure, and social structure, such as land use control and natural environment protection [March and Kornakova, 2017]. It has been promoting the creation of a sustainable urban climate, avoiding urban chaos, and supporting infrastructure development. To cope with new social issues, regional places need sustainable growth, which necessitates effective urban management [Akahoshi et al., 2020].

According to Mouratidis [2021], there is a connection between the built environment and city quality of life. Buildings, as one of the city's most critical components, may provide valuable data for urban planning. A building permit is a permit required for building activities involved in Building Decree [Decree et al., 2012]. A building permit must be obtained for most construction work. A permit is needed to build, modify or demolish a building or change its use, regardless of whether the structure is to be used as a dwelling and whether or not it has foundations. A standard application is used nationwide. Even simple construction projects may require multiple permits. Once the building permit is issued, construction work can start. Figure 2.1 shows the building permit procedures in Turkey. It is also a legal offense to undertake construction work without a building permit if one is required. This can result in a fine or in certain circumstances an order for the building to be returned to its pre-construction state. Therefore, in order to avoid potential safety hazards, the inspection of building permits needs to be supervised by the government. Since the process of building permit inspection involves different laws and requires the joint participation of many departments, it also has many problems, such as low work efficiency, complicated and slow work procedures for permit application generation, and difficult to track Eirinaki et al. [2018].

Therefore, the automation and digitalization of building permits procedures are important since they can improve the quality of permits checks thus the quality of buildings. Many countries give importance to the improvement of the building permit processes. One of the most significant developments in this regard is the European Union's 2002 action plan, which recommends the automation and digitalization of the public services to improve its efficience [Guler and Yomralioglu, 2021]. The digitalization of building permits mainly goes through BIM and related open format IFC, building information can be stored maintained, and queried using these model [Beetz et al., 2010; Mazairac and Beetz, 2013].

In the architecture, engineering, and construction (AEC) industries, comprehensive research has been performed on automated and semi-automated regulatory enforcement inspections in recent years.

Manual inspections of building designs are required in most of countries such as the United Kingdom, based on a collection of constantly evolving and increasingly complex building regulations. For builders and departments in charge of implementing building codes, this is a huge challenge. As a consequence,

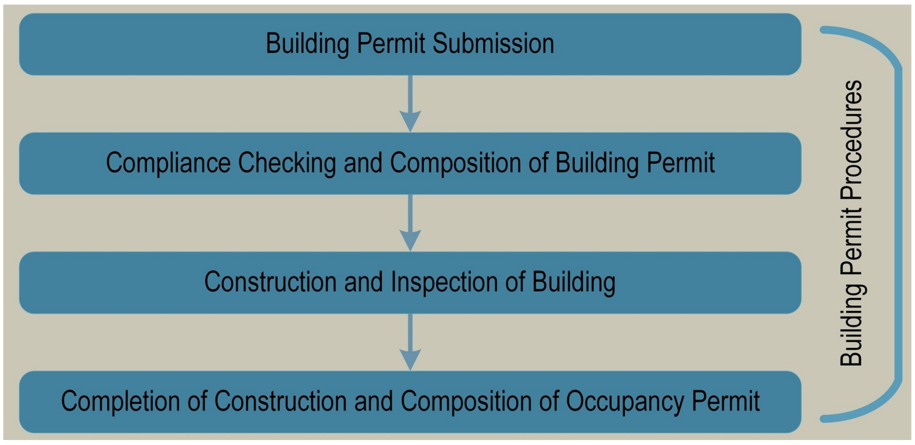

Building Permit Submission
Compliance Checking and Composition of Building Permit
Construction and Inspection of Building
Completion of Construction and Composition of Occupancy Permit
Building Permit Procedures

Figure 2.1: Building permit procedures in Turkey [Guler and Yomralioglu, 2021]

there is always uncertainty, contradictory estimates, and financial loss. As the AEC industry moves from 2D ccomputer-aided design (CAD) drawings to more semantically rich building knowledge models, automated compliance inspection systems for building codes become feasible BIM [Malsane et al., 2015].

According to the findings of Malsane et al. [2015], the IFC format is well suited to automated enforcement tests, they created a semantic-rich object model specific of building fire regulations to meet the needs of automatic enforcement checks in England and Wales.

Dimyadi and Amor [2013] pointed out that due to high-performance personal computers, efficient web-based technologies, and new initiatives in legal knowledge representation modeling, commercial compliance checking systems could be more feasible than ever.

Software reviews that are automated have the ability to save a lot of time and money. It was proposed that the client or architecture firm create the building's procedural criteria and specify basic rules as part of the project design and construction planning and review standards. Eastman et al. [2009] provides a general framework for the implementation of rule-checking systems. It uses a method that translating terms in written statements into the logic of testing conditions and obtains the properties and relations needed for rule queries.

Lee [2011] developed Building Environment Rule and Analysis (BERA), a computer programming language that can be used to help define, interpret, and evaluate rules during the design process, to resolve the issues with the building model.

Preidel and Borrmann [2015] established the minimum criteria for automating regulatory compliance checks and introduced a new approach for automating the process known as Visual Code Checking Language (VCCL).

However, the unified approach to seeking industry recognition does not seem to be over yet. Noardo et al. [2020a] states: "It is widely recognized that a multidisciplinary, multi-stakeholder, and international approach is needed to achieve relevant outcomes in a reasonable amount of time through the exchange of results, expertise, and efforts". With the ambitions and visions, EUnet4DBP was developed in early 2020, bringing together experts from all over Europe in BIM, 3D city modeling, building regulations, urban design, and software development [EUnet4DBP, 2020; Noardo et al., 2020a].

In the framework of automating the inspection of building permits, the ambitions/specification of the EUnet4DBP aims to accelerate the digital transformation of the building permit process. Through exchanging experiences and accumulating information, they aim to develop flexible, scalable, and re-usable digital and (semi-) automated building permit resources and methods in a collaborative effort and under the umbrella of an open science system [EUnet4DBP, 2020; Noardo et al., 2020a].

As part of their research, the 3D geoinformation group at TU Delft developed a tool and methodology to help the municipality of Rotterdam with building permit regulation tests using digital solutions in the project GeoBIM for building permits [1], starting with digital standardised databases, in 2020. To resolve the building permit problems, they used a combination of BIM and 3D city models [Noardo et al., 2020c,b].

Guler and Yomralioglu [2021] provides a reformative framework in Turkey that intends to enhance building permit procedures, 3D property ownership registration, and update 3D city models. It is proposed to link the building permit procedure, the 3D registration of property ownership, and the 3D city model through the effective use of the 3D digital building model to become a new system concept. The reformative framework for building permit procedures, creating 3D city models, and registering property ownership in Turkey is depicted in figure 2.2.

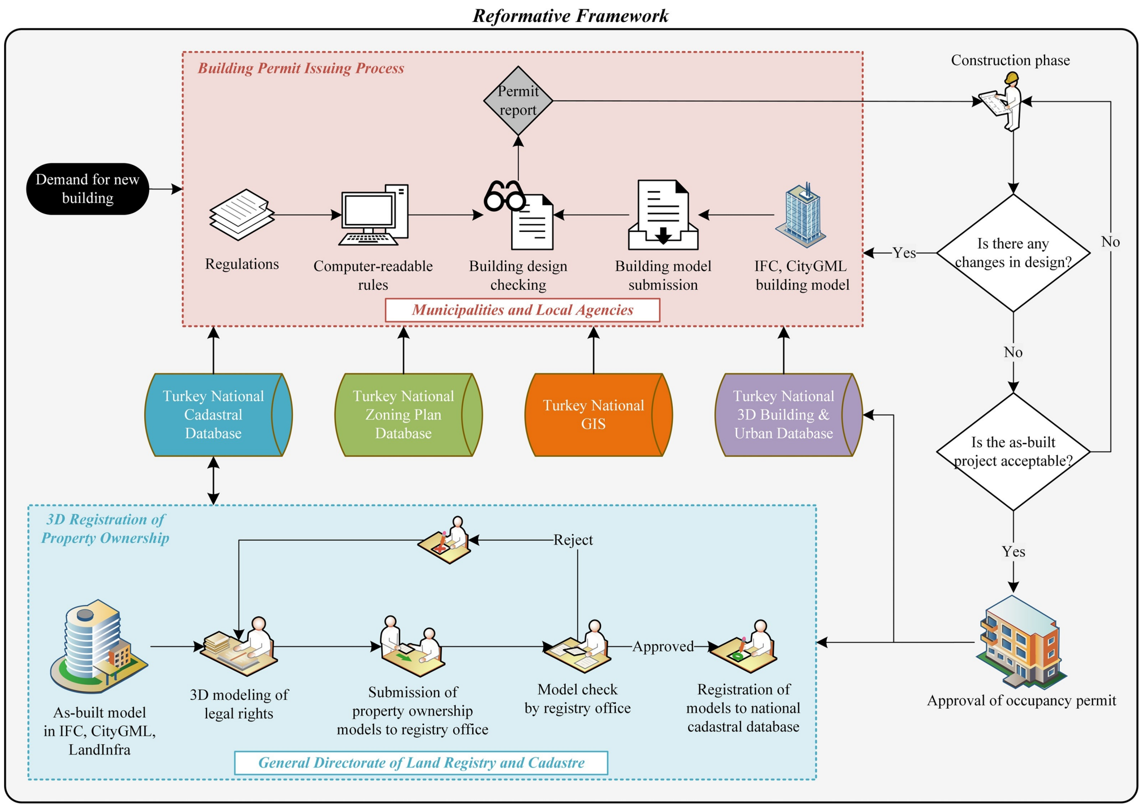

Reformative Framework
Building Permit Issuing Process
Permit report
Construction phase
Demand for new building
Regulations
Computer-readable rules
Building design checking
Building model submission
IFC, CityGML building model
Yes
Is there any changes in design?
No
Municipalities and Local Agencies
No
Turkey National Cadastral Database
Turkey National Zoning Plan Database
Turkey National GIS
Turkey National 3D Building & Urban Database
Is the as-built project acceptable?
Yes
3D Registration of Property Ownership
Reject
Approved
As-built model in IFC, CityGML, LandInfra
3D modeling of legal rights
Submission of property ownership models to registry office
Model check by registry office
Registration of models to national cadastral database
Approval of occupancy permit
General Directorate of Land Registry and Cadastre

Figure 2.2: Reformative framework that combine 3D city model and building permit process [Guler and Yomralioglu, 2021]

2.2 Translation the regulations

One of the difficulties in conducting building permit inspections is the process of interpreting, formalizing the law, and translate the law into machine-readable language. Machine-readable data is data that can be read and interpreted by a computer automatically. Therefore, it must be data that is organized [Open Knowledge Foundation, 2019].

The inherent inconsistencies in the rules, as well as the breadth of circumstances to which they must apply, are major roadblocks to effective rule testing. There are many ways to code the process and lots of rules that can be defined, it's important to systematize the rules which help users to test on them. Using criteria that extend across all current facets of automated rule testing, the author develops a general classification of rules across program domains [Solihin and Eastman, 2015].

[1]https://3D.bk.tudelft.nl/projects/rotterdamgeobim_bp/

To interpret the regulations, Lee et al. [2016] invented a logical rule-based mechanism, refining the objects/properties and predicates of sentences from the Korean Building Act into a database known as the ”logic meta database.” They identify each object and its related properties in accordance with the Korean Building Act. A test server framework has been set up to ensure that this technique complies with the most recent building permit legislation. Figure 2.3 shows the comparison between the rule-making process of their logic-based system and the traditional method.

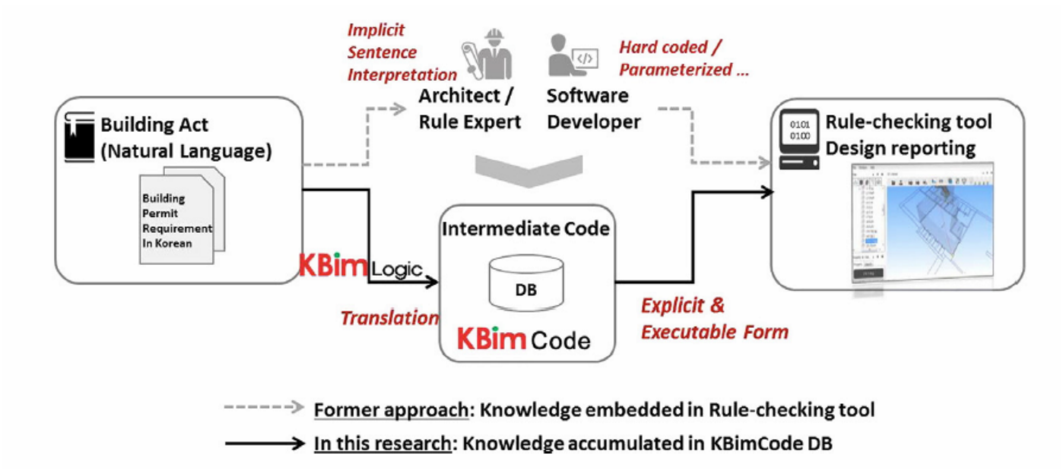

Implicit Sentence Interpretation
Architect / Rule Expert
Software Developer
Hard coded / Parameterized ...
Building Act (Natural Language)
Building Permit Requirement In Korean
Rule-checking tool Design reporting
KBimLogic
Translation
Intermediate Code
DB
KBimCode
Explicit & Executable Form
Former approach: Knowledge embedded in Rule-checking tool
In this research: Knowledge accumulated in KBimCode DB

Figure 2.3: Comparison between the rule-making process of a logic-based system and the traditional method Lee et al. [2016]

In Noardo et al. [2020c], they also provide a reference to formalize the regulation, through definitions and rules designated as true for building permit inspections to translate regulations into codes that can be used for building permit requirements. They both mapped the mechanisms between regulation-specific objects/properties and those in standards (e.g. IFC, BIM, and other 3D city models) needed as shown in figure 2.4.

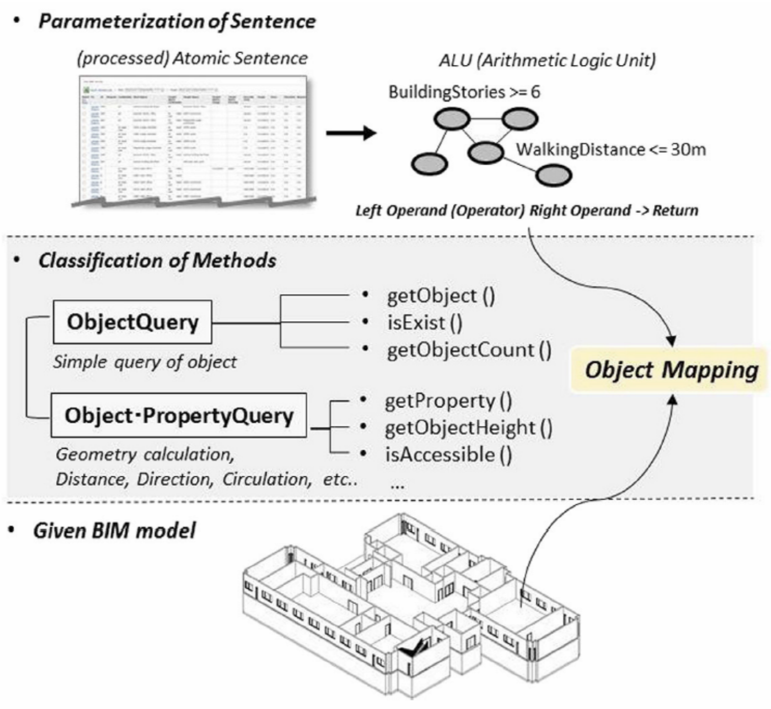

Parameterization of Sentence
(processed) Atomic Sentence
ALU (Arithmetic Logic Unit)
BuildingStories >= 6
WalkingDistance <= 30m
Left Operand (Operator) Right Operand -> Return
Classification of Methods
ObjectQuery
Simple query of object
getObject ()
isExist ()
getObjectCount ()
Object·PropertyQuery
Geometry calculation, Distance, Direction, Circulation, etc..
getProperty ()
getObjectHeight ()
isAccessible ()
...
Object Mapping
Given BIM model

Figure 2.4: An example of mapping information to BIM model [Lee et al., 2016]

2.3 3D City Model

2.3.1 Background of 3D city model

Biljecki et al. [2015] said that "A 3D city model is a representation of an urban environment with a three-dimensional geometry of common urban objects and structures, with buildings as the most prominent feature".

The 3D city model needs to collect the information of various features in the city and then build the model. In Suveg and Vosselman [2004]; Haala and Kada [2010]; Tomljenovic et al. [2015]; Blaschke [2010], they used photogrammetry and laser scanning to build a 3D city model. You can also use existing building models and city drawing data to build models [Donkers et al., 2016; Yin et al., 2008; Lewis and Séquin, 1998]. Use extrusion from 2D footprints get applied in [Ledoux and Meijers, 2011; Arroyo Ohori et al., 2015]. Besides the above methods, there are more other technologies together support the construction of 3D city models. Extrusion from 2D footprints to get the 3D model was adopted in this research. We get the footprint from BAG data source and get height information from 3D BAG data to do extrusion.

3D city models are used in a wide range of applications. In Batty et al. [2001], Batty's team provide a conceptual study on the use of 3D city models, focused on visualization and spatial planning. The visualized virtual simulation of the 3D city model presents the real scene, which can observe the entire city in a macroscopic view. It also supports various analysis functions based on visualization, which is of great help to urban planning and urban construction. They have helped urban development in many areas.

Ross's team offers a taxonomy of 3D use cases as well as a list of applications: (1) applications based solely on geometry; (2) analyses based on geometry and semantic data; and (3) analyses based on domain-specific extensions and external data [Ross, 2011].

3D city models have become useful for a variety of purposes other than visualization and urban planning, and they are used in a wide range of fields. For instance, Sinning-Meister et al. [1996] investigate the applications of 3D city models for CAAD-supported analysis and design of urban areas. Shiode [2000] talks about digital mapping and rendering of urban environments in three dimensions. Ulm [2010] studied satisfaction through sustainability using Virtual 3D City Models. Besides, Ahmed and Sekar [2015] using volumetric analysis of 3D city model in everyday urban planning processes. Lemmens [2011] analyzed the 3D city model that can be used to calculate the sunlight and shadow of the building.

From the above application scenarios, we can find that 3D city models are powerful tools for city-level analysis, so they can be used in building permit inspections. In Guler and Yomralioglu [2021], they proposed a framework including building permit procedures, 3D registration of condominium, and update of 3D city model.

2.3.2 Representing and Exchanging 3D City Models

We need to provide a common description of the basic entities, attributes, and relations of a 3D city model and store it in order to represent and exchange 3D city models [Kolbe, 2009]. Here, two main open data models and exchange formats for storing and exchanging virtual 3D city models and their extensions will be introduced.

CityGML

CityGML is an open data format for storing and exchanging virtual three-dimensional city models. For various thematic areas, it includes a geometry model and a thematic model. It recursively decomposes a city into semantic objects in CityGML 2.0.0. It describes the groups that are most commonly found in a city or an area. Each class in CityGML has five levels of detail, enabling practitioners to use acceptable city representations depending on the application [Gerhard Gröger, Thomas H. Kolbe, Claus Nagel, 2012; Ledoux et al., 2019]. The geometry of the objects is realized using a subset of the ISO 191073 geometry concepts [ISO, 2003]. It is possible to add new classes to the list, as well as define new attributes. Application domain extension (ADE) is a method for accomplishing this, and it entails developing new XML schemas that inherit from the CityGML schemas [Ledoux et al., 2019]. Figure 2.5 provides overview of the UML diagram for the core of CityGML. The public site of CityGML is available at `http://www.citygml.org/`.

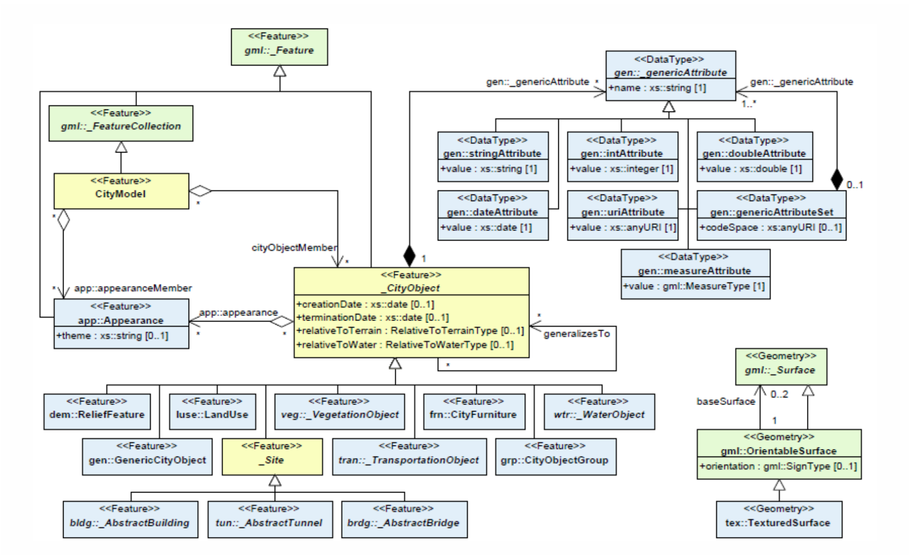

<<Feature>> gml::_Feature
<<Feature>> gml::_FeatureCollection
<<Feature>> CityModel
<<DataType>> gen::_genericAttribute
+name : xs::string [1]
gen::_genericAttribute
gen::_genericAttribute
1..*
<<DataType>> gen::stringAttribute
+value : xs::string [1]
<<DataType>> gen::intAttribute
+value : xs::integer [1]
<<DataType>> gen::doubleAttribute
+value : xs::double [1]
0..1
<<DataType>> gen::dateAttribute
+value : xs::date [1]
<<DataType>> gen::uriAttribute
+value : xs::anyURI [1]
<<DataType>> gen::genericAttributeSet
+codeSpace : xs::anyURI [0..1]
<<DataType>> gen::measureAttribute
+value : gml::MeasureType [1]
cityObjectMember
1
app::appearanceMember
<<Feature>> _CityObject
+creationDate : xs::date [0..1]
+terminationDate : xs::date [0..1]
+relativeToTerrain : RelativeToTerrainType [0..1]
+relativeToWater : RelativeToWaterType [0..1]
<<Feature>> app::Appearance
+theme : xs::string [0..1]
app::appearance
generalizesTo
<<Geometry>> gml::_Surface
baseSurface
0..2
1
<<Geometry>> gml::OrientableSurface
+orientation : gml::SignType [0..1]
<<Geometry>> tex::TexturedSurface
<<Feature>> dem::ReliefFeature
<<Feature>> luse::LandUse
<<Feature>> veg::_VegetationObject
<<Feature>> frn::CityFurniture
<<Feature>> wtr::_WaterObject
<<Feature>> gen::GenericCityObject
<<Feature>> _Site
<<Feature>> tran::_TransportationObject
<<Feature>> grp::CityObjectGroup
<<Feature>> bldg::_AbstractBuilding
<<Feature>> tun::_AbstractTunnel
<<Feature>> brdg::_AbstractBridge

Figure 2.5: Overview of the UML diagram for the core of CityGML [Gerhard Gröger, Thomas H. Kolbe, Claus Nagel, 2012]

CityJSON

A subset of the OGC CityGML data model is encoded in CityJSON, a JSON-based encoding (version 2.0.0). It specifies how digital 3D models of cities and landscapes should be stored. A CityJSON file includes a CityJSON object, the type of which is always "CityJSON," and the minimum acceptable CityJSON object must have the four elements listed below:

```
{
  "type": "CityJSON",
  "version": "1.0",
  "CityObjects": {},
  "vertices": []
}
```

The value of the member "CityObjects" is a set of key-value pairs, with the key being the object's ID and the value being one City Object. A City Object's ID should be unique. For compression purposes, the member "vertice" holds a global collection of 3D vertex coordinates. The aim of CityJSON is to provide an alternative to CityGML's GML encoding. It is built to be lightweight (with real-world datasets, it has a compression factor of around six) and friendly for web and mobile growth. CityJSON strives to be easy to use, both for reading and writing datasets. It was created with programmers in mind, allowing for the rapid development of tools and Application Programming Interface (API) to support it. Via the use of a simplified JSON encoding, CityJSON significantly reduces the difficulty of designing applications for the CityGML data model. JSON is a basic data exchange format that many programming languages, including JavaScript, Python, and Ruby, support natively. In chapter 4, you'll find some tools for creating, parsing, visualizing, editing, and validating CityJSON files [Ledoux et al., 2019]. The public site of CityJSON is available at `http://www.cityjson.org/`.

Extension of 3D city model

In some urban planning applications, attributes or new urban objects that aren't specified in the CityGML data model may be required. CityGML proposed the idea of ADE to enable users to add new attributes or artifacts to the data model. It creates new classes by inheriting the classes that make up the CityGML data model, and it modifies existing classes by introducing new geometric shapes and complex attributes, for example. It saves all of this information in an additional XSD file [Gerhard Gröger, Thomas H. Kolbe, Claus Nagel, 2012].

CityJSON describes extensions in a similar way. The extension is a JSON file that describes how to expand CityJSON's core data, and it is used to validate the CityJSON file [Ledoux et al., 2019]. Their core is to allow users to record in a standardized manner and to validate files with extensions. Following that, current CityGML ADE and CityJSON extensions will be discussed.

In the information infrastructure of i-Urban Revitalization built by the Japanese government, for better urban planning, they used the CityGML ADE mechanism to combine urban semantic information with three-dimensional information to realize the visualization and analysis of the current situation of the city [Akahoshi et al., 2020]. In order to store the data related to urban planning-specific semantics information, they expand the existing data model of CityGML. The data involved in this infrastructure is called i-UR data and they made 3 different data modules for urban objects, urban function and statistical grid. The infrastructure is constructed by make extension of the existing modules for city objects such as building, land use, transportation or create new city object from core::CityObject. They also explored ways to scale the data model globally to compare the development of different cities [Ishimaru et al., 2020]. Figure 2.6 shows some data experiment examples of the Urban Planning ADE which confirmed that the buildings in the area are basically in line with the zoning plan.

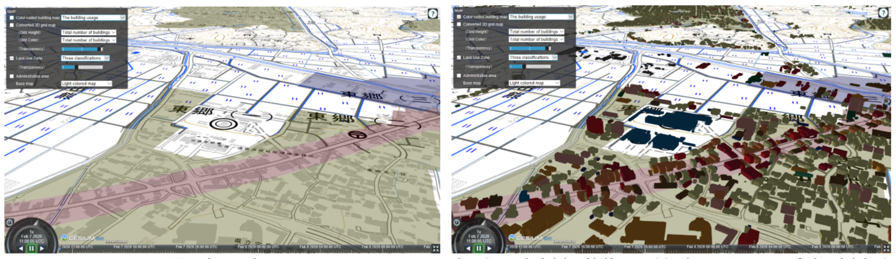

a. Zoning plan b. Overlaid buildings (Color: Usage of the bldg.)

Figure 2.6: Data Experiment Examples of the Urban Planning CityGML ADE [Ishimaru et al., 2020]

Similar to CityGML ADE, CityJSON extensions also support the extensions to the core data model of CityGML for specific applications and use-cases. But the existing CityJSON extensions are still relatively few. 3D geoinformation group of TU Delft developed a CityJSON extension that allows topological information of a city model to be stored as a Combinatorial Map-based Linear Cell Complex. The findings show that the proposed CityJSON extension can accurately reflect all geometric information in a city model while also providing additional topological information for the model's artifacts [Vitalis et al., 2019]. Figure 2.7 shows the Linear Cell Complex (LCC) viewer used to evaluate the topologically reconstructed CityJSON files.

Figure 2.7: A LCC viewer to evaluate the topologically reconstructed CityJSON files [Vitalis et al., 2019]

2.4 Research gap

A gap in current research is how to develop a tool that using 3D city models for building permits checks. At present, the digitization of building permits is mainly done through BIM and the related open format IFC. This study proposes an alternative method of using 3D city models. There is also a gap in the CityGML ADE, the urban planning CityGML ADE published by the Japanese government, which stores data for urban master planning and promotes city revitalization on a large scale [Akahoshi et al., 2020]. But specific urban planning applications are not yet available, such as building permit checks that relate to specific regulations and other needs of users. It require a useful data structure to store the necessary information. In addition, there are few applications for the CityJSON extension, so it is also a challenge to explore the application of the extension in different application fields.

Therefore, this research focuses on this research gap in using 3D city models to complete building permit inspections, and develops tools to complete this process.

3 Methodology

This part gives the overview of the methodology to be used in this research as shown in figure 3.1. The key point of this research is to using 3D city models to complete building permit checks and develops tools to automate the process.

Section 3.1 describes the interpretation and formalization of regulations. This step aims to translate the regulation of natural language into algorithms for further automated computation and analysis. Then, Section 3.2 gives use cases on how to interpret and formalize the parking standard for bicycles and cars. The comparison of CityGML and CityJSON is presented in Section 3.3, After evaluation, this research chose to use CityJSON to build a conceptual model. We then utilize the model in building permission check applications. It will store, evaluate, and verify the final results. Followed by a description of extending 3D city model by using the CityJSON extension in Section 3.4. Finally, Section 3.5 briefly describes the working steps of the tool as the final result, which can automatically complete building permit inspections based on specific regulations.

This research will be carried out with a specific case study, by selecting a small area of the research area in Rotterdam and specific regulations to conduct the research as test scenarios, and then try to test the possibility of extending it to the realization of more regulations.

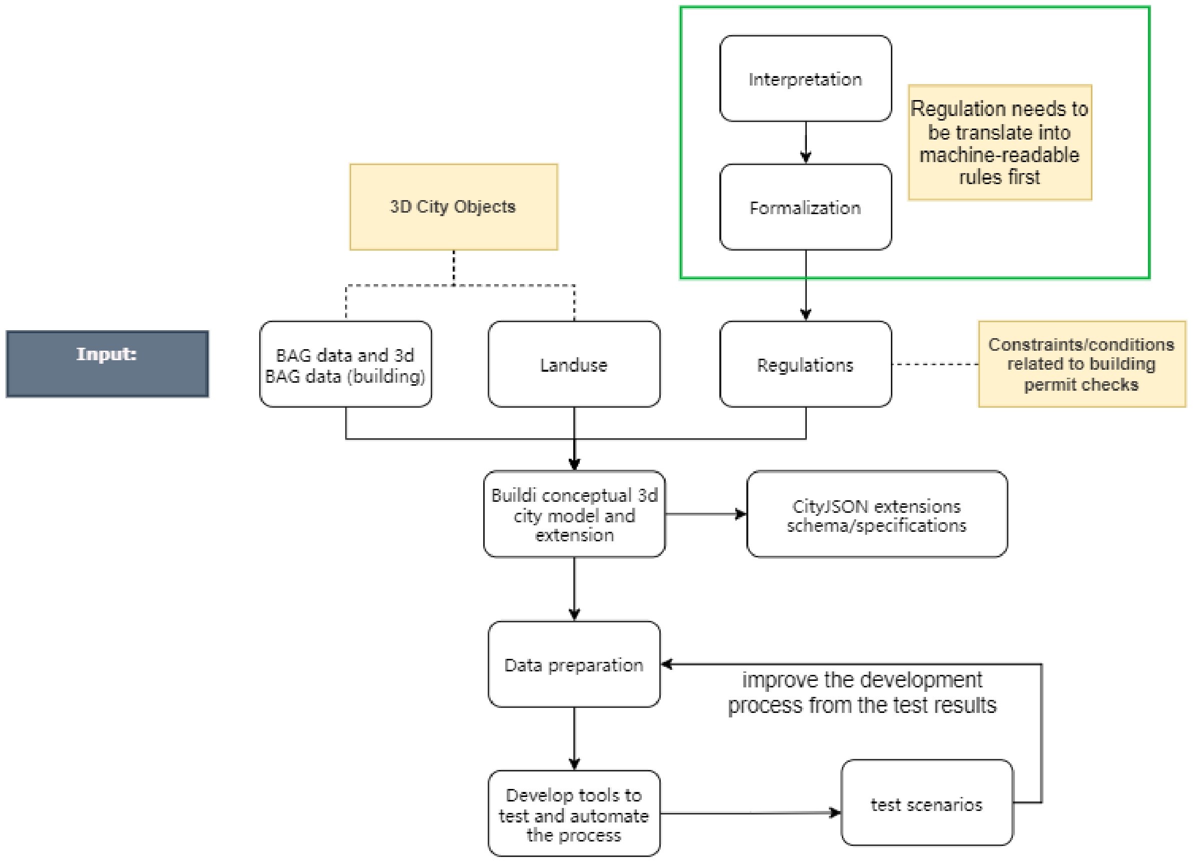

Interpretation
Formalization
Regulation needs to be translate into machine-readable rules first
3D City Objects
Input:
BAG data and 3d BAG data (building)
Landuse
Regulations
Constraints/conditions related to building permit checks
Buildi conceptual 3d city model and extension
CityJSON extensions schema/specifications
Data preparation
improve the development process from the test results
Develop tools to test and automate the process
test scenarios

Figure 3.1: Workflow of methodology

3.1 Interpretation and formalization of regulations

Before interpreting and formalizing the regulations, it is necessary to sort out the regulations that affect the research area, develop methodology through the selected regulations and design corresponding building permit checking tools. For the selection of regulations and the interpretation and formalization of specific use cases, it refer to chapter 3.2.

3.1.1 Interpretation of regulations

The first part is about the interpretation of regulations, which means to understand and summary the regulation. The project 'GeoBIM for building permits' of 3D geoinformation group at TU Delft pointed out that there are ambigraphical user interface (GUI)ties in the interpretation of regulations in natural language. Their solution is to eliminate the ambiGUIty in the interpretation of the rules with the help of municipal officials and experts [Noardo et al., 2020c]. Through the interpretation of the regulations, we can better understand the meaning of the regulations and how to apply them to building permit inspections. Although different people may have different interpretations of the same regulation, this research will be interpreted as close to the original regulation as possible based on its description. For the uncertain part, relevant experts were contacted for consultation.

3.1.2 Formalization of regulations

To minimize the deviation caused by manual interpretation of the regulations and provide corresponding basis support for the interpretation of the regulations for this study, the formalization of the nature language regulations will be based on logic through definitions and rules designated as true for building permit inspections. The process of formalization also called rule-making process. The significance of formalization in translating them into machine-readable language is to support tools that automatically check building permits. The formalization process specifies the attribute information that needs to be obtained, as well as logical relationships and related calculation requirements.

In the studies discussed in section 2.2, it is mentioned that the information of the building model is mapped into the formalization process. To effectively enforce the methods, mapping mechanisms between regulation-specific objects/properties and those in standards (e.g. IFC, BIM, and other 3D city models) are needed. With the relations between regulation and 3D city models, we can then construct the new building data models with CityJSON which can be find in section 3.3.

Although the theme and constraints of different regulations vary, by summarizing their commonalities and concluding from the regulations that have been explained, a working process that can be extended to other regulations is proposed which is shown in the figure 3.2. After a preliminary assessment of the possibility of formalizing the code, we will complete the permit inspection of the building by interpreting and formalizing the provisions of the regulation.The steps are concluded as follow:

1. Manually interpret the regulations and consult local experts if there is any ambiGUIty.

2. Disassemble the legal text as: object + attributes, function, conditions and constraints

3. Transformed the regulations into algorithms.

4. Mapping the object/attribute with 3D city models.

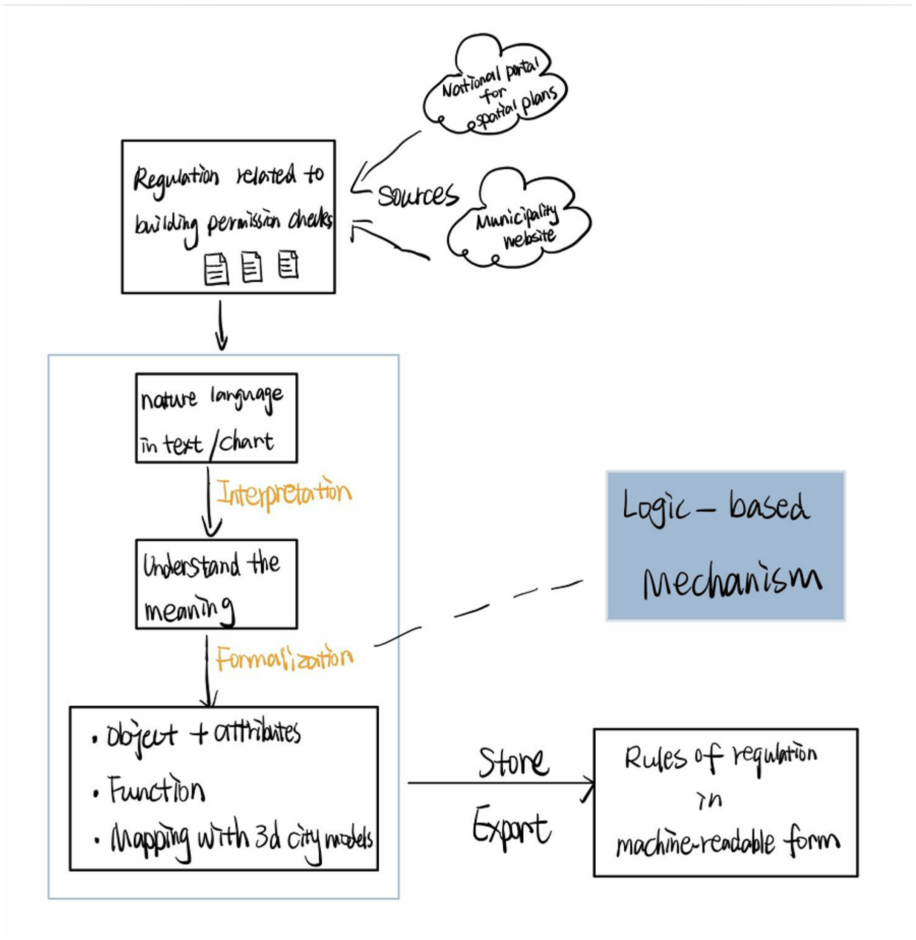

National portal for spatial plans
Regulation related to building permission checks
Sources
Municipality website
nature language in text /chart
Interpretation
Understand the meaning
Formalization
Logic – based Mechanism
• Object + attributes
• Function
• Mapping with 3d city models
Store
Export
Rules of regulation in machine-readable form

Figure 3.2: Workflow of the interpretation and formalization of regulations

3.2 Use cases for regulations

3.2.1 Regulation selections

Zoning plans, structural visions and general rules implemented in Rotterdam can be found on this website: `https://www.ruimtelijkeplannen.nl/`.It is the national portal for spatial plans of the Netherlands. In this website, you will find zoning plans, structural visions and general rules made by municipalities, provinces and the national government. They can be classified according to administrative levels and theme:

- The administrative level: Gemeente, Provincie, Rijk(municipality, province, country).
- Theme: Environmental (water,wind), Spatial related, Transport, Vision.

This study needs to sort out these regulations to select the regulation that can be used for building permit inspection. In Ruimtelijkeplannen portal as shows in figure 3.3. Enter an address or place, and you can see the regulations implemented by the municipality, province, and national government where it is located. Clicking on the corresponding regulation will redirect you to a detailed page describing the content. Determine the selected regulation of the study by screening the regulations that applicable to Rotterdam.

It was decided to start this research and focus on the regulations related to the spatial planning since they are easier to start and implement. these regulations were chosen because they have more clearer description and they don't need so much different external data. Also, the calculation can be easier.

We can check them with the data available and by building suitable analysis tools based on the 3D city model. Use them to see if the approach developed in this study is practicable, after which it can be applied to other areas of regualtions. In Rotterdam, the municipality government announced a policy regulation on parking standards for bicycles and cars based on different building functions. It focuses on the spatial planning itself, so that it is easier to get started and get the desired results than other regulations that covering more themes (such as environmental protection). This regulation applies to the whole city of Rotterdam and has a good application prospect for urban planning traffic and transportation. Therefore, it becomes the use case for validating the methodology.

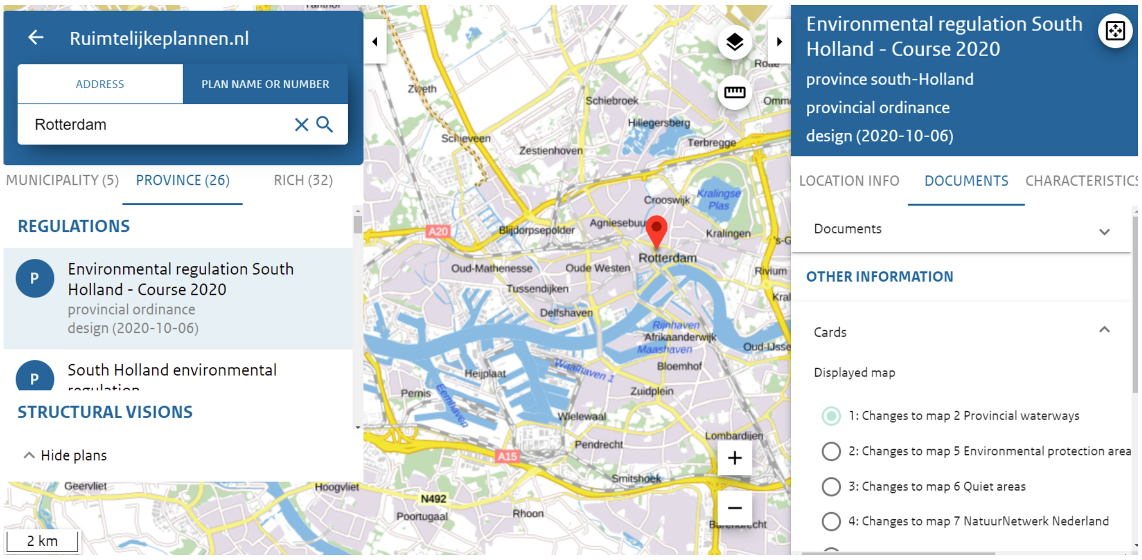


Figure 3.3: Find regulation applied in the Netherlands on Ruimtelijkeplannen.nl

The regulations mentioned here is about calculating the minimum number of bicycle parking spaces and car spaces required in Rotterdam buildings. The regulations' aim is to ensure that enough parking space for cars and bicycles is available for the expected functions in the construction of new buildings. The regulations identify "sufficient parking spaces" as parking spaces for cars and bicycles that achieve the best balance between different municipal policy objectives in order to carry out good space planning. The detail information of this regulation can be find at this website [1]. To meet the needs of Rotterdam's urban planning, new buildings must meet the reasonable number of parking spaces mentioned in the regulation for people to use when checking building permits. With the increasing use of bicycles, in addition to parking spaces for cars, the regulation also included bicycle parking facilities for the first time.

3.2.2 Interpretation of parking regulation

The Rotterdam City Government's policies and regulations on parking standards for cars and bicycles describe the background, reasons, and purpose of the policy for implementing the law. The policy contains three strategies, namely general policy regulations for cars and bicycles, additional vehicle policies, and additional bicycle policies. The competent authority also stipulated in the regulations that building requirements that can completely or partially deviate from the parking requirements. They are divided into the following situations as shows in table 3.1. For example, if the area of a residential building is less than the designated area (300m2), it can be exempted from car parking spaces.

The parking standard provides standard tables that specify the calculation requirements for the parking capacity of bicycles and cars to be realized during the construction of buildings with different functions. These tables respectively indicate the calculation of car parking spaces for residential functions, car parking spaces for non-residential functions, bicycle parking spaces for residential functions, and

[1]https://lokaleregelgeving.overheid.nl/CVDR486392/1#noot_id1-3-2-4-91-1-7-1-2-1-2

Rules	Is it considered in this study?	Why it is not involved in this study?
Close to parking facilities	NO	No calculable conditions are provided in the law
The building area is less than the designated area can be exempted	YES	-
If the parking facilities of the building can be used for multiple functions, the parking requirements can be reduced	NO	Complex content
Different degrees of special exemptions are available for nearby public transportation stations	YES	-
Deviation from nursing home	NO	Individual exceptions are not considered
Separate parking standards for social rental housing	NO	Individual exceptions are not considered

Table 3.1: Some situations that can deviate or be exempt from parking requirements

bicycle parking spaces for non-residential functions. The tables in the regulation can be found in the appendix. For reasonable space planning, Rotterdam is divided into three different zones in the regulation which can be seen in figure 3.4. The calculation of car parking spaces for buildings located in each area is different while the calculation of the number of bicycle parking spaces is not affected by the area. The function of the building also determines the number of parking spaces.

To simplify and speed up the calculation and inspection process, it is necessary to check whether the building meets the exemption conditions. If it meets the requirements, the next step of calculation will not be performed. The table to check each exemption condition is also shown in the appendix. For the exemption conditions for small-area buildings, we only need to check whether the area of the building meets the requirements. For the transportation stations in Rotterdam, public transportation is more convenient, so buildings located near the transportation stations can get parking space reduction. In this regard, we need to extract the corresponding traffic stations from the OSM data and perform buffer analysis according to the requirements given in the table. If the building falls in the corresponding buffer zone, different degrees of exemption can be obtained.

After completing the screening of parking spaces exempt buildings, the next step is to calculate the minimum number of parking spaces based on the four different forms provided by the law. According to regulations, bicycle parking spaces and car parking spaces in each building have different computer systems, so they will be discussed separately in the following formalization sections 3.2.3.

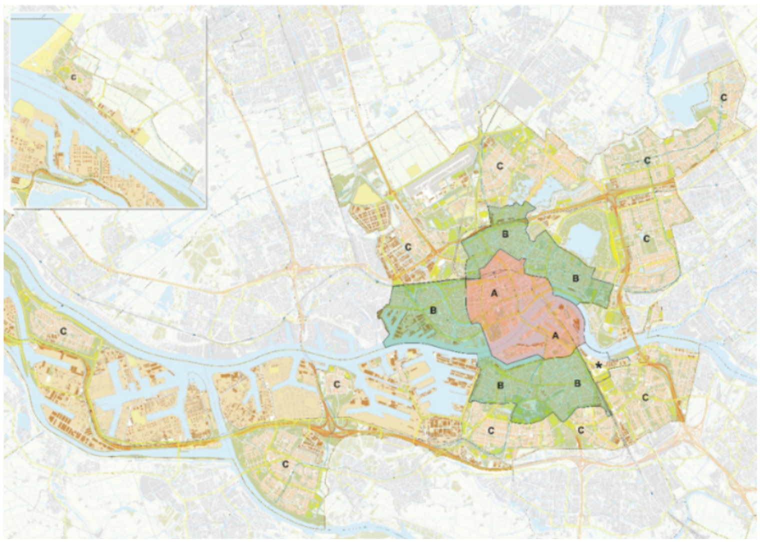


Figure 3.4: Regional division of Rotterdam

3.2.3 Formalization of parking regulation

According to the calculation formula provided by the regulation, the minimum number of parking spaces for bicycle and cars can be obtained. A more clear calculation formulas and condition constraints tables are given in the appendix. Based on the interpretation information, the steps of checking bicycle and car regulation can be summarized together as the following steps:

1. Determine what needs to be defined and its attributes. Such as defining the building, building function, room, room area, and the zone where the building footprint is located.

2. Check whether the building meets the parking space exemption conditions, check its area and whether it is located near a traffic stop.

3. Filter the types of building functions, residential buildings will be calculate first according to the formula table.

4. For non-residential buildings, determine which type of non-residential building they belong to, query the table and get the result.

5. Compare the calculation result with the number of parking spaces in the building design plan to get the building permit inspection result.

3.2.4 Formalization of bicycle parking regulation

For bicycle parking spaces calculation, the same calculation applies to all three zones in the bicycle parking standard. As for building functions, the calculation of residential buildings and non-residential buildings is different:

- The number of parking spaces in a residential building depends on the number of rooms of different areas in the building.
- The number of parking spaces in non-residential buildings depends on the number of recommended parking spaces given by different functional types (e.g. office, shops).

Based on the known knowledge, the formalization of the bicycle regulation can be summarized as shown in the following table 3.2. We first determine what needs to be defined and its attributes as 'Definitions' in the table. For the convenience of expression, we use abbreviations to represent each 'Definitions' and give an explanation in the second column of the table. We then use a series of conditional statements to make judgments and calculate the results. Finally, we use the discriminant of the last line in the table to determine whether the new parking spaces meet the requirements. The terminology used in this formalization process is based on [Noardo et al., 2020c].

3.2.5 Formalization of car parking regulation

For the car parking space calculation, the calculation result depends on which zone of the building is located. As for building functions, the calculation of residential buildings and non-residential buildings is also different:

- The number of parking spaces in a residential building depends on the number of rooms of different sizes in the building and the zone where the building is located.
- The number of parking spaces in non-residential buildings depends on the number of recommended parking spaces given by different functional types (e.g. office, shops), the building area and the zone where the building is located.

Based on the known knowledge, the formalization of the car parking spaces regulation can be summarized as shown in the following table B.2.

Table 3.2: Formalization of bicycle parking regulation

Definitions and rules from regulation	Definitions
For residential buildings:	
BUH40 = Count BU (function."home")	
AND (A(BU) 40 m2)	
BUH40-65 = Count BU (function."home")	
AND (40 A(BU) 65 m2)	
BUH65-85 = Count BU (function."home")	
AND (65 A(BU) 85 m2)	
BUH85 = Count BU (function."home")	
AND (85 m2 A(BU))	
Rules (must be true)	
IF BU(function) = "home"THEN	
MinNPP= (BUH40*2) + (BUH40-65*3)	
+ (BUH65-85*4) + (BUH85*5)	
NewParkings $\geq$ sum(MinNPP) + sum((MinMQPP/parkingArea))	BU = Building unit, single dwelling
For non-residential buildings:	BUH= function "home"
BUH40 = Count BU (function."home")	BUO= function "Office"
AND (A(BU) 40 m2)	
BUH40-65 = Count BU (function."home")	BUI= function "Industry"
AND (40 A(BU) 65 m2)	
BUH65-85 = Count BU (function."home")	BUR= function "Retail1"
AND (65 A(BU) 85 m2)	
BUH85 = Count BU (function."home")	BUS= function "Supermarket"
AND (85 m2 A(BU))	
Rules (must be true)	BUG= function "Gym"
BU(function) != "home"	
	BUM= function "Museum"
IF BU(function) = "Office"	
THEN MinMQPP = 1.7 * A(BU)/100	BUC= function "Cinema"
ELSE IF BU(function) = "Industry"	BUC1= function "catering I"
THEN MinMQPP = 1* A(BU)/100	
	BUC2= function "catering III"
ELSE IF BU(function) = "Retail1"	
THEN MinMQPP = 2.7* A(BU)/100	BUC2= function "catering IV"
ELSE IF BU(function) = "Supermarket"	BUU= function "universities"
THEN MinMQPP = 2.9* A(BU)/100	
	BUH= function "Hospital"
ELSE IF BU(function) = "Gym"	
THEN MinMQPP = 2.5* A(BU)/100	BU(function) = attribute 'function'
	related to each building unit (room) .
ELSE IF BU(function) = "Museum"	
THEN MinMQPP = 0.9* A(BU)/100	A (BU) = attribute'area'related to
	each building unit
ELSE IF BU(function) = "Cinema"	
THEN MinMQPP = 7.8* A(BU)/100	MinNPP = Minimum number of
	parking places.
ELSE IF BU(function) = "catering I"	
THEN MinMQPP = 9* A(BU)/100	MinMQPP = Minimum parking places
	per 100 square meters
ELSE IF BU(function) = "catering III"	
THEN MinMQPP = 18* A(BU)/100	
ELSE IF BU(function) = "catering IV"	
THEN MinMQPP = 18* A(BU)/100	
ELSE IF BU(function) =n "universities"	
THEN MinMQPP = 13* A(BU)/100	
ELSE IF BU(function) = "Hospital"	
THEN MinMQPP = 0.9* A(BU)/100	
NewParkingArea $\geq$ sum(MinMQPP) + sum((MinNPP*parkingArea))	

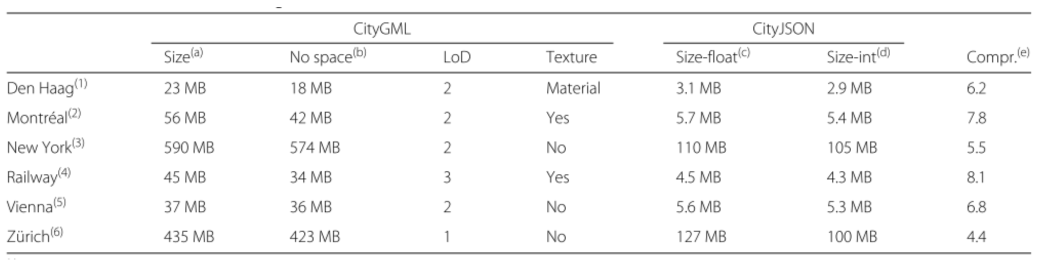
3.3 Construction of conceptual model in CityJSON

After the interpretation and formalization of the regulations, we need to build a 3D city data model and construct a mechanism like extensions to extends different city objects. City Objects related to the regulations will be stored in a data model, which records the attributes and geometry of each city object. In chapter 2, we introduced two types of open data models and exchange formats for storing and exchanging 3D city information that are CityGML and CityJSON. By analyzing the advantages and disadvantages of these two methods, and based on this research situation, the most suitable data format modeling is selected.

3.3.1 CityGML versus CityJSON

CityGML's GML coding is complicated and hard to understand, so it is unfriendly to developers. The difficulty of decoding CityGML files, describing all of the various ways of storing geometry, solving XLink, managing different CRS, and adding support for ADE has made CityGML support more difficult for developers. As a result, the number of tools and software that can support CityGML processing is limited, which dampens researchers' enthusiasm for the standard and is particularly detrimental to the creation of small independent standards and projects, such as the scope of this study. An alternative to CityGML is needed to reduce the difficulty of research and to provide a better user experience [Ledoux et al., 2019].

By using a simplified JSON encoding, CityJSON significantly reduces the difficulty of designing applications for the CityGML data model. JSON is a basic data exchange format that is supported by a wide range of programming languages. CityJSON is a CityGML data model based on JSON encoding that can perform the majority of CityGML's functions. On the basis of the preceding, CityJSON does not require the development of a parser in the same way that CityGML does. Parsing CityJSON files is typically a one-line process, and querying the created data tree is simple with standard functions. This reduces the risk of errors in the parsing and compilation process. The files encoded with CityJSON are more compact than CityGML. In the test showed in figure 3.5, the CityJSON files are shown to be six times more compact. In practice, this feature makes it easier and more versatile, which is beneficial for users manipulating datasets [Ledoux et al., 2019]. Therefore, in this reseach, CityJSON will be used to build a urban planning data model.



	CityGML				CityJSON		
	Size(a)	No space(b)	LoD	Texture	Size-float(c)	Size-int(d)	Compr.(e)
Den Haag(1)	23 MB	18 MB	2	Material	3.1 MB	2.9 MB	6.2
Montréal(2)	56 MB	42 MB	2	Yes	5.7 MB	5.4 MB	7.8
New York(3)	590 MB	574 MB	2	No	110 MB	105 MB	5.5
Railway(4)	45 MB	34 MB	3	Yes	4.5 MB	4.3 MB	8.1
Vienna(5)	37 MB	36 MB	2	No	5.6 MB	5.3 MB	6.8
Zürich(6)	435 MB	423 MB	1	No	127 MB	100 MB	4.4

Figure 3.5: Comparison on the size between CityGML file and CityJSON file [Ledoux et al., 2019]

3.3.2 Data pre-processing

The data and models in this study are all based on CityJSON format, so different source data need to be converted into CityJSON file first. Here, the source data is from BAG 3D BAG and OSM data (The detail information of data can be found in chapter 4). It contains building information that is needed for the analysis. Since they are all of geojson file, we need to covert them into CityJSON data according to the CityJSON Specifications 1.0.2 which can be find at `https://www.cityjson.org/specs/1.0.2/`. We then covert the building data, room data and other city objects data into CityJSON format and extract the data of Rotterdam.

3.3.3 Build conceptual model

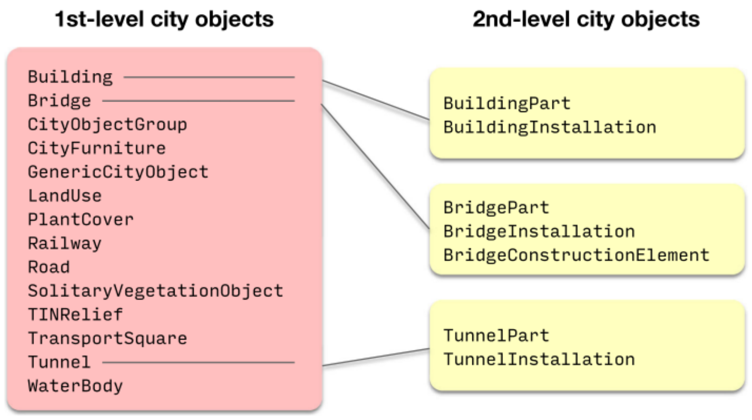


Figure 3.6: The implemented CityJSON classes, they are divided into two levels [Ledoux et al., 2019]

Next, different city object data can be combined to form the conceptual model of urban building. Their attributes and geometry will be stored in the data model. Some city object classes have been defined in CityJSON as shows in figure 3.6. They are divided into 1st-level objects and 2nd-level objects according to whether they can exist alone. City Objects of 1st-level can exist by themselves while 2nd-level ones need to have a parent.

CityGML include 5 well-defined consecutive Levels of Detail (LOD) which is used for represent objects in different fields. In Biljecki et al. [2016], they provide a sequence of 16LODs (4 refined LOD for each of the LOD0–3), allowing for less ambiGUIty and assisting practitioners in standardizing their data through a better characterization of model complexity. Figure 3.7 shows a visual example of the refined LOD. In our research, the LoD of the building is "lod" = 1.2. City objects at different levels of detail in UML diagrams contain different elements at a specific LOD. The city object inherits all the attributes of `core:: _cityObject`, and defines its own attributes that need to be included and the specified geometry class. To build conceptual model, we need to follow the specification of CityGML [Gerhard Gröger, Thomas H. Kolbe, Claus Nagel, 2012].

3.4 Extension of 3D city model by using CityJSON extension

We used CityJSON's existing city object classes to complete the preliminary data model construction. However, for some objects that do not exist in CityJSON's defined city object classes and some additional object attributes, we need to extend the data model as mentioned in section 2.3.2.

CityJSON Extension is a JSON file that allows us to record how to extend the core data model of CityJSON and verify CityJSON files. Although the flexibility of extension is not as good as CityGML ADE, CityJSON Extensions can be read and processed by any software that supports CityJSON, which is convenient and friendly for developers [Ledoux et al., 2019].

The data model built in the previous step will be included in the CityJSON extension, which extends different city objects according to the rules of the CityJSON extension to ensure data interoperability. To verify the validity of the data model, a corresponding schema needs to be written, and then the data and model is validated with cjio and val3Dity.

3.4.1 Extensions

By adhering to the CityJSON extension mechanism, the data model was created to be compatible with CityJSON. According to the approach mentioned in Akahoshi et al. [2020], we can map the method

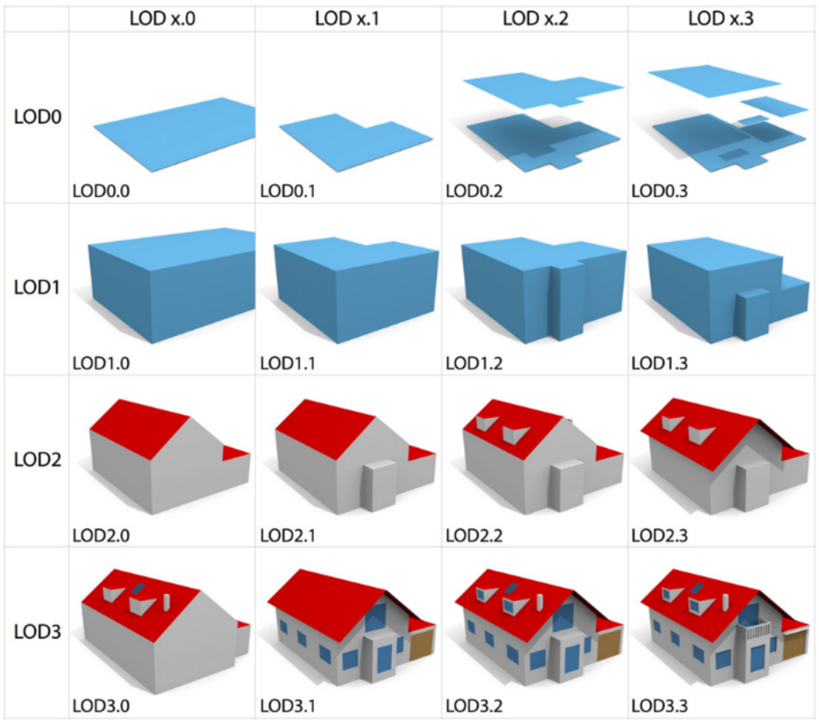


Figure 3.7: Visual example of the refined LODs for a residential building [Biljecki et al., 2016]

to the CityJSON extension modeling. That is, basic city planning items that are already specified in CityJSON can be reused, while items that are not defined in the CityJSON specification are modeled according to the CityJSON extension rules.
First, we classify all the city objects covered by the regulations:

- Objects and properties already defined in CityJSON
- New objects and properties which is not defined

Objects and properties that are not defined are classified according to their theme can be roughly divided into two categories:

- Extension of existing city objects (e.g. buildings, rooms)
- Virtual city objects (e.g. regulation, storing its constraints)

In CityJSON specifications, it already defined some city objects showned in figure 3.6. City objects like building and landuse are already defined here, so they can be directly reused in our data model and add some new attributes if needed. For objects that are not defined in the specification (e.g. regulation), we need to create a new city object for it which inherited from `core:: _cityObject`.

In Ledoux et al. [2019], three ways of extending the core data model are specified:

- Add new attributes to existing city objects
- Create new city objects and define their geometry
- Add new attributes at the root of the document

For the case of this study, the first and second methods are mainly used. For buildings and landuse, besides its basic attributes, we need to add new attributes according to the regulations by add them as ”extraAttributes”. An extra attribute must start with a ”+” to show that it comes from the extensions. For room and regulation, we need to create a new City Object for them and define its attributes and geometry. They are define in the "extraCityObjects" property in extension file. A new City Object must be begin with a "+". The UML model of CityJSON model can be found at section 5.1.

3.4.2 Construction of schema of extensions and validity checks

As we extend CityJSON's existing data model, a new schema is needed for better interoperability. This schema should document the data model and sets specifications for how to extend different City Objects and will be used to verify the validity of the data model. According to CityJSON specification, 7 members need to be specified in the extension file, as shown in the figure 3.8. Because the extension file is also a CityJSON file, in order to ensure that it conforms to the specification, it needs to be validated. A small validation script is used to check its syntax, which can be found at `https://github.com/cityjson/misc-example-code/tree/master/validate-extension`.

After we finished writing the extension schema and have some CityJSON files that containing extensions, we also need to check the validity of those files using the schema. The tools required for validation are mentioned in chapter 4. For syntax validity, we use cjio for validation by specify path of the extension file. It is available at: `https://github.com/cityjson/cjio`. As for 3D primitives, val3Dity is used for geometry checking. These two programs will automatically complete the verification process by setting the parameters [Ledoux, 2018].

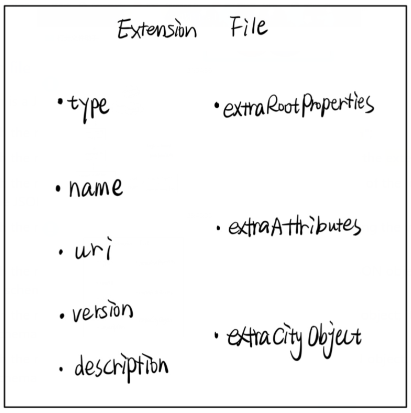


Figure 3.8: 7 members of a CityJSON Extension file

3.5 Development of a program for permit checks automation

After understanding and formalize the regulation into machine-readable language, we aim to develop a tool to automate building permit inspections process. As mentioned in chapter 1, this research intends to develop a tool for building developers and government officials to use to make it easier and simpler to complete the inspection of building permits. The tool can automate the entire inspection process and has a user interface for better interactivity.This GUI program includes the code to complete the automated inspection of building permits. The user specifies the input file and output file. The program will automatically run the corresponding functions, perform calculations and check. Using the PyQt module in python to develop a desktop application with a graphical interface is the solution in this research. We design the UI framwork to include the required UI components, such as adding buttons, text boxes, and so on. We define the behavior and content of each component through scripts. The button defines the event that will respond when the button is clicked through the clicked.connect() function, where the response is a function to calculate the minimum parking space of each building. The text component defines the content displayed in the text box, where the progress of the program running and the result of the validity check of the final file will be displayed. The GUI program has the following structure:

- Create icons and widgets that are displayed to users, and organize them in screen windows. This program is a tool for building permit inspections for new buildings.
- Define the function that will handle user and application events, here is the calculation of parking spaces.
- Associate specific user events with specific functions (for example, clicking the button for the final calculation).
- Write the main code for regulation inspection in the program, which includes the processing and calculation of the input data and the verification of the syntax and geometric validity of the final result. In the final GUI program interface, users only need to import different data sets that need to be used for calculations to get the final CityJSON file containing the building permits checking results, and they can view the final results in a flexible way. The working process of this tool is introduced in section 4.2.5.

4 Implementation

This chapter describes the implementation of doing building permission checking by using a CityJSON extension in details. The theory related to the topic has been introduced in Chapter 3. The working process of implementation shows in figure 4.1.

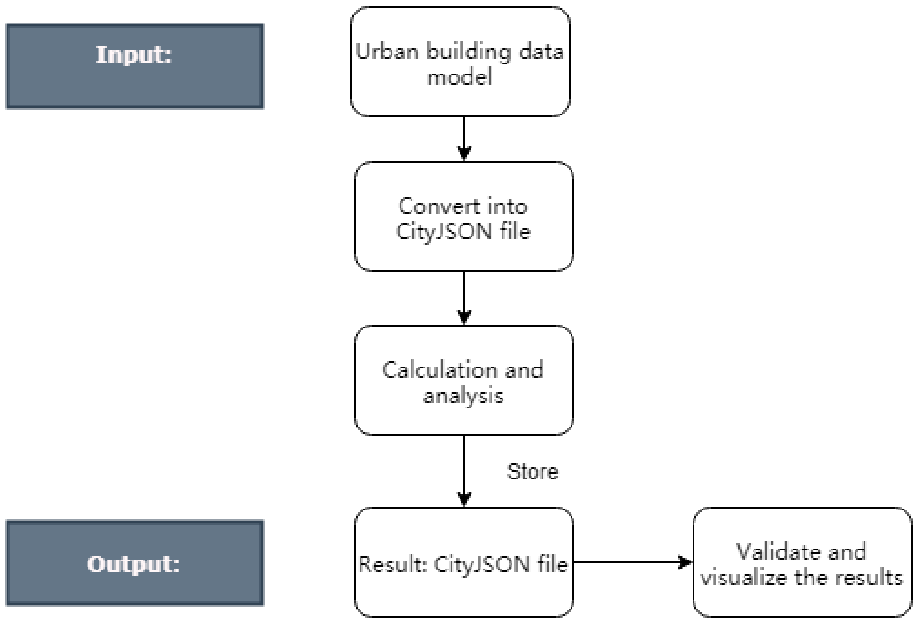

Input:
Urban building data model
Convert into CityJSON file
Calculation and analysis
Store
Output:
Result: CityJSON file
Validate and visualize the results

Figure 4.1: Workflow of the implementation process

The tools used to implement the methods are described in section 4.1, which includes the software, language, hardware, and datasets used in the experiments. Section 3.2 describes the use cases of 2 regulations to validate the methodology in details. The specifics of each phase in the implementation are detailed in section 4.2. The source codes is published on GitHub which available at `https://github.com/Gallon229/Building-permit`.

4.1 Tools and Datasets Used

4.1.1 Software

In this research, many tools will be used to read, process and validate 3D building information data and CityJSON files. The following softwares, tools and libraries is used to implement the methodology:

- Quantum GIS (QGIS)

  QGIS is a desktop GIS application that is free and open source. It is a powerful program for processing geographic data that provides display, editing, and analysis functions for geographic data. It has a number of functions that will aid in the data processing portion of this study. Simultaneously, it has a large number of plug-ins that can be downloaded to extend compatibility and perform more complex functions. The viewing and pre-processing of data will be carried in QGIS.

Toolkits for vector data processing will be frequently used, as well as plugins for viewing CityJSON files: CityJSON Loader, a simple QGIS plugin to load CityJSON files has been developed in Python [Ledoux et al., 2019].

- Python

  We complete the code implementation part of this research through the Python programming language. The first usage is to read the data of buildings and other features in datasets from different sources, write code to calculate and create GUI programs for building permit inspections. As for the coding part, Python will be used and some packages deals with geographical information system (GIS) geometries and attributes are included. For example, Fiona and Shapely. Calling functions of these packages can implement different tasks. The version of python used in this research is 3.7 and PyCharm is used as the IDE of python.

- Cjio and val3dity

  Since we need to convert the geojson file into CityJSON file and write the extension schema in CityJSON format, the data need to be validated by JSON schema validator. Validation of a CityJSON dataset means that one must ensure that it respects the standardized specifications and definitions as given in the CityJSON specification. Here cjio and val3dity are used for the validity check of syntax and geometric of datasets to ensure the CityJSON file meets the specification. Cjio is a python Command-Line Interface (CLI) to process and manipulate CityJSON files. Val3dity doing validation of 3D primitives according to the international standard ISO19107 [Ledoux, 2018]. They are developed by Hugo Ledoux, available at `https://github.com/cityjson/cjio` and `https://github.com/tudelft3D/val3dity`.

- Cesium and Cesium-cityjson

  The 3D visualization part can be implemented with cesium-cityjson that is a viewer for CityJSON and CityGML files based on Cesuim. Cesuim is an open framework for software applications that use 3D data. It can query the global 3D terrain and buildings, and quickly create 3D tiles. It is open source and can be used for accurate 3D visualization on the web. Currently, the Cesuim web page does not support uploading and manipulating CityJSON files. There is a tool generated with Angular CLI called cesuim-cityjson can solve this problem. With this tool, you can view the model on the web interface, and select and filter objects and attributes. They are available at `https://cesium.com/index.html` and `https://github.com/limyyj/cesium-cityjson`.

4.1.2 Hardware

The tests and all the implementation tasks are carried on a MSI Codex S 8RA-003XEU - Gaming Desktop that has the following details: Intel Core i5 processor at 2,8 GHz, video card of NVIDIA GeForce GTX 1050, 8 GB of Random Access Memory (RAM), 240 GB of SSD memory with the operating system of Window 10.

4.1.3 Datasets

The dataset used in this study is from BAG which is available at `https://www.kadaster.nl/zakelijk/registraties/basisregistraties/bag`. It is a register of buildings and addresses in the Netherlands which include the building footprints and some attributes. The BAG dataset contains information about all buildings in the Netherlands. It records the information of 5 types of objects: buildings, residential objects, number identification, public places and residential. These attributes include state, surface, geometric shape, xy coordinates, year of construction, and function of building. The free source of BAG GeoPackage is used here which is provided by GeoParaat. The data is collected in October of 2020. It can be found at `https://geoparaat.baasgeo.com`. Together with the 3D BAG, the building height information is available which is necessary for the construction of 3D building. The dataset is available at `http://3Dbag.bk.tudelft.nl/downloads`. The detailed rules on the implementation of the parking

	Information	Explanation	Sources	name in sources
7*Attributes	id	Bag id of building	BAG	identificatie
	+function	Function of buildings included in codelist	BAG	NR_XXX (different functions)
	+groundHeight	Elevation above sea level at the ground level	3D BAG	ground-0.00
	measuredHeight	Elevation above sea level at rooflevel	3D BAG	roof-0.75
	+zone	Zone where the building is located	Digital map	zone
	+height_valid	Indicate the height is valid	3D BAG	height_valid
	+total_area	Gross floor area (GFA) of building	BAG	Calculation results on different attributs
2*Geometry	type	geometry type of buildings	BAG	type
	coordinates	a lists contain [x,y,z] 3D coordinates	BAG	coordinates

Table 4.1: Information that need to be involve for building permission checking

regulation involve buildings with different functional attributes, so the OSM data is also in the datasets used in this research.

4.2 Overview of experiments

The use case regulations selected in this study aims to conduct building permit inspections on new buildings to be constructed. For new buildings, we need to create new datasets by ourselves, which contains all the attributes required for building permits. In addition, this research also extends the regulation to existing buildings. Although they have been constructed, we can filter buildings that meet the current parking regulations through our inspection. This result can be meaningful for other studies. Therefore, in this study, the old and new buildings were inspected separately in the test area.

4.2.1 Data pre-processing and input

Obtain information from datasets

First, we need to collect data. For old buildings, we can get building attributes and geometry information from BAG data and 3D BAG data. The BAG data and 3D BAG data cover many attributes. To speed up the operation process, we only need to extract the required attributes for completing the building permit inspection. According to the interpretation and formalization of regulations, we need to obtain the properties of the building as shown in the table 4.1. The building in BAG data and 3D BAG data can be joined together by the same id. To get the gross floor area (GFA) of a building, we need to access the components of the building, which are called "Rooms" in this study. They make up the entire building. The `"bag_snapshot_pand"` layer in the bag data represents the building body, and the `"bag_snapshot vbo"` layer represents the room object. The room object has the area of each room. The GFA can be obtained by adding up all the room areas that belong to the same building. The BAG data includes data of different administrative levels. We extract data with an administrative level equal to 8, which represents the municipality level. In connecting BAG data and 3D BAG data, it is found that the amount of 3D BAG data is less than that of BAG data. This is because the elevation data of some buildings are missing and the corresponding 3D data cannot be generated. Therefore, here we exclude buildings without elevation data.

Georeferencing the digital version of the zone classification map

The regulation specifies 3 zones of Rotterdam and provides a digital TIFF map for the zones. The rules of each zone are different, so we need to obtain the vector map of the partition as the input of our data. Through the location relationship between the building and each zone, confirm the zone where each building is located, and add it to the properties of the building. To get a vector file of zones and use it for further analysis, we need to do georeference on the raster map. First, using Ground Control Point (GCP) to do georeference in QGIS, then create a new vector file to store three zones together, finally output them as geojson files. The comparison between the original map and the vector zone file is shown in figure 4.2.

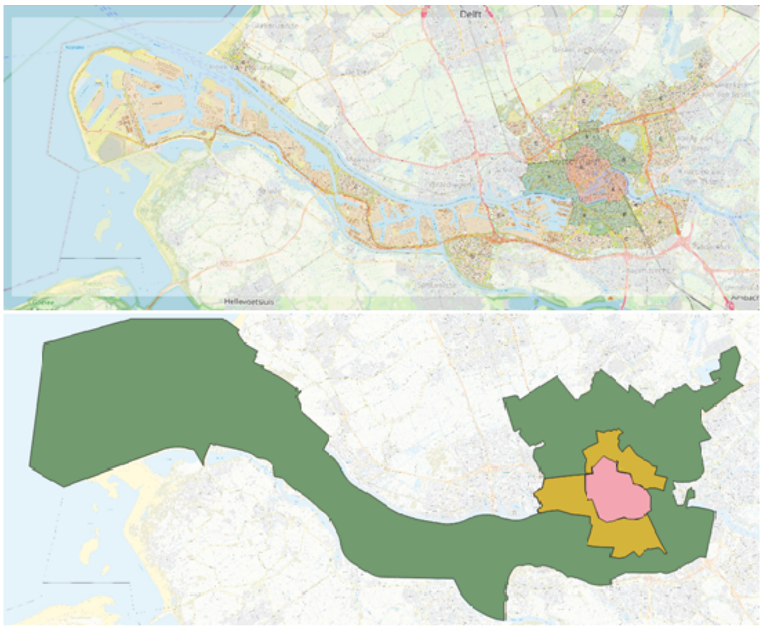


Figure 4.2: Georeferencing the digital map

Create new building datasets

To complete the building permit check for the new building, we need to manually create a file containing the new building. By creating a new vector file in QGIS, and drawing a new polygon we can get datasets for those new buildings. According to the attributes mentioned in the table 4.1, we need to add the corresponding attributes to the attribute list in the new building file and specify the attribute data type. We created two datasets for the new building, one is a building with a single function, and the other is a building with two functions. By generating arbitrary point elements in the polygon to represent parts of different functions, we can simulate buildings with multiple functions. In this process, we got a new 2D building by manually drawing polygons. To extrude it into 3D, we randomly assigned the number of floors to each building, according to the 3.5m height of each floor, and 70% of the ratio to get the GFA. The calculation formula is as follows:

1. new building

area = 3.5 * storeys numbers * footprint area * 70

2. multiple functions

create points in polygon and assign random area values to those points (using tool 'random points in polygon' in QGIS)then use these values to do the calculation

point1_area = 0.3* total area; point2_area = 0.4 * total area

Query OSM data

To understand and get the function of the buildings, in addition to referring to the attributes of the BAG data, we can also use the OSM data to obtain more information. In QGIS, different types of features can be obtained through the overpass query in a plugin called quickOSM. For the buffer analysis of traffic facilities, we need to query traffic-related features, such as bus stops and train stations. The overpass query for query traffic facilities is:

Query for transportation

```
<osm-script output="json" timeout="25">
  <union>
    <query type="node">
      <has-kv k="highway" v="bus_stop"/>
      <bbox-query {{bbox}}/>
    </query>
    <query type="node">
      <has-kv k="public_transport" v="stop_position"/>
      <bbox-query {{bbox}}/>
      </query>
        <query type="node">
      <has-kv k="public_transport" v="platform"/>
      <bbox-query {{bbox}}/>
      </query>
        <query type="way">
      <has-kv k="public_transport" v="platform"/>
      <bbox-query {{bbox}}/>
      </query>

  </union>
  <!-- print results -->
  <print mode="body"/>
  <recurse type="down"/>
  <print mode="skeleton" order="quadtile"/>
</osm-script>
```

For the function of auxiliary analysis of buildings, we need to query the types of features involved in this research, such as office, industry, shops. The query in QGIS for different functions of building is:

Query for different functions of building

```
CASE
        WHEN "vboheeft_kantoor_functie" = 'T'
        OR("vboheeft_kantoor_functie" is NULL
        AND "office_2" is not NULL) THEN 'Office'
        WHEN "vboheeft_industrie_functie" = 'T' THEN 'Industry'
        WHEN "vboheeft_winkel_functie" = 'T'
        AND "shop_2" = 'supermarket' THEN 'Supermarket'
        WHEN "vboheeft_winkel_functie" IS NULL
        AND"shop_2" = 'supermarket'THEN 'Supermarket'
        WHEN "vboheeft_winkel_functie" = 'T'
        AND "shop_2" is not NULL
        AND "shop_2" != 'supermarket' THEN 'Shop'
        WHEN "vboheeft_winkel_functie" is NULL
        AND "shop_2" is not NULL
        AND "shop_2" != 'supermarket' THEN 'Shop'
END
```

Because of the lack of data, for the old buildings, we only got four complete functional layers from QuickOSM, namely'Office','Industry','Retail', and'Supermarket'. Therefore, for non-residential buildings in old buildings, currently only buildings with the above four functions can be inspected for building permits.

Extract test area

After obtaining the required data, we need to specify the test area of this research. This is because the amount of data in Rotterdam is too large, and there are higher requirements for the computer to conduct the research. So we select a small part of Rotterdam to validate the methodology. The test area must cover three different zones to ensure the success of the research. Figure 4.3 shows the tests area of this research.

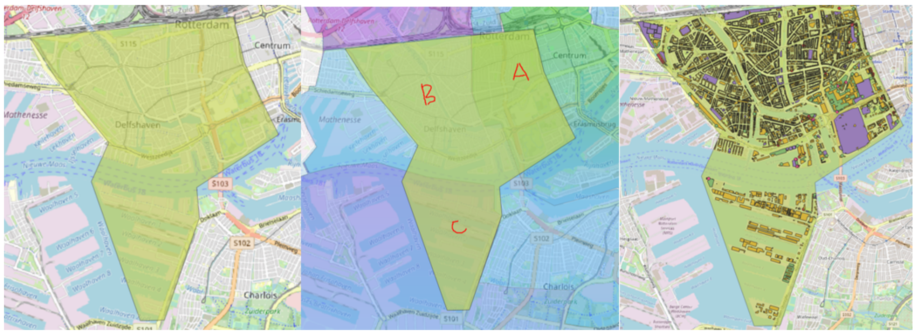


Figure 4.3: 1. Test area 2. Different zones of test area 3. The buildings and city objects located in test area

Buffer analysis on transportation

This analysis is based on the special exemptions from the parking requirement in the regulation. If development is realized in the vicinity of the public transport stations listed in the table B.1 below, the car parking requirement will be reduced in accordance with the table. We obtain traffic facility data through OSM and divide them into 10 different situations in accordance with the law. After analyzing the buffer zone and merging them, a map of the buffer zone of Rotterdam's transportation facilities can be obtained. By associating the buffer map and the building file in the geographic space, and selecting the association relationship as "within", the discount coefficient of the building can be obtained. During our inspection process, a new attribute "discount factor" will be added to the building for the next calculation. The working process of doing buffer analysis on transportation can be summarized as:

1. Get the layer which contains bus station, train station and parking lot

2. Using the criteria to do the buffer analysis on those facilities to get the result (which will show a map that places do not need to have parking spaces

3. join the two layers (to assign discount factor to building layer)

We can store the results of the buffer analysis as a premise, and multiply it by the corresponding discount percentage after calculating the minimum number of parking spaces in the building.

The result of buffer analysis is shows in figure 6.1. Through the discount factor, the number of parking spaces can be reasonably designed to maximize the use of space.

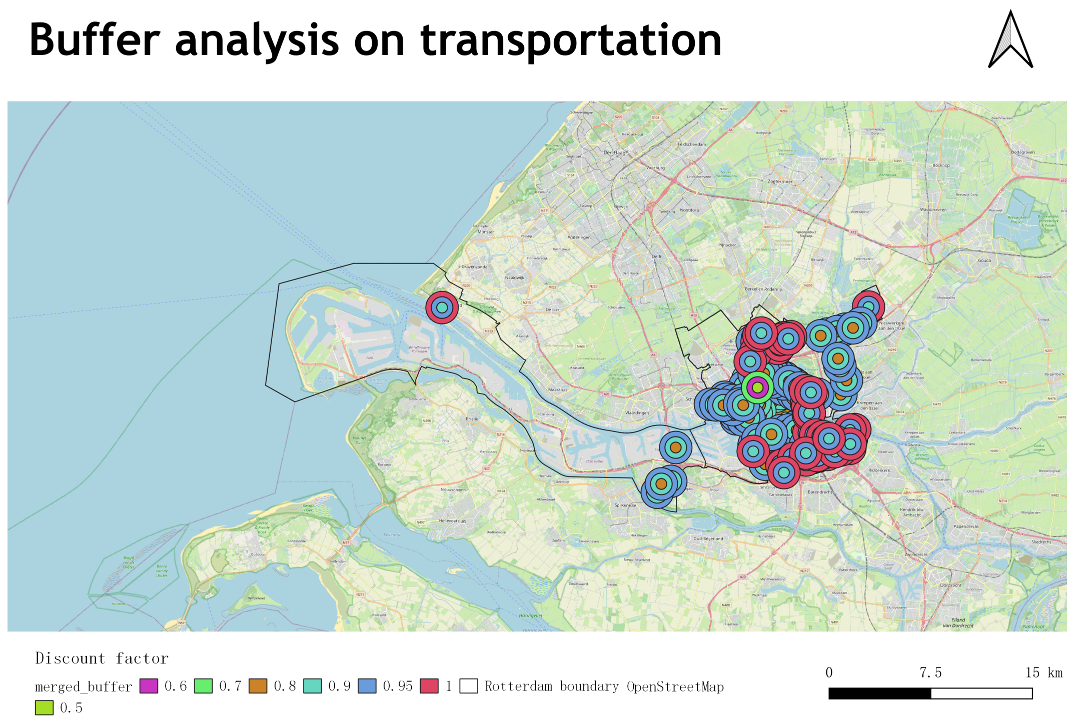

Buffer analysis on transportation
Discount factor
merged_buffer 0.6 0.7 0.8 0.9 0.95 1 Rotterdam boundary OpenStreetMap
0.5
0 7.5 15 km

Figure 4.4: Buffer analysis on transportation

4.2.2 Bicycle parking regulation (For existing buildings and new buildings)

Through python programming, we can complete the calculation by computer. The idea of calculating the minimum parking space of the new building and the minimum parking space of the old building is the same, but they have different ways of obtaining attributes. For old buildings, we get the corresponding attributes from BAG and 3D BAG data. For new buildings, we directly get the attributes from the new vector file.

1. Residential buildings:

For residential buildings, we count the number of rooms of different areas, calculate the results according to the formula.

Key part of code: the calculation process for residential buildings

```
N_40 = int(f['properties']['N_40'])
        N_40_65 = int(f['properties']['N_40_65'])
        N_65_85 = int(f['properties']['N_65_85'])
        N_85 = int(f['properties']['N_85'])
        oneb['attributes']['+min_bicycle_parking_spaces'] = int(
            N_40 * 2 + N_40_65 * 3 + N_65_85 * 0.4 + N_85 * 5)
```

2. Non-residential buildings:

For non-residential buildings, according to the function classification of the building, we obtain the minimum number of bicycle parking spaces per 100 square meters of building area according to the table mentioned in regulation.

Key part of code: the calculation process for non-residential buildings (e.g office function)

```
            if oneb['attributes']['+function'] == 'Office':
                    oneb['attributes']['+min_car_parking_spaces'] = 1.7
```

4.2.3 Car parking regulation (For existing buildings and new buildings)

Same as mentioned in the implementation of bicycle regulation, the idea is same for old buildings and new buildings related to the car parking regulation. Calculating car parking spaces is more complicated, because we need to take into account that the zone where the building located.

1. Residential buildings:

For residential buildings, we count the number of rooms of different areas, calculate the results according to the formula.

Key part of code: the calculation process for residential buildings

```
N_40 = int(f['properties']['N_40'])
N_40_65 = int(f['properties']['N_40_65'])
N_65_85 = int(f['properties']['N_65_85'])
N_85_120 = int(f['properties']['N_85_120'])
N_120 = int(f['properties']['N_120'])
if f['properties']['zone'] == 'A':
    oneb['attributes']['+min_car_parking_spaces'] = int(
    N_40 * 0.1 + N_40_65 * 0.4 + N_65_85 * 0.6 + N_85_120 * 1 +
    N_120 * 1.2)
if f['properties']['zone'] == 'B':
    oneb['attributes']['+min_car_parking_spaces'] = int(
    N_40 * 0.1 + N_40_65 * 0.5 + N_65_85 * 0.8 + N_85_120 * 1 +
    N_120 * 1.2)
if f['properties']['zone'] == 'C':
    oneb['attributes']['+min_car_parking_spaces'] = int(
    N_40 * 0.1 + N_40_65 * 0.6 + N_65_85 * 1.4 + N_85_120 * 1.6
    + N_120 * 1.8)
```

2. Non-residential buildings: For non-residential buildings, according to the function and zone classification of the building, we obtain the minimum number of car parking spaces per 100 square meters of building area according to the table mentioned in regulation.

Key part of code: the calculation process for non-residential buildings (e.g office function)

```
if oneb['attributes']['+function'] == 'Office':
    if f['properties']['zone'] == 'A':
        oneb['attributes']['+min_car_parking_spaces'] =
        round(0.76 * oneb['attributes']['+total_area'] / 100)
    elif f['properties']['zone'] == 'B':
        oneb['attributes']['+min_car_parking_spaces'] =
        round(1 * oneb['attributes']['+total_area'] / 100)
    else:
        oneb['attributes']['+min_car_parking_spaces'] =
        round(1.2 * oneb['attributes']['+total_area'] / 100)
```

4.2.4 Validation and Evaluation

After we get the inspection result and store it in a CityJSON file, we need to check the validity of the file. It includes both syntax and geometry checks. The verification process will be completed through cjio and val3dity. The verification result needs to show that the file is valid. We can query the tools to do the checking by the following commands:

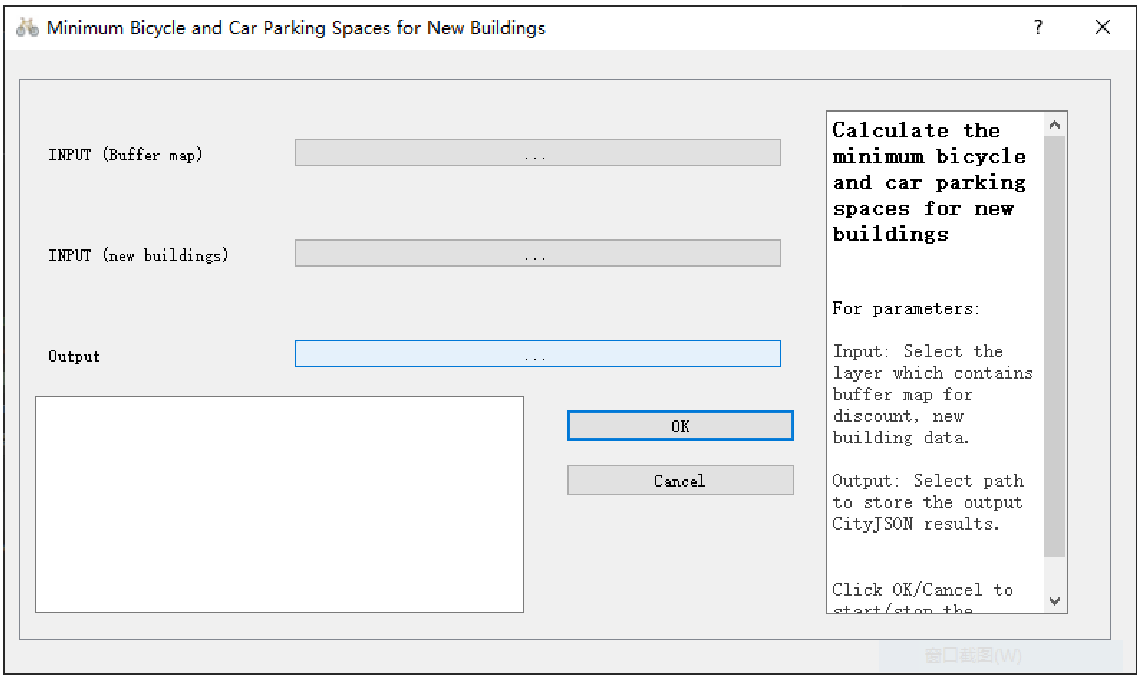
commands for validation

```
cjio path of the file validate --folder_schema path of extension

val3dity path of the file
```

4.2.5 GUI program application

To reproduce the experimental results, we developed a GUI program as a tool for the automation of building permit checking. In this program, the user only needs to provide the input data and the specified output file path and it will automatically complete the entire building automation inspection process. During the process, the corresponding attributes are extracted from the input datasets, the minimum car parking space and the minimum bicycle parking space are calculated according to the formalized results of the regulation, and finally they are stored in a file of CityJSON format. The program will also validate the file. Finally, the output result can be visualized through Cesium-cityjson or open in QGIS. The UI of this program is shows in figure 4.5.


Minimum Bicycle and Car Parking Spaces for New Buildings

INPUT (Buffer map) ...

INPUT (new buildings) ...

Output ...

OK

Cancel

Calculate the minimum bicycle and car parking spaces for new buildings

For parameters:

Input: Select the layer which contains buffer map for discount, new building data.

Output: Select path to store the output CityJSON results.

Click OK/Cancel to start/stop the

Figure 4.5: Components of UI

For the work steps mentioned above, the specific steps of regulation implemented in the program can be summarized as the following 7 steps:

1. User will input the datasets which is needed for the calculations

2. The computer filters buildings based on the building function's properties.

3. The computer check the exemptions cases, accumulates the total living area of the building through the room area information and check if it is in the traffic station buffer map.

4. According to the formula, calculate the minimum bicycle parking space for each building.

5. According to the formula, calculate the minimum car parking space for each building.

6. Get the calculation results, and the computer stores the results in a new CityJSON file with extensions.

7. Display the results in visual form, provide the calculation results of the number of bicycle parking spaces as a reference or compare with the actual situation, and complete the building permit inspection.

To save and shorten the calculation time, we need to set the order of code execution in the program. The inspection of parking regulation includes several conditions. Write them as statements to determine whether the conditions are true or false. If the previous one is satisfied, you can jump in the next conditional judgment. In the parking regulation, there are exemptions for small projects, so the area of the building is judged first, and if the parking space exemption is met, the subsequent calculations can be omitted.

5 Experiments and Results

In this chapter, the CityJSON extension data model of this research and the results of the parking regulation permit checks of the building will be presented.

5.1 The developed CityJSON extension

At present, the conceptual model of CityJSON urban building extension has been built as part of the results. The 3D model is based on the Rotterdam parking standard, so it is based on this use case. The Unified Modeling Language (UML) diagrams shows how to extend city objects in the data model used in this study. The corresponding extension schema is also completed. Figure 5.1 shows the UML diagram of the framework of the CityJSON extension. It is developed for urban planning especially with regard to building permit checks. In this study, we construct a 3D city model in CityJSON format with 4 City Objects. As a result, we have completed the extension of the 4 City Objects: building, room, landuse, and regulation. Among them, buildings and landuse are existing objects in the CityJSON model, so we extend them by adding new attributes to their existing models. Rooms and regulation are objects that do not exist in the CityJSON model, so we create corresponding new City Objects for them and add them to the data model, and then add attributes and geometric information to them. In the data model, the attributes and geometric types that each city object should contain are formulated. Therefore, all files and objects involved in analysis and calculations need to meet the standards given by the city model and follow the rules of the CityJSON extension mechanism to ensure data interoperability.The complete schema and extension validity results can be found at appendix B.3 and figure B.1.

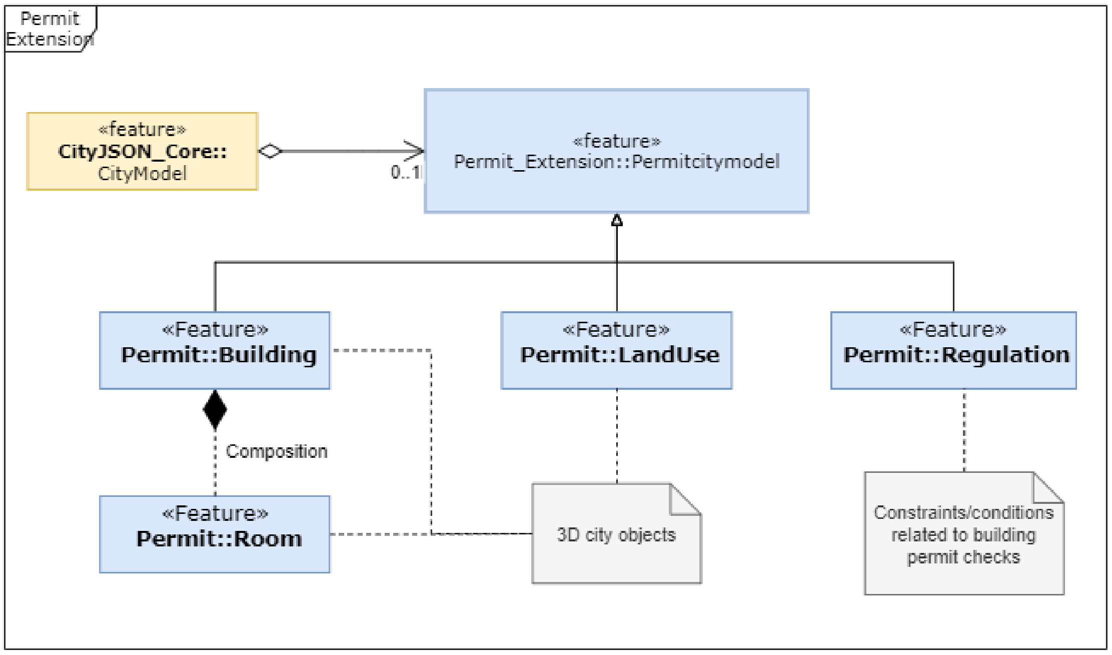


Figure 5.1: UML diagram of the CityJSON extension

Through the interpretation and translation of the Rotterdam parking standards, we found that the detailed information about the research subject which are building, room, and the zoning involved are

Table 5.1: The meaning of extended building attributes

Attributes	Meaning	Source
+permit::non_residential	Whether the building is a residential building. Its value range is 0 to 2. 0 - residential building 1 - non-resident building with a single function 2 - non-resident building with a dual function	BAG
+permit::groundHeight	Height above ground level	3D BAG
+permit::total_area	Gross floor area (GFA) of building	BAG
+permit::discount_factor	Parking quantity discount	Calculated results
+permit::min_bicycle_parking_spaces	Minimum parking spaces for bicycles	Calculated results
+permit::min_car_parking_spaces	Minimum parking spaces for cars	Calculated results
+permit::function1	The first function of the building	BAG
+permit::function2	The second function of the building, if not, it can be ignored	BAG

necessary information for permitting. Therefore, in the following UML module, we define the additional information of the City Objects necessary for the parking standard permit checks as the attributes of the City Objects, and specify the coding format of each attribute. Next is the detailed information explanation of each module:

- Extension of Building.

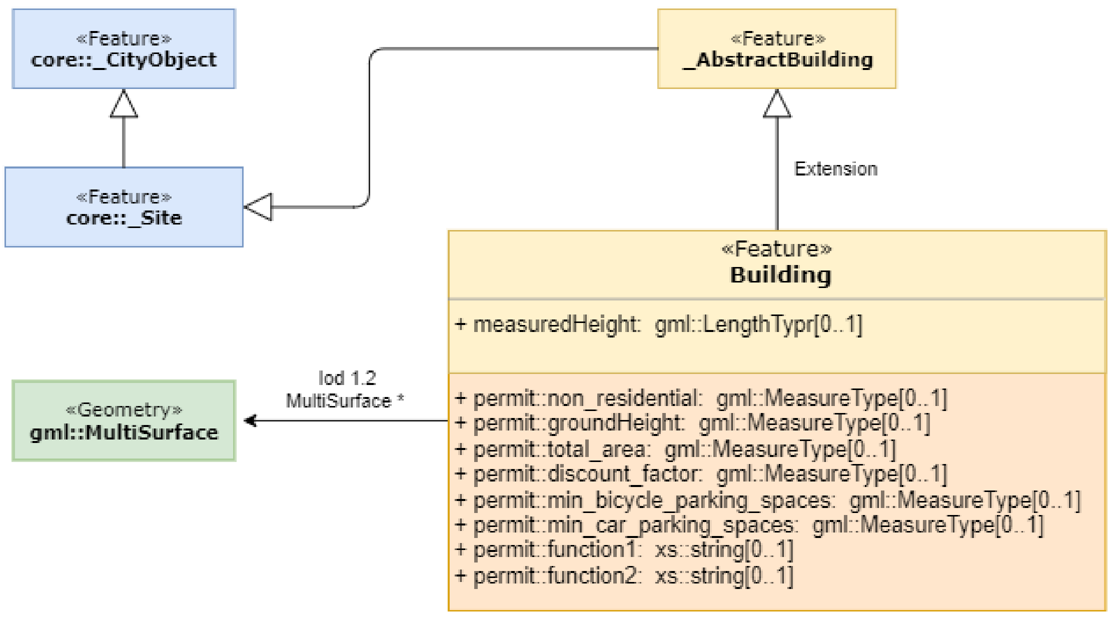

«Feature»
core::_CityObject
«Feature»
_AbstractBuilding
Extension
«Feature»
core::_Site
«Feature»
Building
+ measuredHeight: gml::LengthTypr[0..1]
«Geometry»
gml::MultiSurface
lod 1.2
MultiSurface *
+ permit::non_residential: gml::MeasureType[0..1]
+ permit::groundHeight: gml::MeasureType[0..1]
+ permit::total_area: gml::MeasureType[0..1]
+ permit::discount_factor: gml::MeasureType[0..1]
+ permit::min_bicycle_parking_spaces: gml::MeasureType[0..1]
+ permit::min_car_parking_spaces: gml::MeasureType[0..1]
+ permit::function1: xs::string[0..1]
+ permit::function2: xs::string[0..1]

Figure 5.2: UML diagram of Building

This module contains the basic detailed information of buildings that are necessary for the analysis of building permission checks in urban areas and extends the existing modules. The additional information is related to parking standard or stores some analysis results. Figure 5.2 shows the UML model of the building module, here the building is extended by directly adding new attributes with prefix '+permit'. Attribute 'measuredHeight' is the measured relative height of the building that already defined in the model. Here, according to the formalization of parking standard, we add 8 new attributes to 'Building'. The meaning of each attributes is described in table 5.1.

- Extension of Room.

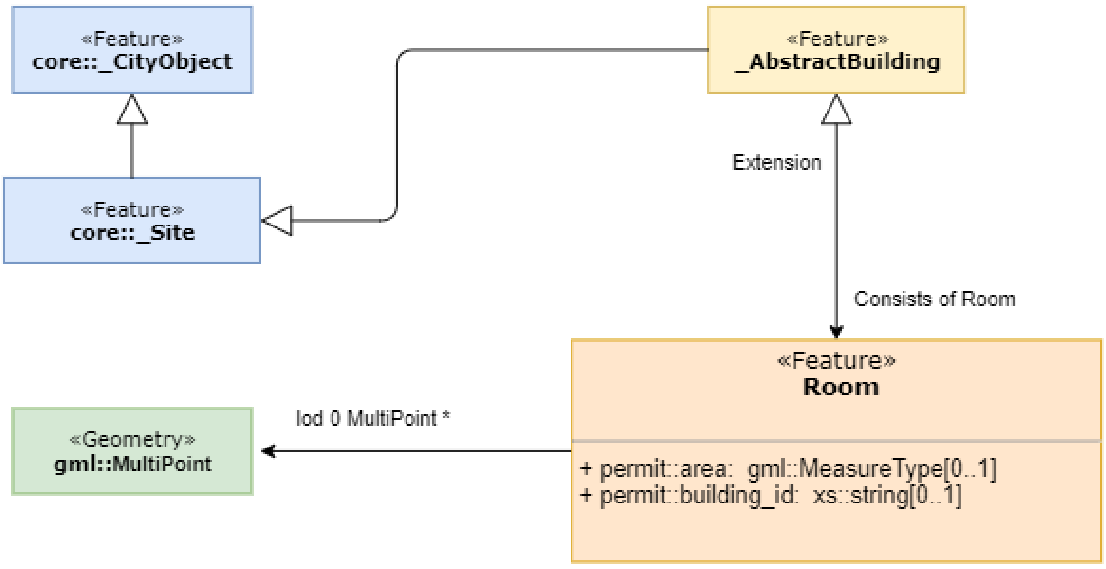


Figure 5.3: UML diagram of Room

The BAG data contains information about different rooms/accommodations in the building, such as area and coordinates. These can be useful when conducting permit checks that take into account rooms in a building. In the existing City Objects, BuildingPart is very close to this concept. But the geometric definition of BuildingPart requires that it should be Solid, CompositeSolid, or MultiSurface. Limited by the experimental data, it was decided to store this information by creating/extending a new City Object to this data model. This new City Object is called 'Room' as figure 5.3 shows, which inherits from `core:: _cityObject` and contains unique id information, the area of room area, and the corresponding building id information. The geometry is defined as lod 0 multipoint by its two-dimensional coordinates. Therefore, the address data in BAG can be used for creating this new City Object.

- Extension of Landuse.

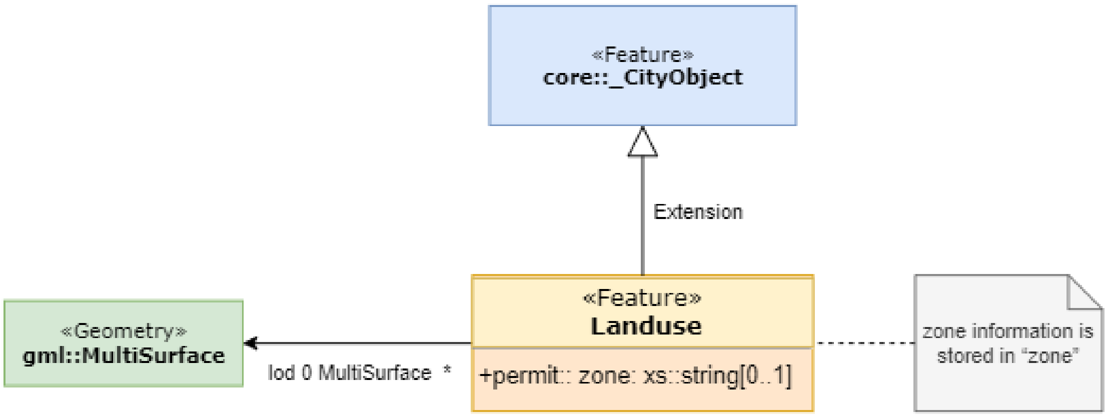


Figure 5.4: UML diagram of Landuse

For CityObject 'LandUse', we extend it by adding a new attribute to the data model. The attribute '++permit:: zone' refers to zoning information in the parking standard regulation. We store zoning information in LandUse. When calculating the number of parking spaces for a building, we need to first determine the zone where the building is located. The geometry is defined as lod 0 MultiSurface by the two-dimensional coordinates of polygons. Figure 5.4 shows the landuse module.

Table 5.2: The meaning of extended regulation attributes

Attributes	Meaning
+permit::Validity	Document whether the regulation is in effect.
+permit::enactmentDate	Document the date the law came into effect, if have.
+permit::expirationDate	Document the date the law expires, if have.
+permit::adminLevel	Among(municipality, province, country).
+permit::class	The legal category to which it belongs, which can be of spatial planning; Environment; Tranport;
+permit::function	Usage of the regulation (e.g Calculate the appropriate parking spaces for each building)
+permit::Administration_area	Where does it apply

- Extension of regulation.

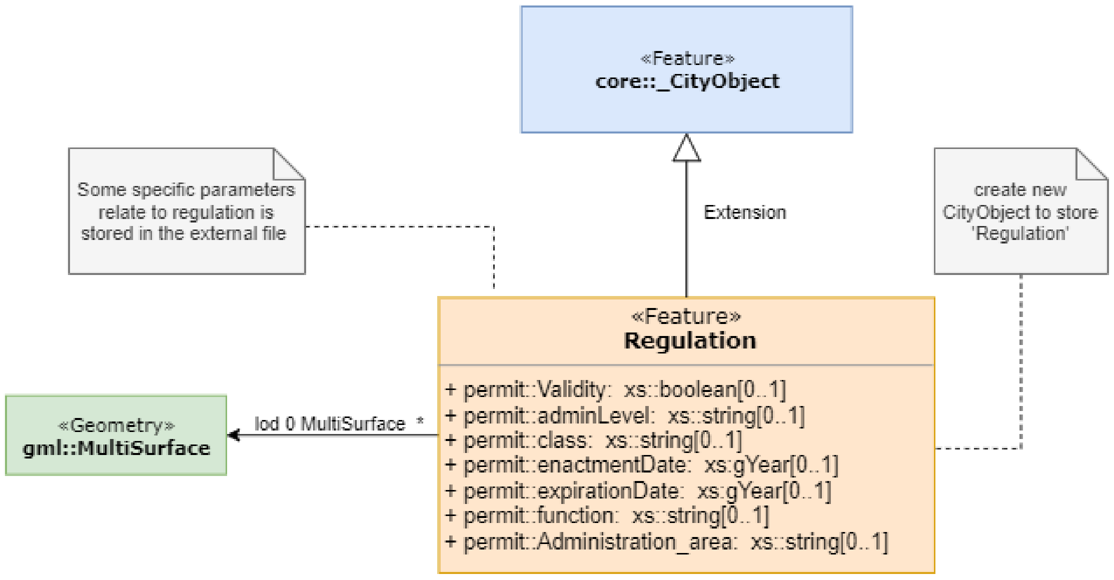

«Feature»
core::_CityObject
Some specific parameters relate to regulation is stored in the external file
Extension
create new CityObject to store 'Regulation'
«Feature»
Regulation
+ permit::Validity: xs::boolean[0..1]
+ permit::adminLevel: xs::string[0..1]
+ permit::class: xs::string[0..1]
+ permit::enactmentDate: xs:gYear[0..1]
+ permit::expirationDate: xs:gYear[0..1]
+ permit::function: xs::string[0..1]
+ permit::Administration_area: xs::string[0..1]
«Geometry»
gml::MultiSurface
lod 0 MultiSurface *

Figure 5.5: UML diagram of Regulation

This study selects the parking standard regulations applicable to Rotterdam city, and conducts the inspection of building permits according to its criteria.

These constraints and conditions have an impact on the analysis and evaluation of buildings and other City Objects, so they also need to be recorded in the data model, that is, the conditions involved in the regulation, and the parameters will be stored. "Regulation", which is also inherited from `core:: _cityObject`, as shown in Figure 5.5. We defined what needs to be stored in the regulation, they are basic and important attributes. The description of each attribute can be found at table 5.2. Here the data model is extended by creating a new City Object. This model also applies to other regulations. But every regulation has its specific rules and constraints, which may not be expressed through attributes. To help users understand what we did with the City Object 'regulation', we constructed an external file which is a specification/guideline that documents the regulation in natural language, and we store the results of interpretation and formalization of the regulation. The specification can be found at B.1.

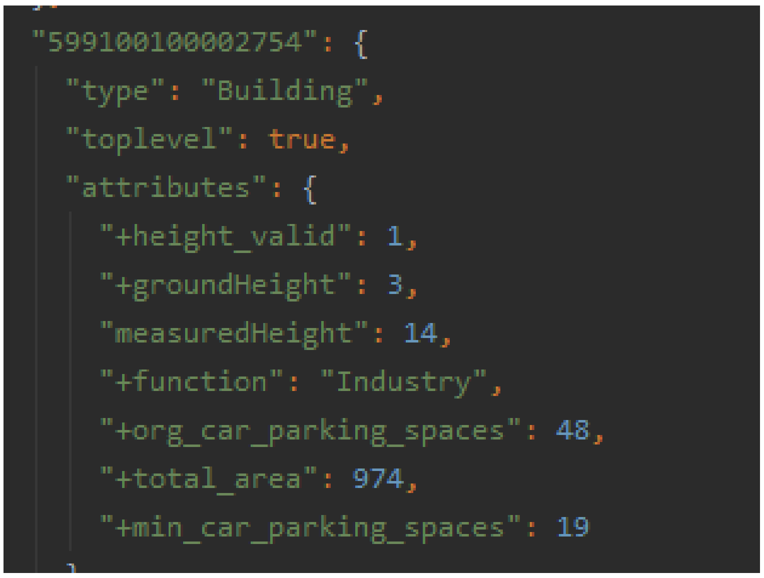
After we have constructed the data model, we create the extension schema to document how the core data model of CityJSON is extended by following the rules of CityJSON. We then use a validation script to validate the schema file. After that, we can validate CityJSON files that containing our extensions. The complete schema and extension validity results can be found at appendix B.3 and figure B.1.

5.2 Results of building permit checking of parking regulation

This research adopts the research method of case study, so the results will be displayed and discussed based on the case. Because BAG and 3D BAG have a large amount of data in Rotterdam, considering the performance of the equipment, we selected a small test area for inspection which shows in figure 4.3.

Building permit checks of old buildings

For old buildings, we developed a small calculation script for them. Table 5.3 shows the datasets chosen in this research. We will use examples of non-residential buildings in old buildings. We combined the function attributes in the OSM and BAG data to filter out four types of buildings with functional purposes. They are Office, Industry, Retail, Supermarket. Then, read the file containing these building information and add new attributes to them according to the extension. Through the corresponding room area of each building, we can get the gross floor area of the building. When the gross floor area of non-residential building is less than 600 square meters, the number of parking requirements can be exempted. Next, the building function is judged, and the four different types of buildings are calculated according to the formula, and the final result is obtained. Due to the lack of data, it is difficult to obtain the original number of parking spaces in old buildings. In order to compare the data we calculated, we randomly set the attribute '"+org_car_parking_spaces' to each building, which is used to assume the number of parking spaces in the building. Figure 5.6 shows the calculation results of car parking spaces of the example above which store in a CityJSON file. The attributes suggest it is an industrial building of area 974 square meters. The number of original parking spaces is randomly allocated of 48, and the minimum number of parking spaces we calculated for the building is 19. It satisfies the condition that the number of original parking spaces is greater than the minimum number of parking spaces, so the building meets the building permit checks of parking standards. Figure 5.7 shows the results displayed in QGIS using the plugin 'Load CityJSON'. You can click on one specific building to access its attributes which are the same as the figure 5.6.



```
"599100100002754": {
  "type": "Building",
  "toplevel": true,
  "attributes": {
    "+height_valid": 1,
    "+groundHeight": 3,
    "measuredHeight": 14,
    "+function": "Industry",
    "+org_car_parking_spaces": 48,
    "+total_area": 974,
    "+min_car_parking_spaces": 19
```

Figure 5.6: Result of an industrial building

Table 5.3: Overview of datasets used in the research

File	File Size(MB)	Characteristics	Number of City Objects	Number of City Objects in test area	File size of test area(MB)	Source
BAG	550.4	Building	402412	13647	38.5	geoparaat
BAG	727.4	Room	760394	24194	35.3	geoparaat
3D BAG	644.6	Building	368590	11229	38.5	tudelft3D
OSM	2.2	Building	12642	414	1.2	OSM

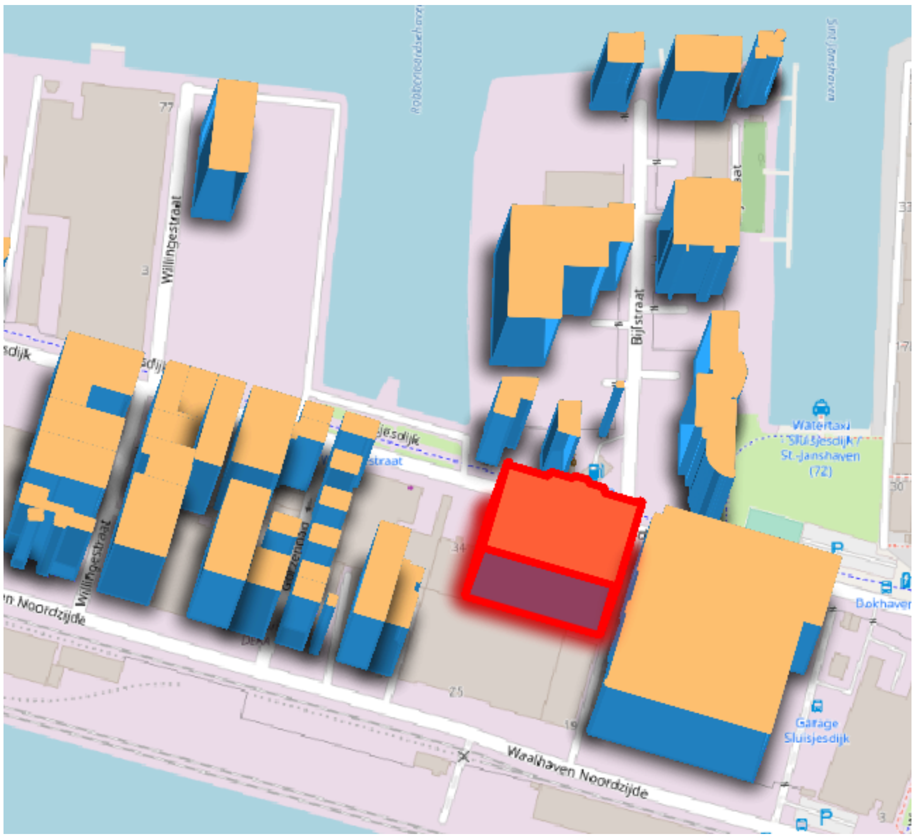
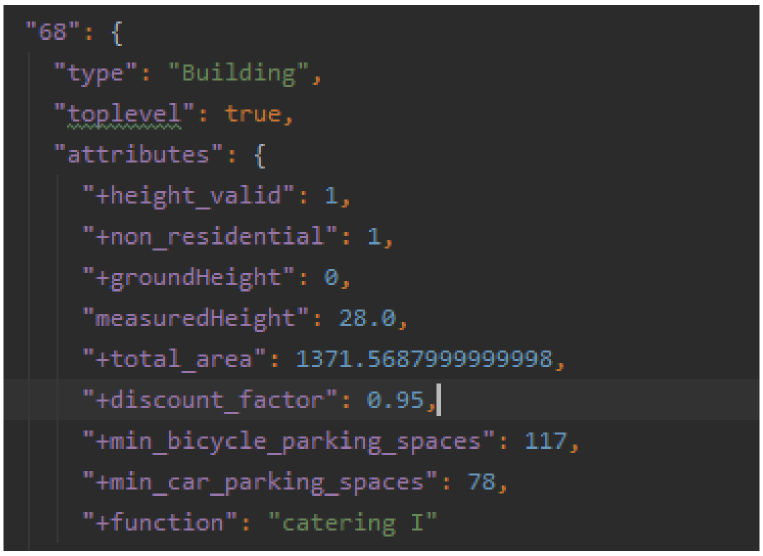


Figure 5.7: Load the results in QGIS

Building permit checks of new buildings

The main purpose of our research is to conduct permit inspections for newly constructed buildings and provide results for reference. In this research, we created new building datasets and assign attributes to them. In the real situation, the data regarding the new buildings are likely supposed to come from the conversion of BIM delivered as part of a digital building permit from the building developer. The attribute `non_residential` indicates the function and purpose of the building. 0 represents a residential building, 1 represents a single-function non-residential building, and 2 represents a multi-function non-residential building. The attribute `discount_factor` indicates the coefficient of the parking quantity discount that the building can get after the traffic buffer analysis. Then the attributes `min_bicycle_parking_spaces` and `min_car_parking_spaces` respectively represent the results of the calculated minimum parking quantity. We created 100 new buildings in QGIS, each of which has one or two different functions (randomly assigned from a codelist containing 12 building functions). Then we calculate differently according to three situations. The first is a single-function residential building, the second is a non-residential building with any single function, and the third is a random combination of two non-residential functions. In this research, we created new building datasets and assign attributes to them. In the real situation, the data regarding the new buildings are likely supposed to come from the conversion of BIM delivered as part of a digital building permit from the building developer. The attribute `non_residential` indicates the function and purpose of the building. 0 represents a residential building, 1 represents a single-function non-residential building, and 2 represents a multi-function non-residential building. The attribute `discount_factor` indicates the coefficient of the parking quantity discount that the building can get after the traffic buffer analysis. Then the attributes `min_bicycle_parking_spaces` and `min_car_parking_spaces` respectively represent the results of the calculated minimum parking quantity. In the code, we first calculate the "discount coefficient" of each building. This is done by judging the relative position of the different discount areas in the building and the buffer map, using the sjoin function in geopandas in python to complete the discount coefficient calculate. Then, calculate the results according to the formulas for the buildings in three different situations. The calculation basis for 12 different functions comes from the table B.3.

Here, use one new building as example. Figure 5.8 shows the calculation results of car parking spaces based on test data which store in a CityJSON file. The attributes suggest it is a non-residential building of 'catering I' use, its area is about 1372 square meters. It is calculated that its discount factor is 0.95. Therefore, the calculated number of parking spaces for bicycles and cars is the result of multiplying by 0.95. It shows that the minimum number of bicycle parking spaces is 117 and the minimum number of car parking spaces is 78. Therefore, as long as the number of parking spaces actually built is greater than these two numbers, it can pass the parking standard building permit checks. Figure 5.9 shows the results displayed in QGIS using plugin 'Load CityJSON'.



```
"68": {
  "type": "Building",
  "toplevel": true,
  "attributes": {
    "+height_valid": 1,
    "+non_residential": 1,
    "+groundHeight": 0,
    "measuredHeight": 28.0,
    "+total_area": 1371.5687999999998,
    "+discount_factor": 0.95,
    "+min_bicycle_parking_spaces": 117,
    "+min_car_parking_spaces": 78,
    "+function": "catering I"
```

Figure 5.8: Result of an new building of catering use

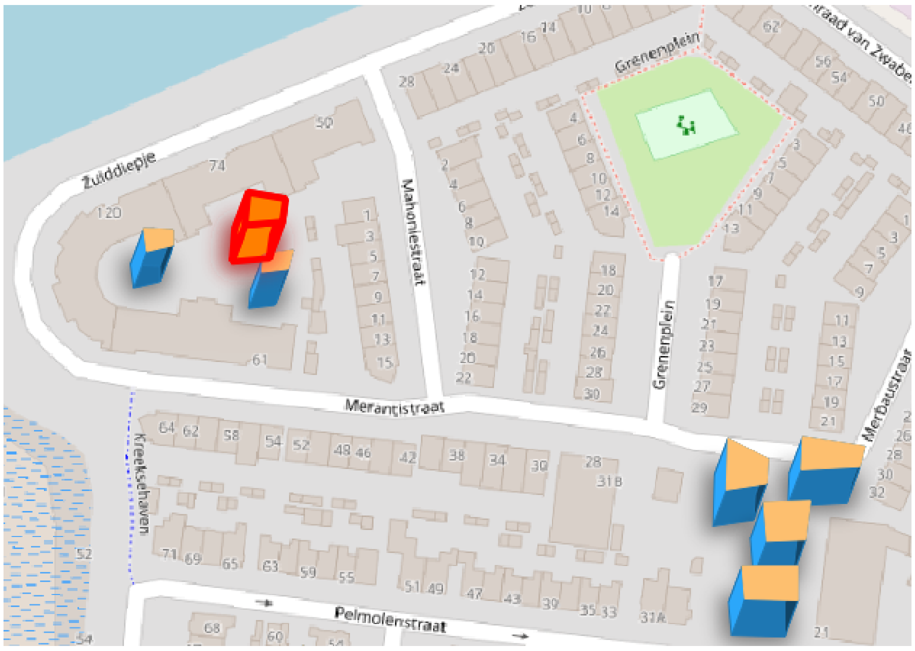


Figure 5.9: Load the results of new buildings in QGIS

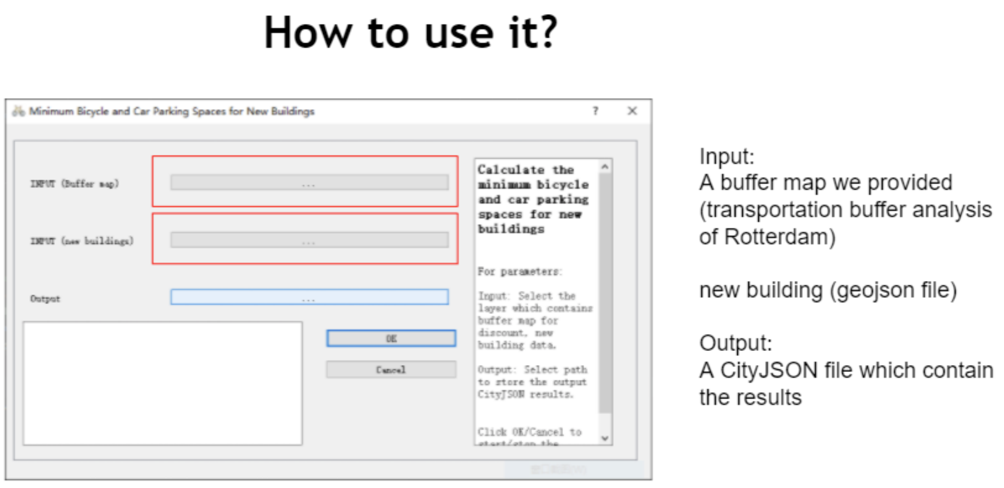
5.3 Demo of tool

After we completed the main code for calculating the number of parking spaces in the building, we combined it with the developed GUI program. It constitutes a tool that can calculate results through a graphical interface program. This will make it easier for those who don't know much about the code to reproduce our calculation results. A video demo of the GUI program used to conduct automated building permit inspections can be found here: `https://github.com/Gallon229/Building-permit`. It shows the whole process for doing builidng permission checks by this tool.

5.4 Tool performance

The expected audience of this research is building developers, designers and government officials, who can use the developed GUI tools to check the building permits of the parking standards of the new buildings. This tool allows them to reproduce the results of this experiment. As shown in the figure 5.10, they only need to specify three parameters to run the tool. These tools are respectively the buffer map of the specified discount factor generated during our experiment (the map will be placed in the same folder as the prerequisite and the tool)), a file containing the geojson format of the new building, and specify an output file path. Click 'OK' to run the program, and you can get the building permit inspection CityJSON file containing the parking standards for the new building also the validation results of the file. Finally, the result is obtained by reading the `min_bicycle_parking_spaces` and `min_car_parking_spaces` in the file.

The results can be used for reference to architectural design, and to participate in the process of architectural design. This result can also be used as a reference for government officials, indicating the minimum number of parking required for the building to obtain a building permit. The results show that the number of parking spaces in a building is affected by many parameters, such as the area of the building, the function of the building, the area where the building is located, the distance between the building and the transportation facilities, and the number of functions the building has. The parking standard law states that the number of parking given in accordance with such rules is the result that can best promote the healthy development of the city and facilitate the lives of the people. Therefore, the results given by this tool will help construction developers and government officials.


How to use it?

Minimum Bicycle and Car Parking Spaces for New Buildings

INPUT (Buffer map)

INPUT (new buildings)

Output

Calculate the minimum bicycle and car parking spaces for new buildings

For parameters:

Input: Select the layer which contains buffer map for discount, new building data.

Output: Select path to store the output CityJSON results.

Click OK/Cancel to start/stop the

Input:
A buffer map we provided (transportation buffer analysis of Rotterdam)

new building (geojson file)

Output:
A CityJSON file which contain the results

Figure 5.10: How to use the tool by building developers or government officials

6 Discussion

In this chapter, a discussion based on the results will be provided in different aspects.

6.1 Limitation

The method we proposed aims to complete the building permits checks related to some regulations digitally. However, it still has some limitations. First of all, there are fewer data sources, especially new buildings, which results in fewer types of building conditions that can be displayed (for example, for non-residential buildings, it may have a combination of multiple functions). For old buildings, when comparing their calculation results, they lacked data on the number of their original parking spaces, so we can only assume an original number of parking spaces by random assignment. If there are data on the number of parking spaces in some old buildings, our tests in this section will be more convincing.

In addition, the automation of the inspection process needs to be improved. At present, the GUI program can be used to complete the inspection of the parking standard of the new building, but the user needs to input the specified file for the program to calculate. In the future, we will continue to explore how to use this tool function. More powerful functions to achieve richer visual rendering.

This study adopts the user case method and only studies the Rotterdam parking regulation, so it can only test the application of this research method to parking law. At the same time, due to the limitation of the performance of computer equipment, the running speed and running ability of the program need to be improved.

In this study, because of the lack of data for the new building under design, we manually created the layer in QGIS and completed the input of the new building data through the code assignment of attributes. The advantage of this method is that it is relatively simple, you only need to specify Some attributes related to the law do not need to consider the shape modeling of the building. The disadvantage is that it simplifies the shape of the building. For some more complex building structures, the data may not be very close to the actual situation of the building. In the future work, we expect to obtain the model of the new building from the construction developer, which can be in BIM format or 3D model format. With these models, subsequent calculations will be more accurate.

6.2 Contributions and application

Our work is most relevant to the work of building permit checks of the 3D geoinformation group at TU Delft and a CityGML urban planning ADE.

In Noardo et al. [2020b], they checked the building permit through GeoBIM, which combines geographic information with BIM. In their study, two recently designed buildings in the center of Rotterdam were selected for relevant legal inspections. Their research also involves parking laws, but only covers the inspection of the number of car parking in residential buildings. Our research extends the scope to non-resident buildings and the number of bicycle parking inspections. Therefore, we have more detailed and comprehensive analysis on parking laws. In the research of this study, we created a new building dataset to test the feasibility of the method which provides more data for testing. Besides, we built a 3D city model for the building permit inspection process, which can store more city information and has stronger interoperability than BIM used in Noardo et al. [2020b].

In research of Akahoshi et al. [2020], they constructed the CityGML ADE architecture related to urban planning, which covers the City Objects involved in urban planning in the model, and can complete analysis related to urban planning. Our research adopts the use of CityJSON extension to extend the 3D city model. It has a simpler structure and a more flexible mechanism, so it can better interoperate and integrate with other data formats. Although CityGML ADE can support inheritance and namespaces, its flexibility may bring troubles for processing software. To make it work properly in the software, specific code needs to be written. While CityJSON extensions usually do not need to change the parsing code, because they can be treated as standard CityJSON files, so software that supports CityJSON can read files containing CityJSON extensions normally. This provides convenience for analysis and data processing beyond the extension.

The method proposed in this study uses a 3D city model to complete an automated inspection of building permits of parking standards and develops corresponding tools to reproduce the automated process. The models and tools developed in this research can be used in more detailed regulation inspections, such as the parking standards in Rotterdam that appeared in this research. When users want to apply our tools to other cities or other regulations, they can use the CityJSON extension data model we built. If it is the parking standards of different cities, then more attention needs to be paid to the acquisition of building data in that city. If it is a different law, then through the interpretation and formalization of the law, we need to summarize the City Objects and attributes that need to be added to the CityJSON extended model. In subsequent work, this method can be applied to more regions and different regulations to promote the healthy development of the city through efficient and high-quality building permit inspections. It can provide support for government employees, which saves time and manpower compared with traditional building permit inspections as shows in figure 6.1. In the rich picture, we researcher aims to develop a tool to automate the building permit checks process by constructing a 3D city model and interpret and formalize the regulation. By achieving these goals, we developed a tool that can calculate the minimum parking spaces and give the results of the checks of buildings. By using our tool, the building developer can know if the building is meeting the requirements then made some modifications to the design. On the other hand, government officials can use this tool to get the permit checks results and assist their decision making. The 3D city model constructed among them is not only used for building permit inspections, but also for other applications related to city analysis. For example, building energy analysis, sunlight analysis, etc., can be expanded into a more comprehensive urban planning model.

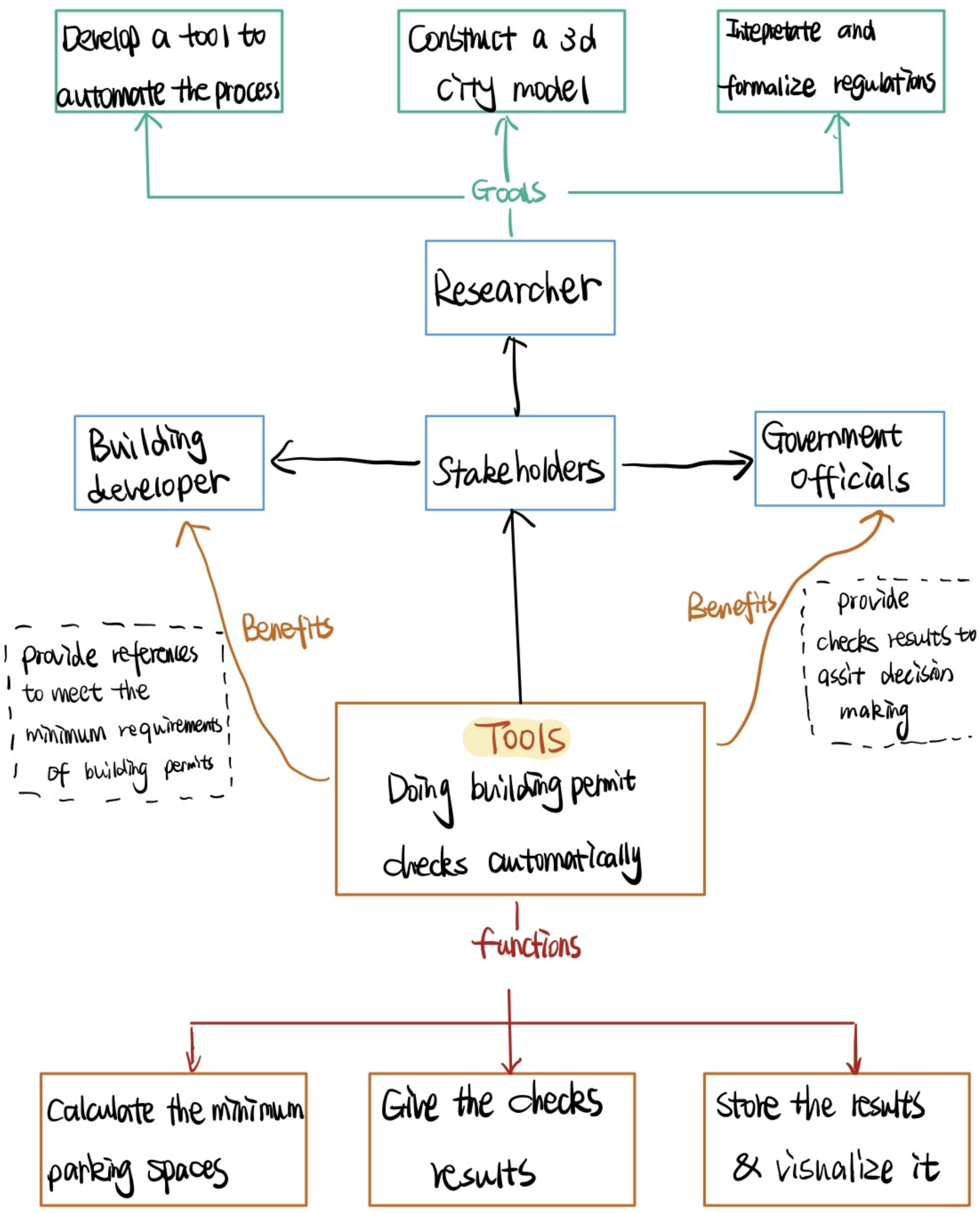

Develop a tool to automate the process
Construct a 3d city model
Intepretate and formalize regulations
Goals
Researcher
Building developer
Stakeholders
Government officials
Benefits
provide references to meet the minimum requirements of building permits
Benefits
provide checks results to assit decision making
Tools
Doing building permit checks automatically
functions
Calculate the minimum parking spaces
Give the checks results
Store the results & visualize it

Figure 6.1: Rich picture of the research

7 Conclusion and future work

The aim of this study is to use the CityJSON extension to complete the automation of building permission checks. The key result will be summarized in this chapter, as well as the answers to the main research question and sub questions. In addition, work that has not yet been completed but may be done in the future will be addressed.

7.1 Overview of research

This study completed the inspection of the building permit through the study of Rotterdam's parking standard and the test building data. The experiment explained and formalized the parking standard, thereby extracting regulation-related information and creating corresponding input data. At the same time, the construction of a 3D city model based on the CityJSON extension was completed. This model extends the CityJSON core data model and is applicable to Rotterdam's parking standards. The main results of the research are shown in the table 7.1.

7.2 Conclusion

In this section, answers will be given to the research questions raised in Chapter 1. First, the sub-questions are answered, and the answer towards to main research questions comes after. To achieve the automation of building permission check using the CityJSON extension, the following sub-questions are derived:

- ***How to select and obtain the information needed by building permit checks?***

  This depends on the content of the specific regulation inspection. After we have completed the interpretation and formalization of the regulation, we can find the information needed to complete the building permit inspection. They generally come from definitions and rules in the regulation. For example, the regulation in this study is parking standard, which involves the functional use of the building, the area of different parts of the building, the gross floor area of the building, and the zone information where the building is located. There are also some basic information about the building, such as the id of the building in the BAG datasets, the ground height and the measured height, the geographic coordinates of the building, etc. This information is retrieved through the datasets sources and stored in our data model. It is used for subsequent calculations.

Table 7.1: Overview of main results

Results
The chosen regulation and the formalization (parking standard of Rotterdam)
Georeferencing the digital zoning map
A GUI program tool to automate the checking process
Specification of the CityJSON extension model/Guidelines to users
The implemented demonstrator - tool performance
Results in CityJSON file (building permit checking results of new buildings and old buildings)
The CityJSON extension conceptual model

- ***How to store and extend the information needed by building permit checks?***

  In order to better represent the data involved in this research and the process of exchanging data, we have constructed a 3D model based on CityJSON, and all City Objects involved in regulation will be stored in this data model. Two situations are involved in this research. In this case, the first is City Object that has been defined in the OGC CityGML standard, and the other is a new object that has not been defined. For existing objects, we can directly use the definition in the CityGML standard. As for the undefined cases, we can create new objects that inherits all the attributes of `core:: _cityObject`. The second case is extend City Objects in different ways. For defined objects, we can directly add new attributes. In our case, we want to document the function of each building. This is a new attribute of existing City Objects so we can directly add it in "extraAttributes" of building. For the extension of new City Objects, We can add them in one member with the name "extraCityObjects" in the extension schema to specify new City Objects. Regulation is a new City Objects defined in our research, it is inherits from `core:: _cityObject`, we also define its attributs and geometry in the data model.

- ***How to test the 3D city model-based tool for the building permit regulation checking?***

  As mentioned in chapter 3, we have developed a method to implement this inspection process, starting with the interpretation and formalization of the law, and converting it into a machine-readable language. Then obtain the required information from the data model, calculate or judge according to the checking rules, and finally get the checking results. In this study, a user case method was adopted to test the feasibility of the entire method. The target regulation is the parking standard of Rotterdam. Through the understanding, interpretation and formalization of the parking standard, the attributes and building function required to calculate the minimum parking quantity are obtained by mapping the required attributes to the built 3D city model. Finally, we write a program to perform calculations and get the final CityJSON file containing the verification results.

After reviewing all the sub-questions, we can give answers to the main research questions:

How to do building permit checks automatically by developing a tool in form of the 3D city model?

From the results and analysis above, it is concluded that this paper proposes the use of CityJSON extensions to realize the automated inspection process for building permits, and develops a corresponding tool to reproduce automated inspections. Use the features of CityJSON that it is more developer-friendly and single-line parsing, we can store the City Objects information related to regulation inspections in the CityJSON data model and extend the existing data model. Through formalization methods based on logical mechanisms, we can translate natural language regulations into machines-readable language. Finally, through the development of the GUI program, the user can directly obtain the CityJSON file containing the verification result. The framework and CityJSON extension data model proposed in this paper can also be applied to other regulations and other cities by users. If the issue is various cities' parking rules, then greater emphasis should be devoted to the collection of building data in that city. If the legislation is different, we must describe the City Objects and its attributes that must be added to the CityJSON extended model through the interpretation and formalization of the law. Therefore, it has good application value.

7.3 Future work

Putting aside the limitations of time and computer performance, we can better improve this process and tools for automated building permit inspection and apply it in more directions. in In the future, we are willing to continue to develop in the following directions:

In this study, a methodology starting from a specific use case was used, and only the parking standard law in Rotterdam was studied. In the future, we hope to apply this framework of building permit inspection to other places in the Netherlands and other fields of law. Large cities such as Amsterdam and The Hague also need to improve the quality and efficiency of building permit inspections. At the same time, laws related to environmental and spatial planning also have research value.

At this stage, the performance of computer equipment is limited. In the future, we will use more powerful equipment to test the applicability of the method, transform the code and better pre-process the data to speed up the running speed of the program and improve the ability of the program to operate.

In the future, the visual rendering of the final result will also be improved, and the web-based result display will provide more interactive functions and filtering functions.

Improve the construction of 3D city models. Urban objects involved in different laws will be included in the model. Therefore, the consistency and stability of the model are very important.

A Reproducibility self-assessment

A.1 Marks for each of the criteria

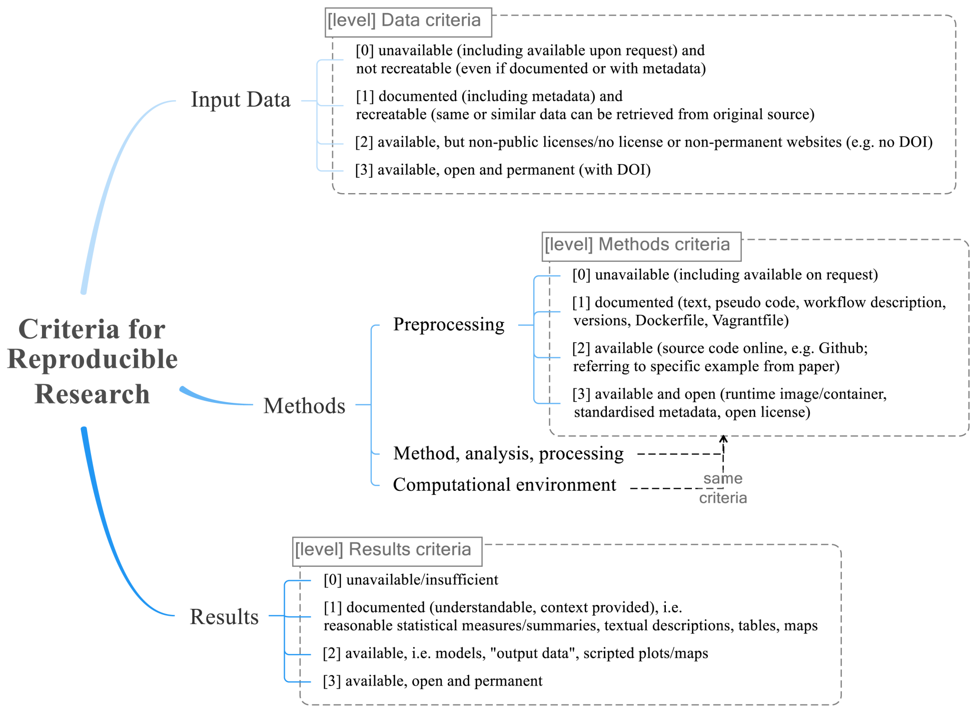

Criteria for Reproducible Research
Input Data
[level] Data criteria
[0] unavailable (including available upon request) and not recreatable (even if documented or with metadata)
[1] documented (including metadata) and recreatable (same or similar data can be retrieved from original source)
[2] available, but non-public licenses/no license or non-permanent websites (e.g. no DOI)
[3] available, open and permanent (with DOI)
Methods
Preprocessing
[level] Methods criteria
[0] unavailable (including available on request)
[1] documented (text, pseudo code, workflow description, versions, Dockerfile, Vagrantfile)
[2] available (source code online, e.g. Github; referring to specific example from paper)
[3] available and open (runtime image/container, standardised metadata, open license)
Method, analysis, processing
Computational environment
same criteria
Results
[level] Results criteria
[0] unavailable/insufficient
[1] documented (understandable, context provided), i.e. reasonable statistical measures/summaries, textual descriptions, tables, maps
[2] available, i.e. models, "output data", scripted plots/maps
[3] available, open and permanent

Figure A.1: Reproducibility criteria to be assessed.

A.2 Self-reflection

The data used in this research is publicly available, and the code related to the research can be publicly accessed on `https://github.com/Gallon229/Building-permit`. When reproducing the research, we need to use some files that we have processed, and these files can also be accessed on GitHub.

Table A.1: Reproducibility evaluation according to the criteria

Category CCriteria	**Comments**	**Score**
Input data	All the data is open source	3
Preprocessing	The preprocessing for data is needed here	2
Method, analysis, processing	The source code can be found at github	3
Computational environment	Mainly used python and QGIS	2
Results	Results and some guidelines files can be found at github	2

B Diagrams and texts

B.1 Specifications of urban planning extensions

1. Regulation

The geometry of a City Object of type '"Regulation"' can be of type '"MultiSurface"' or '"MultiPoint"'. Example of one bicycle parking regulation of Rotterdam can be stored as:

```
"mim_bicycle_parking_spaces": {
  "type": "Regulation",
  "attributes": {
    "+Class": "spatial planning",
    "+Validity": "true",
    "+enactmentDate": "24-03-2018",
    "+expirationDate": "none",
    "+adminLevel": "municipality",
    "+toplevel": "true",
    "+Administration_area": "Rotterdam"
  },
  "geometry": [
    {
      "type": "MultiSurface",
      "lod": 0,
    "boundaries": [
      [[0, 3, 2, 1]], [[4, 5, 6, 7]], [[0, 1, 5, 4]]
    ]
  }]
}
```

The meaning of each attributes is:

"+Class": The legal category to which it belongs, which can be of spatial planning; Environment; Tranport;

"+function": Usage of the regulation (e.g Calculate the appropriate parking spaces for each building)

"+Validity": Document whether the regulation is in effect.

"+enactmentDate": Document the date the law came into effect, if have.

"+expirationDate": Document the date the law expires, if have.

"+Administration_area": Where does it apply

"+adminLevel": Among(municipality, province, country).

"toplevel": whose value is a Boolean (true = 1st-level; false = 2nd-level).

Here, besides these attributes shows in the example, more information used in the calculation will be explained here in the specification.

"buildingarea": It is related to '"Building"' city objects, document the total living area of each building.

"roomarea": It is related to '"Room"' city objects, document the area of each room in the building.

The more detailed version can be accessed at GitHub.

B.2 Tables of rules

Table B.1: Exemption from parking requirement for proximity to public transport

Public transport stop	*Discount factor*		
	0 - 400 meters	*400 - 800 meters*	*800 - 1200 meters*
Rotterdam Central	0.5	0.6	0.7
Beurs, Blaak and Schiedam Center	0.6	0.7	0.8
Other train stations	0.7	0.8	0.9
Other RSR / metro stations within zones A and B	0.7	0.8	0.9
Other tram stops within zone A	0.7	0.8	0.9
RSR / metro stations zone C (except Hoek van Holland and Nesselande)	0.8	0.9	0.95
Other tram stops in zone B	0.8	0.9	0.95
RSR / metro stations in Hoek van Holland and Nesselande	0.9	0.95	1
Other tram stops in zone C.	0.9	0.95	1
Alexander	1	1	1

B.3 CityJSON urban planning extension schema

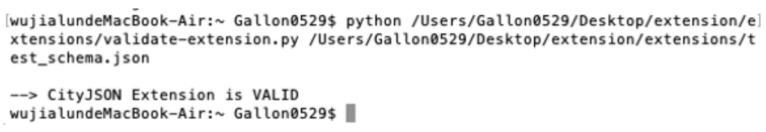
```
{
  "type": "CityJSON_Extension",
  "name": "Urban_planning_extensions",
  "uri": "file://D:/TUD/thesis/fixed_file/test_schema.json",
  "version": "1.0.1",
  "description": "Extension for the urban building permit checking",

  "extraAttributes": {
    "Building": {
      "+height_valid": { "type": "integer" },
      "+groundHeight": { "type": "number" },
      "+non_residential": { "type": "integer" },
      "+total_area": { "type": "number" },
      "+min_bicycle_parking_spaces": { "type": "number" },
      "+min_car_parking_spaces": { "type": "number" },
      "+function1": { "type": "string" },
      "+function2": { "type": "string" },
      "+discount_factor": { "type": "number" }
    },
    "LandUse": {
      "+zone": { "type": "string"}
      }
  },
  "extraCityObjects": {
    "+Room": {
      "allOf": [
        { "$ref": "../cityobjects.schema.json#/_AbstractCityObject" },
        {
          "properties": {
            "type": { "enum": [ "+Room" ] },
            "toplevel": {"type": "boolean"},
            "attributes": {
              "properties": {
                "building_id": { "type": "string" },
                "area": { "type": "number" }
              }
            },
            "geometry": {
              "type": "array",
              "items": {
                "oneOf": [
                  { "$ref": "../geomprimitives.schema.json#/MultiPoint" }
                ]
              }
            }
          },
          "required": [ "type", "toplevel",  "geometry" ]
        }
      ]
    },
     "+Regulation": {
      "allOf": [
        { "$ref": "../cityobjects.schema.json#/_AbstractCityObject" },
        {
          "properties": {
            "type": { "enum": [ "+Regualation" ] },
            "toplevel": {"type": "boolean"},
            "attributes": {
              "properties": {
                 "function": { "type": "string" },
                 "Validity": {"type": "boolean"},
                 "enactmentDate": { "type": "string" },
                 "expirationDate": { "type": "string" },
                 "adminLevel": { "type": "string" },
                 "Administration_area":  { "type": "string" }
              }
            },
            "geometry": {
              "type": "array",
              "items": {
                "oneOf": [
                  { "$ref": "../geomprimitives.schema.json#/MultiSurface" }
                ]
              }
            }
          },
          "required": [ "type", "toplevel", "geometry" ]
        }
      ]
    }
  }
}
```

Table B.2: Formalization of car parking regulation

Definitions and rules from regulation	Definitions

BUH40 = Count BU (function."home") AND (A(BU) 40 m2)
BUH40-65 = Count BU (function."home") AND (40 A(BU) 65 m2)
BUH65-85 = Count BU (function."home") AND (65 A(BU) 85 m2)
BUH85-120 = Count BU (function."home") AND (85 A(BU) 120 m2)
BUH120 = Count BU (function."home") AND (120 m2 A(BU))
Rules (must be true)
 	
For residential buildings:
IF BU(function) = "home"
IF BU intersects Zone = 'A' THEN MinNPP= ((BUH40*0.1) + (BUH40-65*0.4)
+ (BUH65-85*0.6) + (BUH85-120*1)+ (BUH120*1.2))* DF
ELSE IF BU intersects Zone = 'B' THEN
MinNPP= ((BUH40*0.1) + (BUH40-65*0.5)
+ (BUH65-85*0.8) + (BUH85-120*1)+ (BUH120*1.2))* DF
ElSE IF BU intersects Zone = 'C'
THEN MinNPP= ((BUH40*0.1) + (BUH40-65*0.6)
+ (BUH65-85*1.4) + (BUH85-120*1.6)+ (BUH120*1.8)) * DF
 	
NewParkings $\geq$ sum(MinNPP) + sum((MinMQPP/parkingArea))
 	
For non-residential buildings:
Rules (must be true)
BU(function) != "home"
 	
IF BU(function) = "Office"
IF BU intersects Zone = 'A' THEN MinMQPP = 0.76*A(BU)/100 * DF
ELSE IF BU intersects Zone = 'B' THEN MinMQPP = 0.76*A(BU)/100 * DF
ELSE IF BU intersects Zone = 'C' THEN MinMQPP = 0.76*A(BU)/100 * DF
 	
ELSE IF BU(function) = "Industry"
IF BU intersects Zone = 'A' THEN MinMQPP = 0.67*A(BU)/100 * DF
ELSE IF BU intersects Zone = 'B' THEN MinMQPP = 1.2*A(BU)/100 * DF
ELSE IF BU intersects Zone = 'C' THEN MinMQPP = 2*A(BU)/100 * DF
 	
ELSE IF BU(function) = "Retail1"
IF BU intersects Zone = 'A' THEN MinMQPP = 0.38*A(BU)/100 * DF
ELSE IF BU intersects Zone = 'B' THEN MinMQPP = 2.5*A(BU)/100 * DF
ELSE IF BU intersects Zone = 'C' THEN MinMQPP = 2.5*A(BU)/100 * DF
 	
ELSE IF BU(function) = "Supermarket"
IF BU intersects Zone = 'A' THEN MinMQPP = 0.38*A(BU)/100 * DF
ELSE IF BU intersects Zone = 'B' THEN MinMQPP = 2.5*A(BU)/100 * DF
ELSE IF BU intersects Zone = 'C' THEN MinMQPP = 2.8*A(BU)/100 * DF
 	
ELSE IF BU(function) = "Gym"
IF BU intersects Zone = 'A' THEN MinMQPP = 0.08*A(BU)/100 * DF
ELSE IF BU intersects Zone = 'B' THEN MinMQPP = 1.7*A(BU)/100 * DF
ELSE IF BU intersects Zone = 'C' THEN MinMQPP = 2*A(BU)/100 * DF
 	
ELSE IF BU(function) = "Museum"
IF BU intersects Zone = 'A' THEN MinMQPP = 0.02*A(BU)/100 * DF
ELSE IF BU intersects Zone = 'B' THEN MinMQPP = 0.4*A(BU)/100 * DF
ELSE IF BU intersects Zone = 'C' THEN MinMQPP = 0.7*A(BU)/100 * DF
 	
ELSE IF BU(function) = "Cinema"
IF BU intersects Zone = 'A' THEN MinMQPP = 0.01*A(BU)/100 * DF
ELSE IF BU intersects Zone = 'B' THEN MinMQPP = 0.1*A(BU)/100 * DF
ELSE IF BU intersects Zone = 'C' THEN MinMQPP = 0.2*A(BU)/100 * DF
 	
ELSE IF BU(function) = "catering I"
IF BU intersects Zone = 'A' THEN MinMQPP = 0.4*A(BU)/100 * DF
ELSE IF BU intersects Zone = 'B' THEN MinMQPP = 4*A(BU)/100 * DF
ELSE IF BU intersects Zone = 'C' THEN MinMQPP = 6*A(BU)/100 * DF
 	
ELSE IF BU(function) = "catering III"
IF BU intersects Zone = 'A' THEN MinMQPP = 0.4*A(BU)/100 * DF
ELSE IF BU intersects Zone = 'B' THEN MinMQPP = 4*A(BU)/100 * DF
ELSE IF BU intersects Zone = 'C' THEN MinMQPP = 6*A(BU)/100 * DF
 	
ELSE IF BU(function) = "catering IV"
IF BU intersects Zone = 'A' THEN MinMQPP = 1.6*A(BU)/100 * DF
ELSE IF BU intersects Zone = 'B' THEN MinMQPP = 8*A(BU)/100 * DF
ELSE IF BU intersects Zone = 'C' THEN MinMQPP = 12*A(BU)/100 * DF
 	
ELSE IF BU(function) =n "universities"
IF BU intersects Zone = 'A' THEN MinMQPP = 0.5*A(BU)/100 * DF
ELSE IF BU intersects Zone = 'B' THEN MinMQPP = 2*A(BU)/100 * DF
ELSE IF BU intersects Zone = 'C' THEN MinMQPP = 3*A(BU)/100 * DF
 	
ELSE IF BU(function) = "Hospital"
IF BU intersects Zone = 'A' THEN MinMQPP = 1.1*A(BU)/100 * DF
ELSE IF BU intersects Zone = 'B' THEN MinMQPP = 1.3*A(BU)/100 * DF
ELSE IF BU intersects Zone = 'C' THEN MinMQPP = 1.5*A(BU)/100 * DF
 	
NewParkingArea $\geq$ sum(MinMQPP) + sum((MinNPP*parkingArea))
	BU = Building unit, single dwelling
	
	Zone = attribute describing the name
	of the Parking zone to which
	the area (building) belongs to.
	
	DF = Discount factor
	
	BUH= function "home"
	
	BUO= function "Office"
	
	BUI= function "Industry"
	
	BUR= function "Retail1"
	
	BUS= function "Supermarket"
	
	BUG= function "Gym"
	
	BUM= function "Museum"
	
	BUC= function "Cinema"
	
	BUC1= function "catering I"
	
	BUC2= function "catering III"
	
	BUC2= function "catering IV"
	
	BUU= function "universities"
	
	BUH= function "Hospital"
	
	BU(function) = attribute 'function'
	related to each building unit (room) .
	
	A (BU) = attribute'area'related to
	each building unit
	
	MinNPP = Minimum number of
	parking places.
	
	MinMQPP = Minimum parking placesper 100 square meters



```
wujialundeMacBook-Air:~ Gallon0529$ python /Users/Gallon0529/Desktop/extension/e
xtensions/validate-extension.py /Users/Gallon0529/Desktop/extension/extensions/t
est_schema.json

--> CityJSON Extension is VALID
wujialundeMacBook-Air:~ Gallon0529$ 
```

Figure B.1: Extension validity results

Table B.3: Standards table for car non-residential function

	Function	***Number of car parking spaces per 100 m2 GFA unless otherwise specified***		
		Zone A Metropolitan area	Zone B City districts	Zone C Other place
To work	Office	0.76	1.00	1.20
	Labor-intensive / visitor-intensive company (Industry)	0.67	1.20	2.00
Shop	Retail (including thrift store and pharmacy)	0.38	2.50	2.50
	Supermarket	0.38	2.50	2.80
Sports and recreation	Gym, indoor sports hall (incl. Squash, tennis)	0.08	1.70	2.00
Culture	Museum	0.02	0.40	0.70
	Cinema, theater, theater	0.01	0.10	0.20
Catering industry	Cafeteria / snack bar (catering I)	0.40	4.00	6.00
	Café / bar (catering III)	0.40	4.00	6.00
	Restaurant (catering IV)	1.60	8.00	12.00
Education	Vocational education and WO/universities	0.50	2.00	3.00
Care	Hospital	1.10	1.30	1.50

Bibliography

S. Agyeman, H. Abeka, and S. Assiamah. Are the challenges in the processing of building permits a precursor for development of illegal structures in Ghana. *US-China L. Rev.*, 13:337, 2016.

F. C. Ahmed and S. P. Sekar. Using three-dimensional volumetric analysis in everyday urban planning processes. *Applied Spatial Analysis and Policy*, 8(4):393–408, 2015. ISSN 1874-4621.

K. Akahoshi, N. Ishimaru, C. Kurokawa, Y. Tanaka, T. Oishi, T. Kutzner, and T. H. Kolbe. I-urban revitalization: Conceptual modeling, implementation, and visualization towards sustainable urban planning using citygml. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 5(4):179–186, 2020. ISSN 21949050. doi: 10.5194/isprs-Annals-V-4-2020-179-2020.

K. Arroyo Ohori, H. Ledoux, and J. Stoter. A dimension-independent extrusion algorithm using generalised maps. *International Journal of Geographical Information Science*, 29(7):1166–1186, 2015. ISSN 1365-8816.

J. . Barberis, D. W. Arner, and R. P. T. A. T. T. Buckley. The regtech book : the financial technology handbook for investors, entrepreneurs and visionaries in regulation LK - https://tudelft.on.worldcat.org/oclc/1052902944, 2019. URL https://search.ebscohost.com/login.aspx?direct=true{&}scope=site{&}db=nlebk{&}db=nlabk{&}AN=2226927https://rbdigital.rbdigital.comhttps://proquest.safaribooksonline.com/9781119362142https://onlinelibrary.wiley.com/doi/book/10.1002/9781119362197https://doi.org/10.1002/9781119362197.

M. Batty, D. Chapman, S. Evans, M. Haklay, S. Kueppers, N. Shiode, A. Smith, and P. M. Torrens. Visualizing the city: communicating urban design to planners and decision-makers. 2001. ISSN 1589480112.

J. Beetz, van Lahm Léon Berlo, de R Laat, and van den P. Helm. BIMSERVER.Org – An Open Source IFC Model Server. 2010. URL https://semanticscholar.org/paper/c6829db2d4f95464bbda16094c863c5a266954e0.

F. Biljecki, J. Stoter, H. Ledoux, S. Zlatanova, and A. Çöltekin. Applications of 3D city models: State of the art review. *ISPRS International Journal of Geo-Information*, 4(4):2842–2889, 2015. ISSN 22209964. doi: 10.3390/ijgi4042842.

F. Biljecki, H. Ledoux, and J. Stoter. An improved LOD specification for 3D building models. *Computers, Environment and Urban Systems*, 59:25–37, 2016. ISSN 01989715. doi: 10.1016/j.compenvurbsys.2016.04.005.

R. Billen, A.-F. Cutting-Decelle, O. Marina, and J.-P. De Almeida. *3D City Models and urban information: Current issues and perspectives*. edp Sciences, 2021. ISBN 2759811530.

T. Blaschke. Object based image analysis for remote sensing. *ISPRS journal of photogrammetry and remote sensing*, 65(1):2–16, 2010. ISSN 0924-2716.

B. Decree, K. Relations, N. Czw, H. Act, D. No, E. Communities, M. States, D. No, D. No, E. Parliament, T.-e. R. Network, D. No, E. Parliament, A. Section, K. Relations, N. Czw, H. Decree, and A. S. Follows. Draft Building Decree [Bouwbesluit] 2012. (April 2011), 2012.

J. Dimyadi and R. Amor. Automated Building Code Compliance Checking – Where is it at? 2013. URL https://semanticscholar.org/paper/3e6e4b32759797aca585b1e8e6034e8f399ab654.

S. Donkers, H. Ledoux, J. Zhao, and J. Stoter. Automatic conversion of IFC datasets to geometrically and semantically correct CityGML LOD3 buildings. *Transactions in GIS*, 20(4):547–569, 2016. ISSN 1361-1682.

C. Eastman, J. min Lee, Y. suk Jeong, and J. kook Lee. Automatic rule-based checking of building designs. *Automation in Construction*, 18(8):1011–1033, 2009. ISSN 09265805. doi: 10.1016/j.autcon.2009.07.002. URL http://dx.doi.org/10.1016/j.autcon.2009.07.002.

M. Eirinaki, S. Dhar, S. Mathur, A. Kaley, A. Patel, A. Joshi, and D. Shah. A building permit system for smart cities: A cloud-based framework. *Computers, Environment and Urban Systems*, 70(March): 175–188, 2018. ISSN 01989715. doi: 10.1016/j.compenvurbsys.2018.03.006. URL https://doi.org/10.1016/j.compenvurbsys.2018.03.006.

EUnet4DBP. about @ 3d.bk.tudelft.nl, 2020. URL https://3d.bk.tudelft.nl/projects/eunet4dbp/about.html.

K.-H. H. Gerhard Gröger, Thomas H. Kolbe, Claus Nagel. Open Geospatial Consortium OGC City Geography Markup Language (CityGML) En- coding Standard. 2012.

D. Guler and T. Yomralioglu. A reformative framework for processes from building permit issuing to property ownership in Turkey. *Land Use Policy*, 101(October 2020):105115, 2021. ISSN 02648377. doi: 10.1016/j.landusepol.2020.105115. URL https://doi.org/10.1016/j.landusepol.2020.105115.

N. Haala and M. Kada. An update on automatic 3D building reconstruction. *ISPRS Journal of Photogrammetry and Remote Sensing*, 65(6):570–580, 2010. ISSN 0924-2716.

E. N. Ishimaru, C. Kurokawa, Y. Tanaka, and T. Oishi. Open Geospatial Consortium CityGML Urban Planning ADE for i-Urban Revitalization. 2020.

I. S. O. ISO. 19107: 2003 Geographic information-Spatial schema. *International Organization for Standardization*, 90, 2003.

T. H. Kolbe. Representing and exchanging 3D city models with CityGML. In *3D geo-information sciences*, pages 15–31. Springer, 2009.

H. Ledoux. val3dity: validation of 3D GIS primitives according to the international standards. *Open Geospatial Data, Software and Standards*, 3(1):1, dec 2018. ISSN 2363-7501. doi: 10.1186/s40965-018-0043-x. URL https://opengeospatialdata.springeropen.com/articles/10.1186/s40965-018-0043-x.

H. Ledoux and M. Meijers. Topologically consistent 3D city models obtained by extrusion. *International Journal of Geographical Information Science*, 25(4):557–574, 2011. ISSN 1365-8816.

H. Ledoux, K. Arroyo Ohori, K. Kumar, B. Dukai, A. Labetski, and S. Vitalis. CityJSON: A compact and easy-to-use encoding of the CityGML data model. *arXiv*, pages 0–12, 2019. ISSN 2363-7501. doi: 10.1186/s40965-019-0064-0.

H. Lee, J.-K. Lee, S. Park, and I. Kim. Translating building legislation into a computer-executable format for evaluating building permit requirements. *Automation in Construction*, 71:49–61, nov 2016. ISSN 09265805. doi: 10.1016/j.autcon.2016.04.008. URL https://linkinghub.elsevier.com/retrieve/pii/S0926580516300796.

J. Lee. Building environment rule and analysis (BERA) languageand its application for evaluating building circulation and spatial program. 2011. URL https://semanticscholar.org/paper/0877d934c912d49fe71dc74b5b12b132aae30e38.

M. Lemmens. Applying geo-information technology. In *Geo-information*, pages 229–258. Springer, 2011.

R. Lewis and C. Séquin. Generation of 3D building models from 2D architectural plans. *Computer-Aided Design*, 30(10):765–779, 1998. ISSN 0010-4485.

S. M. Malsane, J. Matthews, S. Lockley, P. Love, and D. Greenwood. Development of an object model for automated compliance checking. 2015. doi: 10.1016/J.AUTCON.2014.10.004. URL https://semanticscholar.org/paper/e031c61ad8c203d49a656d8444d7635ecb3e4d16.

A. March and M. Kornakova. *Urban planning for disaster recovery*. Butterworth-Heinemann, 2017. ISBN 0128043237.

P. J. May. Regulatory implementation: Examining barriers from regulatory processes. *Cityscape*, pages 209–232, 2005. ISSN 1936-007X.

W. Mazairac and J. Beetz. BIMQL - An open query language for building information models. 2013. doi: 10.1016/j.aei.2013.06.001. URL https://semanticscholar.org/paper/b7240e3028f568d587bccfe9a0bee42fd8044212.

K. Mouratidis. Urban planning and quality of life: A review of pathways linking the built environment to subjective well-being. *Cities*, 115:103229, aug 2021. ISSN 02642751. doi: 10.1016/j.cities.2021.103229. URL https://linkinghub.elsevier.com/retrieve/pii/S0264275121001293.

F. Noardo, G. Malacarne, V. S. Mastrolembo, L. Tagliabue, A. Ciribini, C. Ellul, D. Guler, L. Harrie, L. Senger, A. Waha, et al. Integrating expertises and ambitions for data-driven digital building permits-the eunet4dbp. *ISPRS Archives; volume 44, 4, W1*, 44(4/W1):103–110, 2020a.

F. Noardo, T. Wu, K. Arroyo Ohori, T. Krijnen, H. Tezerdi, and J. Stoter. Geobim for digital building permit process: Learning from a case study in Rotterdam. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 6(4/W1):151–158, 2020b. ISSN 21949050. doi: 10.5194/isprs-annals-VI-4-W1-2020-151-2020.

F. Noardo, T. Wu, K. A. Ohori, T. Krijnen, and J. Stoter. Investigating the Automation of the Building Permits Issuing process through 3D GeoBIM information - Final Deliverable. 2020c.

Open Knowledge Foundation. ccfb4e5bb908e67ac1c8462dd3d3592fd738be30 @ opendatahandbook.org, 2019. URL http://opendatahandbook.org/glossary/en/terms/machine-readable/.

C. Preidel and A. Borrmann. Automated Code Compliance Checking Based on a Visual Language and Building Information Modeling. 2015. doi: 10.22260/ISARC2015/0033. URL https://semanticscholar.org/paper/8bad2035f6e35af8aaeda7bf7647933df7779dc1.

L. Ross. Virtual 3D City Models in Urban Land Management-Technologies and Applications. 2011.

N. Shiode. 3D urban models: Recent developments in the digital modelling of urban environments in three-dimensions. *GeoJournal*, 52(3):263–269, 2000. ISSN 1572-9893.

M. Sinning-Meister, A. Gruen, and H. Dan. 3D City models for CAAD-supported analysis and design of urban areas. *ISPRS Journal of Photogrammetry and Remote Sensing*, 51(4):196–208, 1996. ISSN 0924-2716.

W. Solihin and C. Eastman. Classification of rules for automated BIM rule checking development. 2015. doi: 10.1016/J.AUTCON.2015.03.003. URL https://semanticscholar.org/paper/675565c7a29e4c393b89e082f75c6a696a8eb70a.

J. Stoter, M. Reuvers, G. Vosselman, J. Goos, L. van Berlo, S. Zlatanova, E. Verbree, and R. Klooster. Towards a 3d geo-information standard in the netherlands. In *Proceedings of 5th international conference on 3D geoinformation, Berlin, Germany, 3-4 November 2010:(ISPRS: Volume XXXVIII, part 4/W15)*, pages 63–67. International Society for Photogrammetry and Remote Sensing (ISPRS), 2010.

I. Suveg and G. Vosselman. Reconstruction of 3D building models from aerial images and maps. *ISPRS Journal of Photogrammetry and remote sensing*, 58(3-4):202–224, 2004. ISSN 0924-2716.

I. Tomljenovic, B. Höfle, D. Tiede, and T. Blaschke. Building extraction from airborne laser scanning data: An analysis of the state of the art. *Remote Sensing*, 7(4):3826–3862, 2015.

K. Ulm. Virtual 3D City Models-satisfaction through sustainability'. *Geomatics World*, 18(6):16–18, 2010.

S. Vitalis, K. Ohori, and J. Stoter. Incorporating Topological Representation in 3D City Models. *ISPRS International Journal of Geo-Information*, 8(8):347, 2019. doi: 10.3390/ijgi8080347.

J. Wendel, A. Simons, A. Nichersu, and S. M. Murshed. Rapid development of semantic 3D city models for urban energy analysis based on free and open data sources and software. In *Proceedings of the 3rd ACM SIGSPATIAL Workshop on Smart Cities and Urban Analytics*, pages 1–7, 2017.

X. Yin, P. Wonka, and A. Razdan. Generating 3d building models from architectural drawings: A survey. *IEEE computer graphics and applications*, 29(1):20–30, 2008. ISSN 0272-1716.

Colophon

This document was typeset using LaTeX, using the KOMA-Script class `scrbook`. The main font is Palatino.