

Automatic building permits checks by means of 3D city models

Jialun Wu

Supervisors: Francesca Noardo
Ken Arroyo Ohori
Co-reader: Jantien Stoter



Contents

INTRODUCTION

01

METHODOLOGY

02

IMPLEMENTATION

03

RESULTS & ANALYSIS

04

CONCLUSIONS

05

Introduction

01

Background

02

Related research & research gap

03

Solution

04

Research Objective

Background

-Importance of building permit



Image sources:
<https://images.app.goo.gl/GLPGnYGsTwjczUrd8>

Motivation

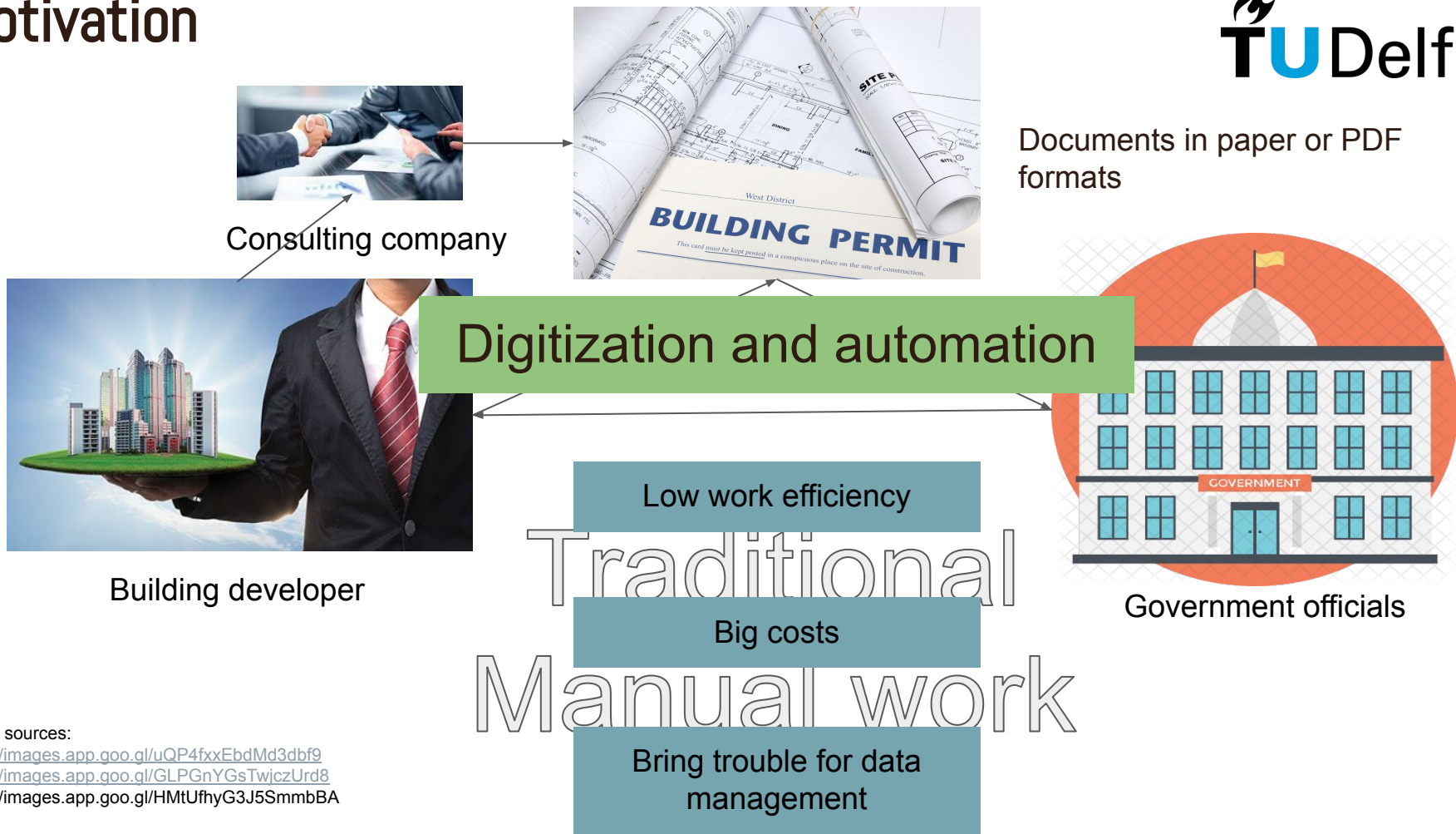


Image sources:

<https://images.app.goo.gl/uQP4fxxEbdMd3dbf9>

<https://images.app.goo.gl/GLPGnYGsTwiczUrd8>

<https://images.app.goo.gl/HMtUfhyG3J5SmmbBA>

Related research & research gap

Building permit

Checked the building permit through GeoBIM (Noardo et al., 2020c)

3D city model

CityGML & CityJSON

Extensions of 3D city model

*Urban planning CityGML ADE (Ishimaru et al., 2020)
Noise CityJSON extension (Dukai, 2018)*

Research gap/challenges

Some building context is not supported by the current buildingSMART

Specific urban planning applications are not yet available
Only a few applications at the moment

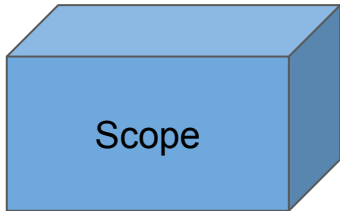
Solution

Automatic building permits
checks by means of 3D city
models



Benefits:

- Improve work efficiency
- Speed up the working process
- Save costs
- Better data management



Use cases method : start with a few regulations and part of Rotterdam

Consulted the officials for ambiguous rules

Research Objective

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

Select and obtain
the information

Store and extend
information

Test the 3D city
model-based tool

Methodology

01

Interpretation and formalization of regulations

02

Use cases

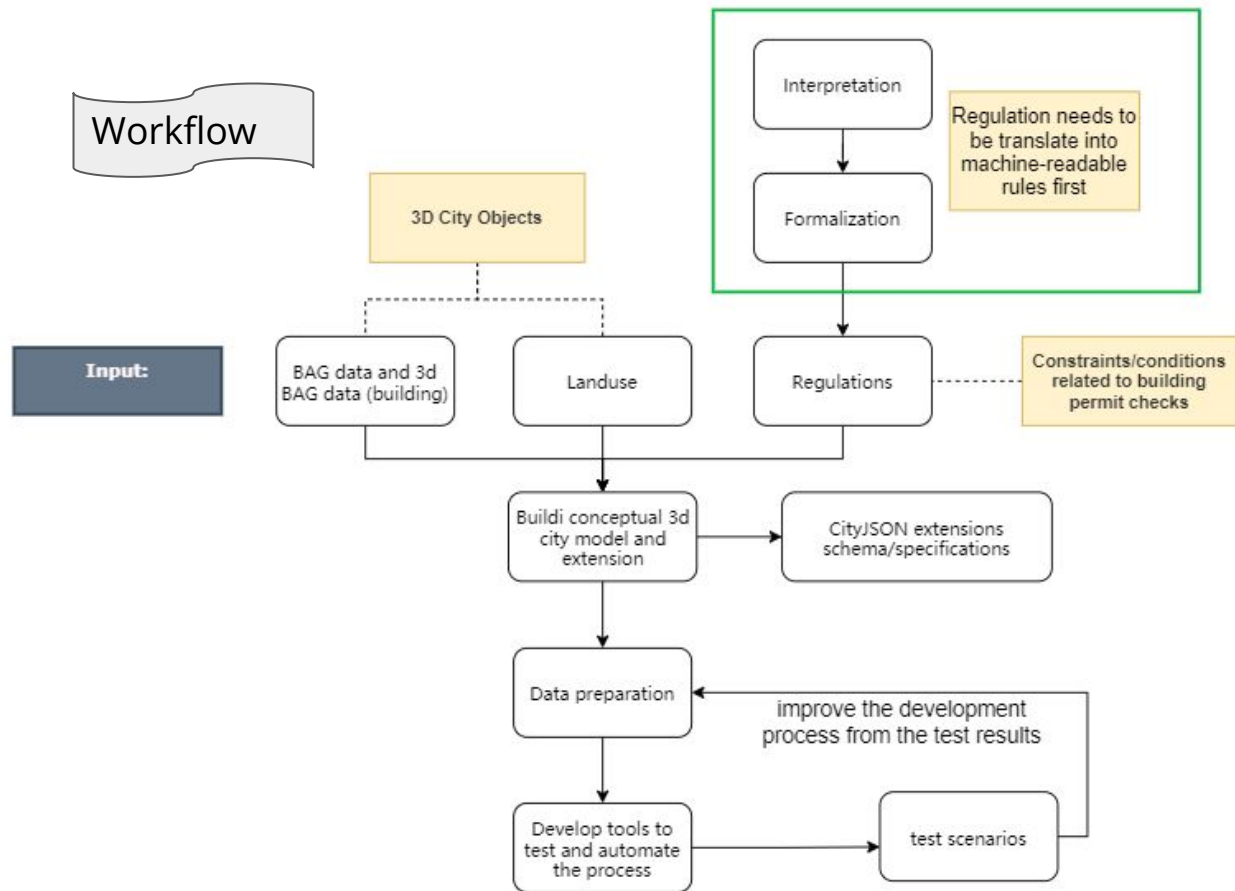
03

Build conceptual data model

04

Develop a program for automate the check process

Methodology



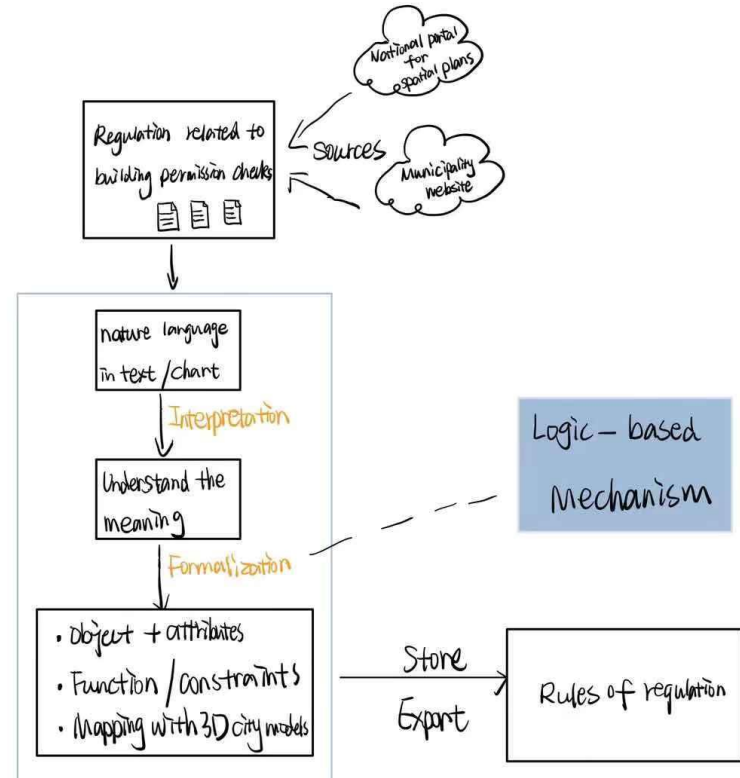
Aim:

using 3D city models and its extensions to complete building permit checks, and develops tools to automate the process

1. Interpretation and formalization of regulations

Purpose: Understand and translate regulations into rules/algorithms

A working process that can be applied on some basic regulations is proposed:



2.1 3D city model

*3D city models are powerful tools for city-level analysis,
so they can be used in building permit checks*

CityGML

versus

CityJSON



More
compact

Simplified encoding &
Easy decoding

Supported by many
programming languages

Friendly to
developers

2.2 Build conceptual data model

Purpose: Build CityJSON data model and write CityJSON extensions to extends different City Objects



1. Data pre-processing



Convert source file into CityJSON file and extract data in Rotterdam



2. Conceptual model

Classify all the City Objects



Existing objects



New objects

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

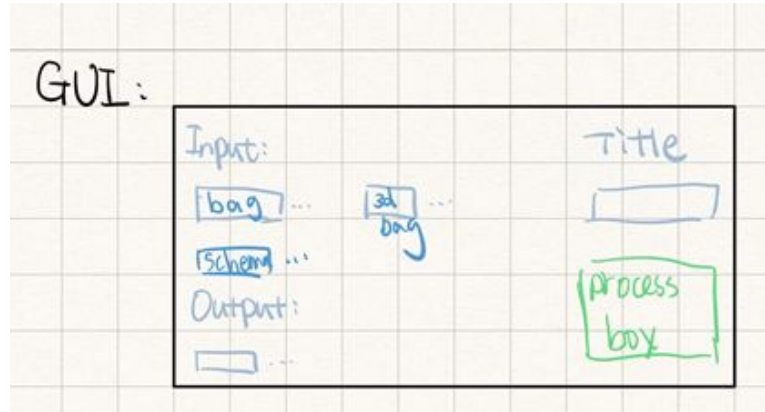
- Add new attributes to existing City Objects
- Create new City Objects and define their geometry
- Add new attributes at the root of the document



3. Write schema of extensions and check validity

3. Develop a program for automate the check process

Using the PyQt module in python to develop a GUI program



design of the GUI program

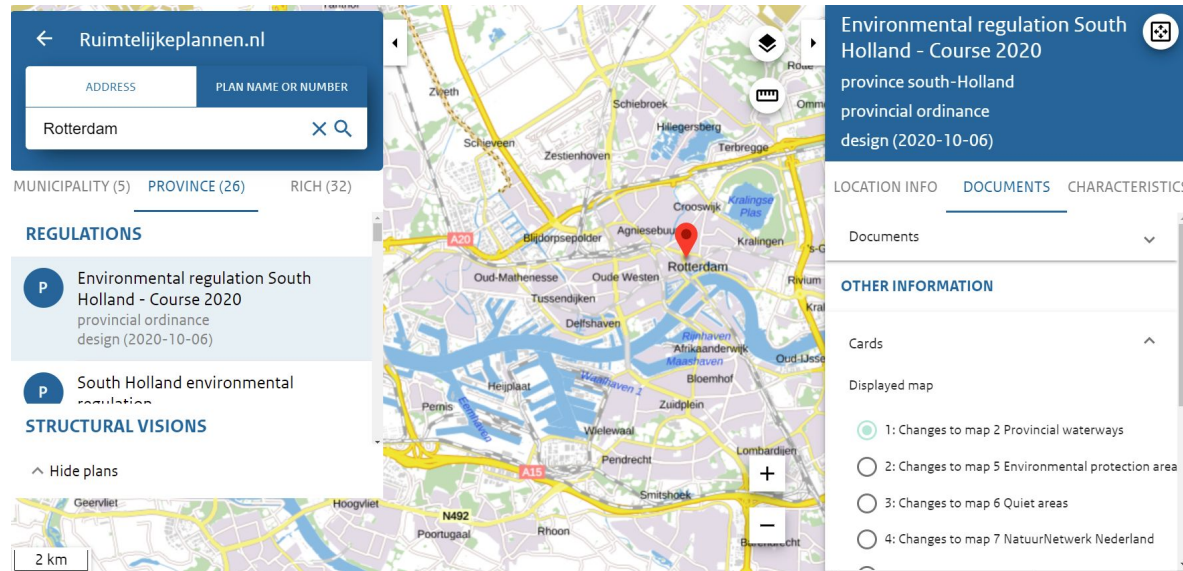
- Create and organize icons, widgets that are displayed to users.
- Define the function that will handle user and application events.
- Associate specific user events with specific functions.
- Write the main code.

Use cases for regulations

1. Selection of regulations
2. Interpretation of parking regulation
3. Formalization of parking regulation

Selection of regulations

Zoning plans, structural visions and general rules implemented in Rotterdam can be found on this website: [Ruimtelijke plannen](https://ruimtelijkeplannen.nl)



The screenshot displays the Ruimtelijkeplannen.nl website interface. On the left, a search bar is set to 'Rotterdam'. Below it, a list of regulations is shown, with the first entry highlighted: 'Environmental regulation South Holland - Course 2020', a provincial ordinance design (2020-10-06). The central part of the image is a map of Rotterdam, showing various districts and waterways. On the right, a detailed view of the selected regulation is shown, including its title, location information, and a list of documents. Below this, there is a section for 'OTHER INFORMATION' which includes a list of cards and a displayed map. The map shows the location of the regulation in Rotterdam, with a red pin indicating the specific area.

Ruimtelijkeplannen.nl

ADDRESS PLAN NAME OR NUMBER

Rotterdam

MUNICIPALITY (5) PROVINCE (26) RICH (32)

REGULATIONS

- P** Environmental regulation South Holland - Course 2020
provincial ordinance design (2020-10-06)
- P** South Holland environmental regulation

STRUCTURAL VISIONS

Hide plans

2 km

Environmental regulation South Holland - Course 2020
province south-Holland
provincial ordinance design (2020-10-06)

LOCATION INFO DOCUMENTS CHARACTERISTICS

Documents

OTHER INFORMATION

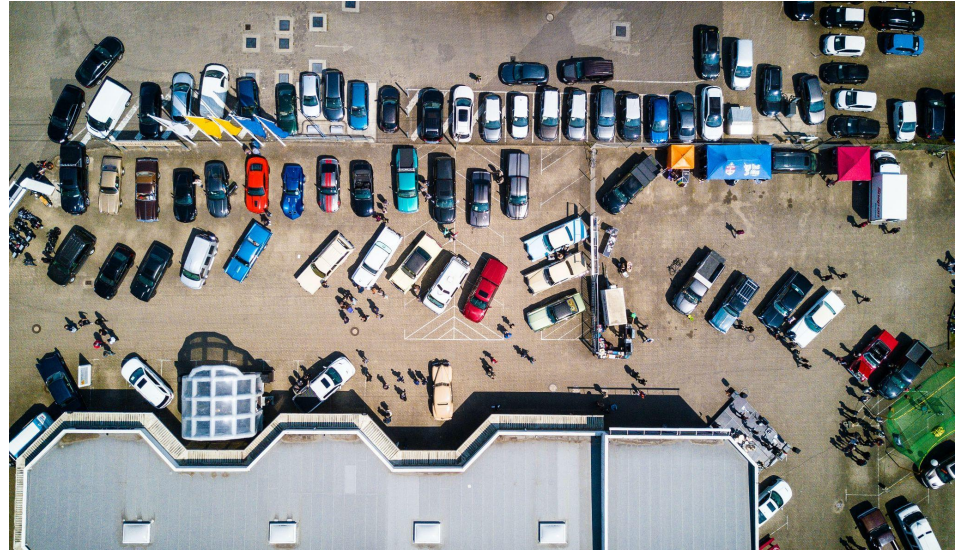
Cards

Displayed map

- ☒ 1: Changes to map 2 Provincial waterways
- ☐ 2: Changes to map 5 Environmental protection area
- ☐ 3: Changes to map 6 Quiet areas
- ☐ 4: Changes to map 7 NatuurNetwerk Nederland

Selection of regulations

In Rotterdam, the municipality government announced a policy regulation on **parking standards** for bicycles and cars based on different building functions.



Interpretation of parking regulation

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



Chiang San Lin-Harpal N.D. (Nielma) <nd.chiangsanlinharpal@rotter

Beste, Op woensdag 19 mei ben ik weer bereikbaar. Met vriendelijke groet, Nielma Chian...



Jialun Wu

Dear Wouter, Thanks for your reply and the interpretation !! It is much clearer for me no...



Streefkerk W. (Wouter) <w.streefkerk@rotterdam.nl>

Dear Jialun, Below the answers in blue to your questions. The dutch answer is from my c...



Jialun Wu

Dear Rolf and Wouter, Hello, this is Jialun from Geomatics, TU Delft. I'm working on my...

Article 5. Exemptions for small projects

Small projects are completely exempt from the parking requirement. Small projects include developments that comply with the maximum surface areas in table 2.2.

	Residential functions (GBO total project)	Non-residential functions (GFA total project)
Parking requirement car	Exempt up to 300 m2 gbo	Exempt up to 600 m2 GFA
Parking requirement bicycle	No exemption	Exempt up to 600 m2 GFA, catering functions up to 200 m2 GFA

Formalization of parking regulation

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





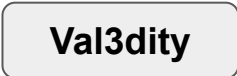


For residential buildings:
BUH40 = Count BU (function."home")
AND (A(BU) 40 m2)
BUH40-65 = Count BU (function."home")
AND (40 A(BU) 65 m2)
BUH65-85 = Count BU (function."home")
AND (65 A(BU) 85 m2)
BUH85 = Count BU (function."home")
AND (85 m2 A(BU))

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))

Implementation

- Tools and Datasets Used
- Overview of experiments




Software and tools

- viewing and pre-processing of data 
- JSON schema validator  
- Coding  python
- 3d visualization: Cesium-cityjson 

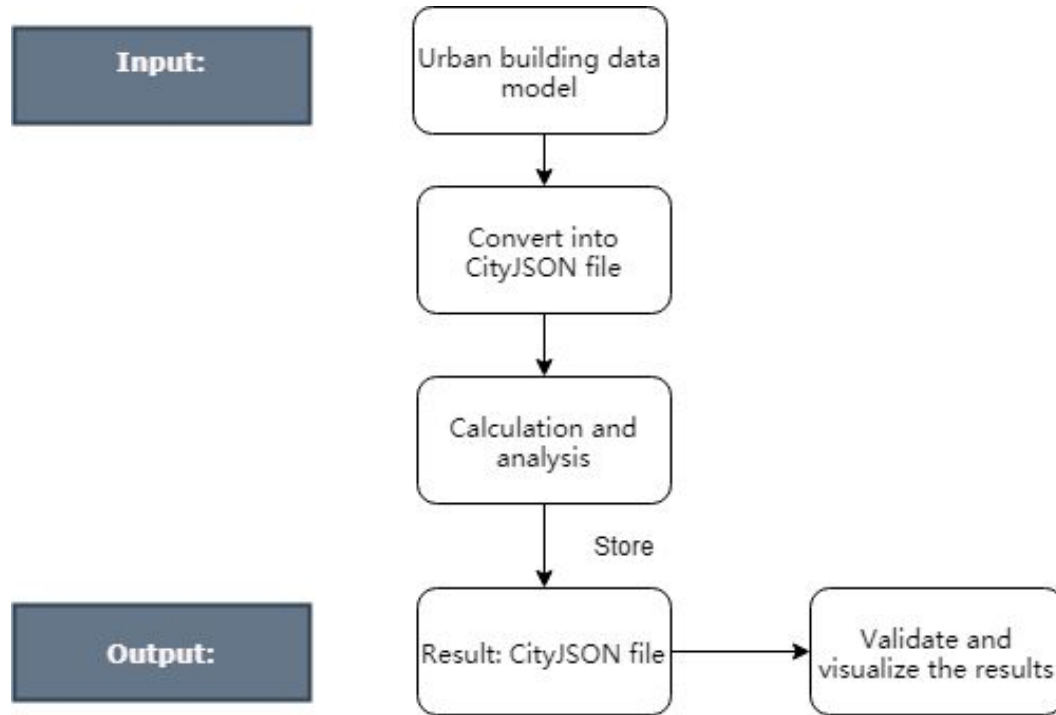
Hardware

MSI Codex S 8RA-003XEU - Gaming Desktop
Intel Core i5 processor at 2,8 GHz
8 GB of RAM
240 GB of SSD memory

Datasets

- Basisregistratie Adressen en Gebouwen (BAG) 
- 3D BAG - (height information) 
- OSM - (buildings with different functional attributes) 

Implementation

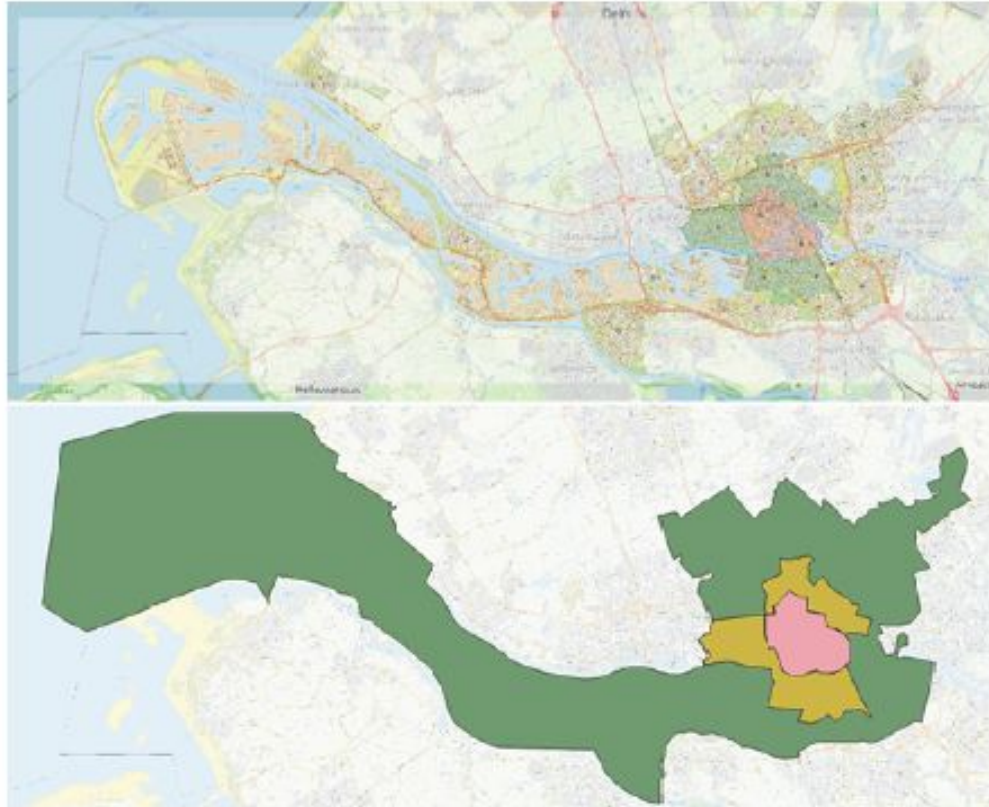


Data pre-processing and input

1. Obtain information from datasets
2. Georeferencing the digital version of the zone classification map
3. Create new building datasets
4. Query OSM data (transportation)
5. Extract test area
6. Buffer analysis on transportation

	Information	Explanation	Sources	name in sources
7*Attributes	id	Bag id of building	BAG	identificatie
	+function	Function of buildings included inodelist	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

Georeferencing the digital version of the zone classification map



Create new datasets

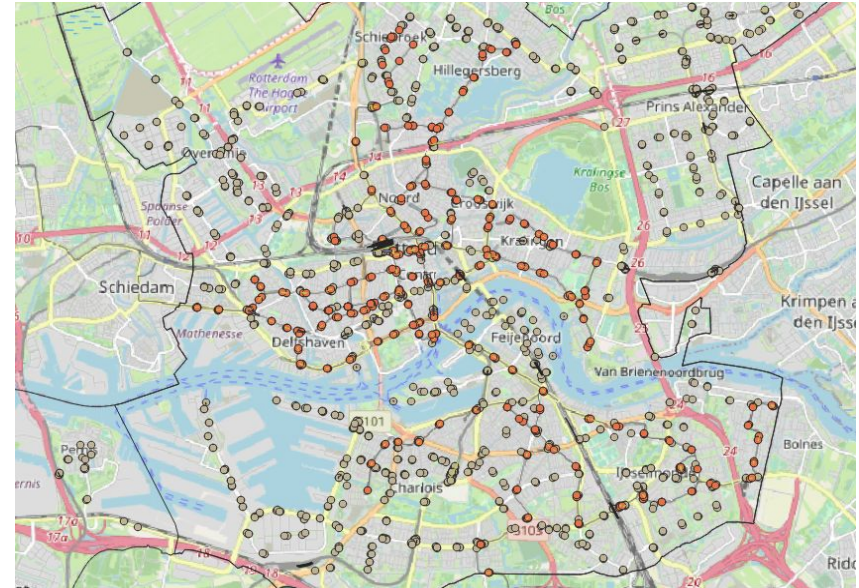


id	zone	footprint_area	point1_area	point2_area
5	B	134.221	93.9547	107.3768
4	B	121.479	85.0353	97.1832
7	B	55.419	38.7933	44.3352
6	B	63.357	44.3499	50.6856
97	C	920.004	644.0028	736.0032
96	C	681.05	476.735	544.84
99	C	571.11	399.777	456.888
98	C	800.069	560.0483	640.0552
100	C	682.689	477.8823	546.1512
89	C	74.818	52.3726	59.8544

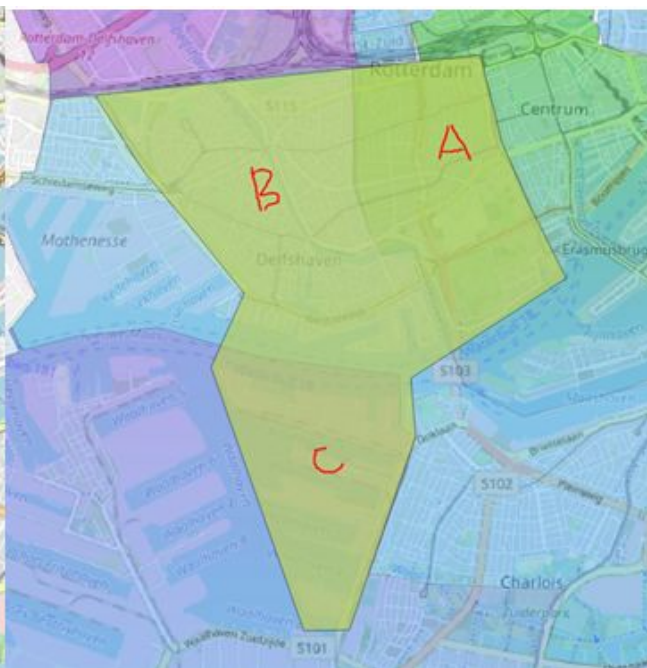
Create new datasets & Query OSM data

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>
```



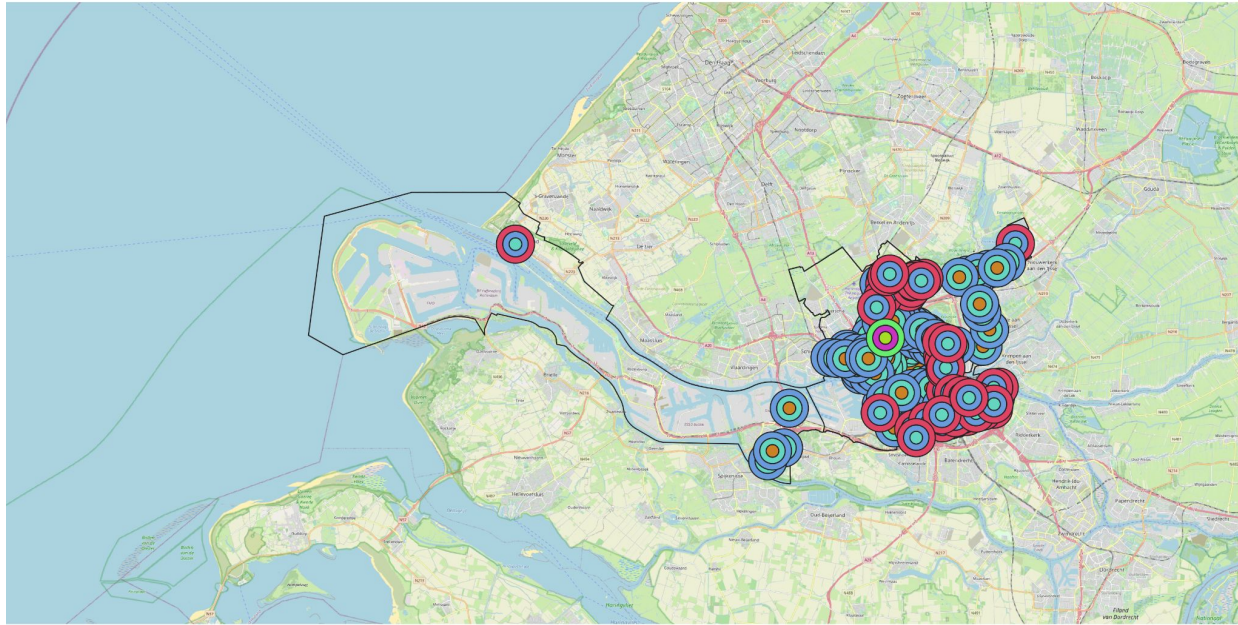
Implementation - test area



This analysis is based on the special exemptions from the parking requirement in the regulation.

1. Get the layer which contains tram/metro/train station
2. Using the criteria to do the buffer analysis on those facilities to get the discount factor (0-400m;400-800m;800-1200m)
3. Overlap the two layers (join the attribute of discount factor to new buildings)

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

Buffer analysis on transportation

Public transport stop⁴³

Development as the crow flies - public transport
stop entrance⁴³

	0 - 400 meters ⁴³	400 - 800 meters ⁴³	800 - 1200 meters ⁴³
Rotterdam Central ⁴³	0.5 ⁴³	0.6 ⁴³	0.7 ⁴³
Beurs, Blaak and Schiedam Center ⁴³	0.6 ⁴³	0.7 ⁴³	0.8 ⁴³
Other train stations ⁴³	0.7 ⁴³	0.8 ⁴³	0.9 ⁴³
Other RSR / metro stations within zones A and B ⁴³	0.7 ⁴³	0.8 ⁴³	0.9 ⁴³
Other tram stops within zone A ⁴³	0.7 ⁴³	0.8 ⁴³	0.9 ⁴³
RSR / metro stations zone C (except Hoek van Holland and Nesselande) ⁴³	0.8 ⁴³	0.9 ⁴³	0.95 ⁴³
Other tram stops in zone B ⁴³	0.8 ⁴³	0.9 ⁴³	0.95 ⁴³
RSR / metro stations in Hoek van Holland and Nesselande ⁴³	0.9 ⁴³	0.95 ⁴³	1 ⁴³
Other tram stops in zone C ⁴³	0.9 ⁴³	0.95 ⁴³	1 ⁴³
Alexander ⁴³	1 ⁴³	1 ⁴³	1 ⁴³

Table 2.3: Exemption from parking requirement for proximity to public transport⁴⁴

id	zone	footprint_area	point1_area	point2_area	discount_factor	min_discount
1	B	226.024	158.2168	180.8192	0.9	0.8
1	B	226.024	158.2168	180.8192	0.9	0.8
1	B	226.024	158.2168	180.8192	0.8	0.8
1	B	226.024	158.2168	180.8192	0.9	0.8
1	B	226.024	158.2168	180.8192	0.95	0.8
1	B	226.024	158.2168	180.8192	0.95	0.8
1	B	226.024	158.2168	180.8192	0.95	0.8
1	B	226.024	158.2168	180.8192	0.95	0.8
1	B	226.024	158.2168	180.8192	0.95	0.8
1	B	226.024	158.2168	180.8192	0.8	0.8
1	B	226.024	158.2168	180.8192	0.9	0.8

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)
```

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
```


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)
```

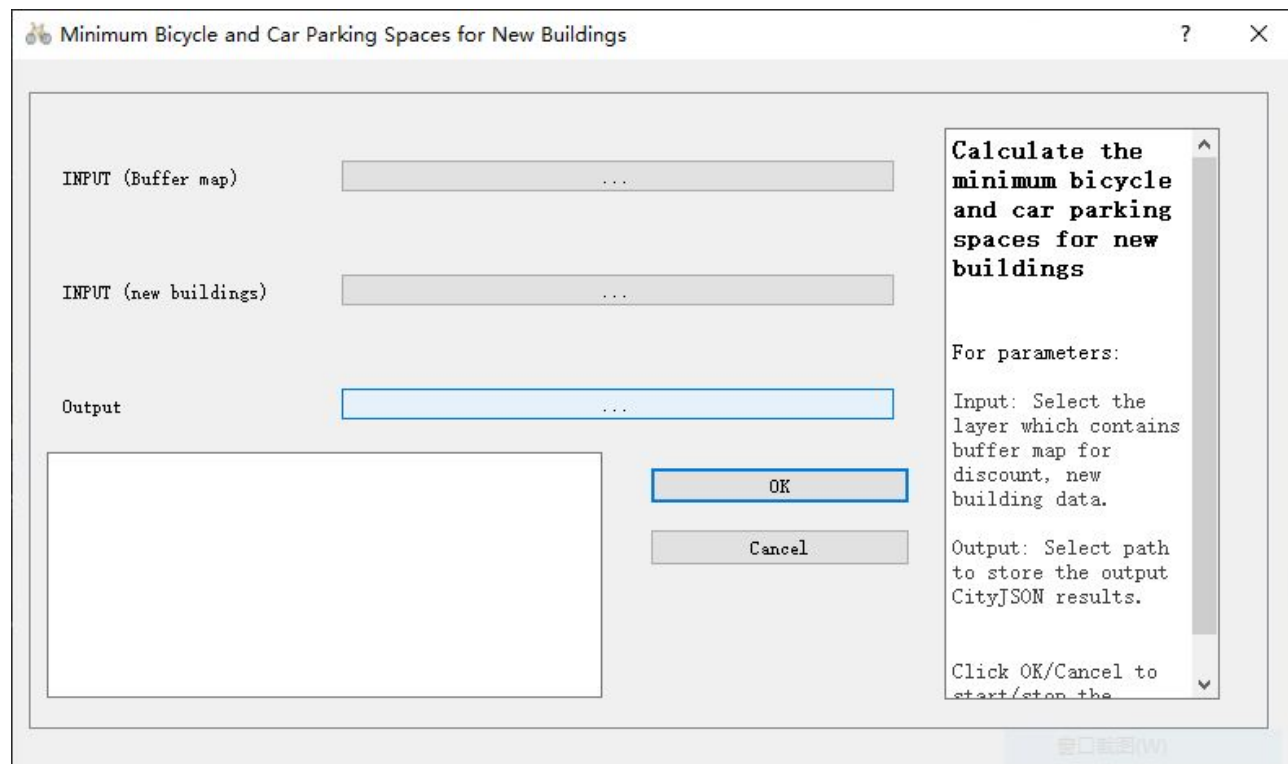
Key part of code: the calculation process for non-residential buildings (e.g office function)

```
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)
```

commands for validation

```
cjio path of the file validate --folder_schema path of extension  
val3dity path of the file
```

GUI program application



Results and Discussion

- Formalization of the chosen regulation (parking standard of Rotterdam)
- Georeferencing the digital zoning map & Buffer analysis on transportation
- The developed CityJSON extension and the extension schema
- Results in CityJSON file (building permit checking results of new buildings and old buildings)
- Specification of the CityJSON extension model/Guidelines to users
- A GUI program tool to automate the checking process

Building permit checks of old buildings (only for test)

```
"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
```

```
esktop/extension  
Parsing /Users/Gallon0529/Desktop/extension/extensions/buildings_calculation.js  
n  
==== Validation (with provided schemas) =====  
-- Validating the syntax of the file  
  (using the schemas /Users/Gallon0529/Desktop/extension/cityjson.schema.js  
son)  
-- Validating the Extensions  
  Urban_planning_extensions [test_schema.json]  
-- Validating the internal consistency of the file (see docs for list)  
  --Vertex indices coherent  
  --Specific for CityGroups  
  --Semantic arrays coherent with geometry  
  --Root properties  
  --Empty geometries  
  --Duplicate vertices  
  --Orphan vertices  
  --CityGML attributes  
=====  
File is valid
```



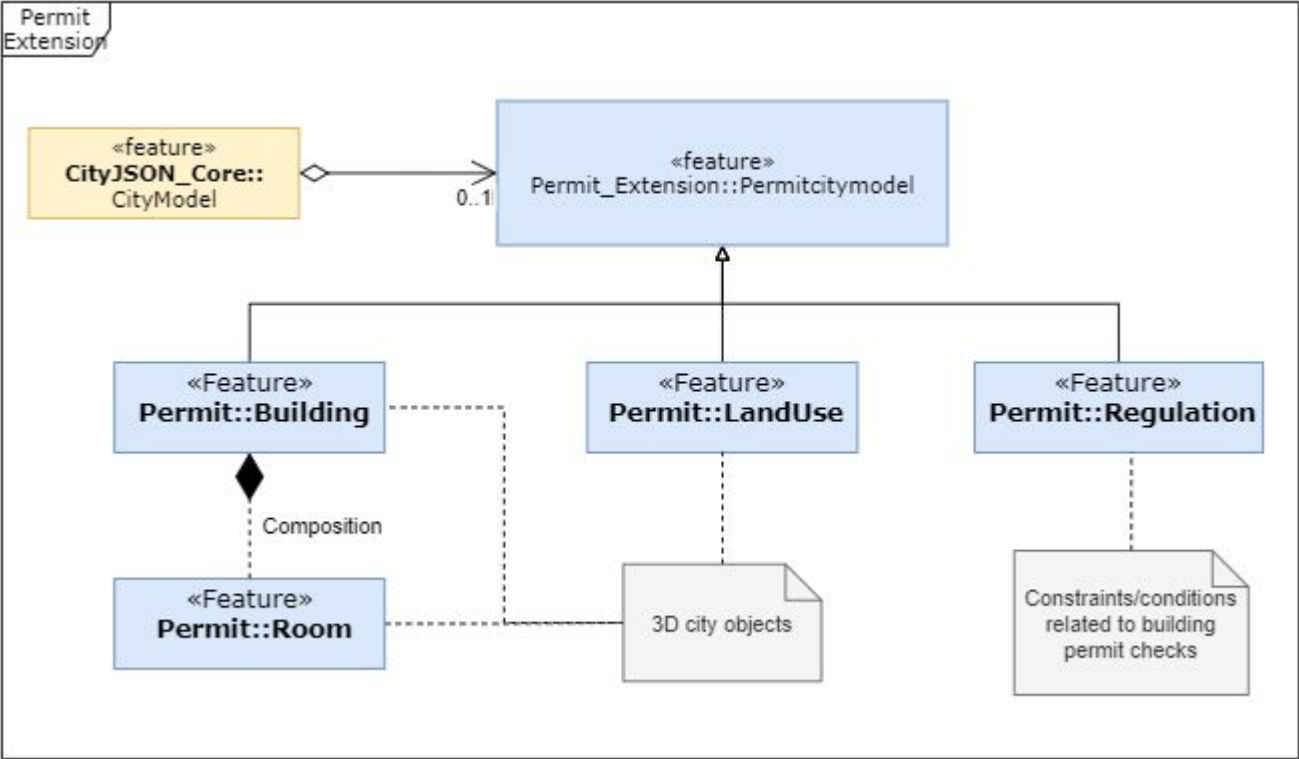
Building permit checks of new buildings

```
"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"
```

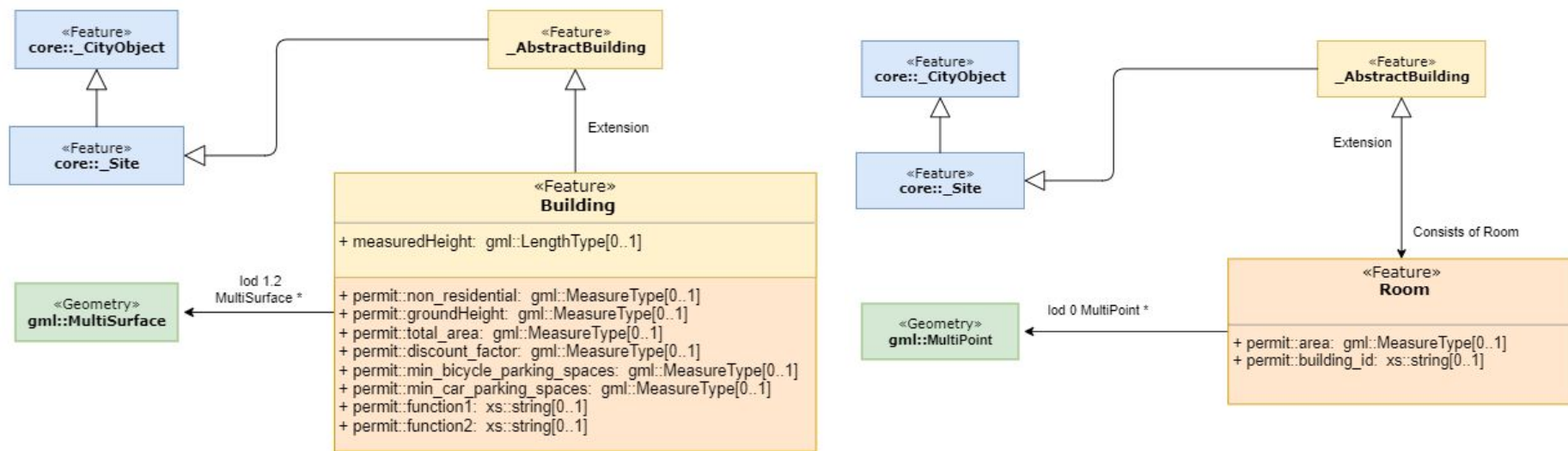
```
C:\Users\Gallon>cjio D:\TUD\thesis\p4\car_parking\car_parking_new_buildings_with_points.json validate --folder_schemas "D:\TUD\thesis\about extension schema\extension"  
Parsing D:\TUD\thesis\p4\car_parking\car_parking_new_buildings_with_points.json  
==== Validation (with provided schemas) =====  
-- Validating the syntax of the file  
  (using the schemas D:\TUD\thesis\about extension schema\extension\cityjson.schema.json)  
-- Validating the Extensions  
  Urban_planning_extensions [test_schema.json]  
-- Validating the internal consistency of the file (see docs for list)  
  --Vertex indices coherent  
  --Specific for CityGroups  
  --Semantic arrays coherent with geometry  
  --Root properties  
  --Empty geometries  
  --Duplicate vertices  
  --Orphan vertices  
  --CityXML attributes  
=====  
File is valid  
File has no warnings  
=====
```



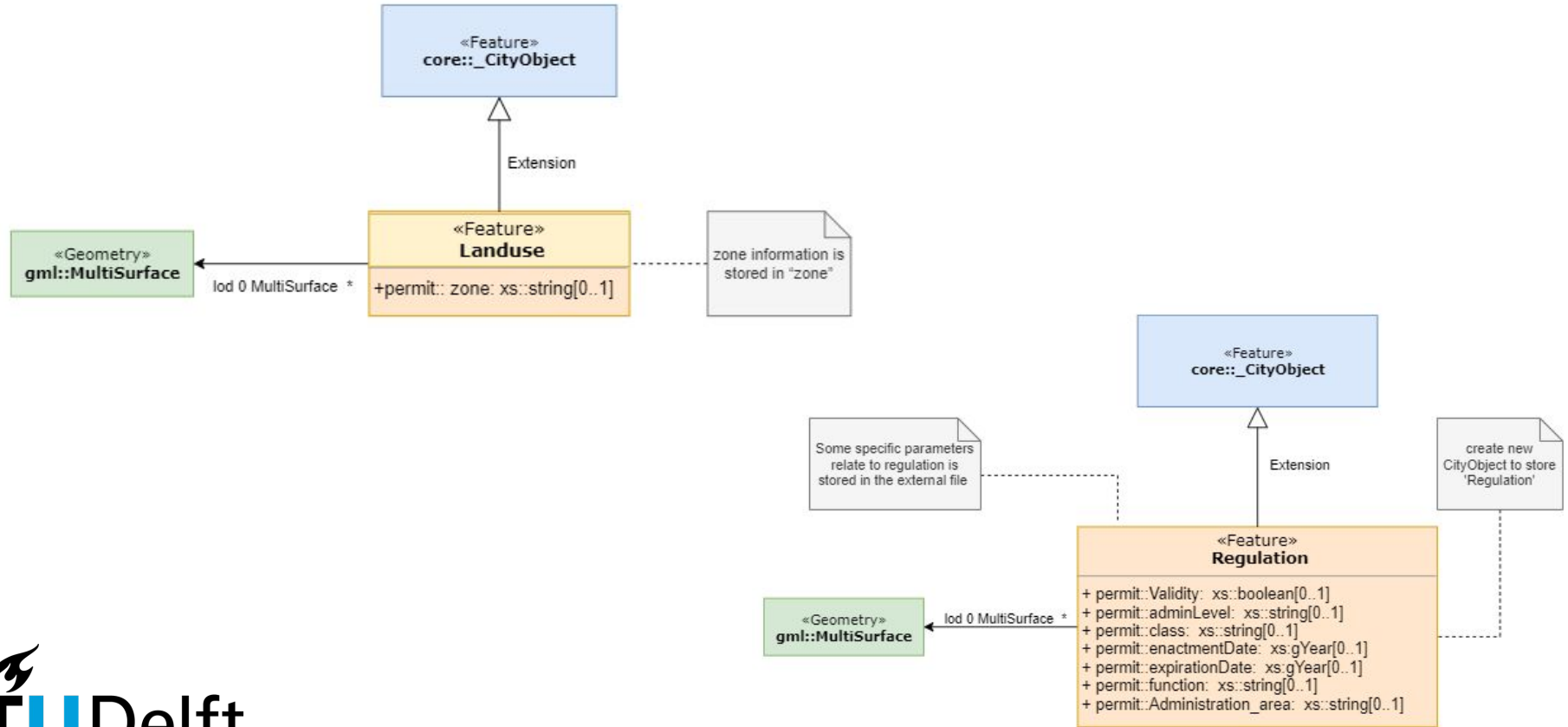
The developed CityJSON extension



The developed CityJSON extension



The developed CityJSON extension



The developed CityJSON extension

Store the regulation in data model and write specification for it

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:

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; <u>Transport</u> ;
+permit::function	Usage of the regulation (e.g Calculate the appropriate parking spaces for each building)
+permit::Administration_area	Where does it apply
<u>toplevel</u>	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.

```
{
  "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" },

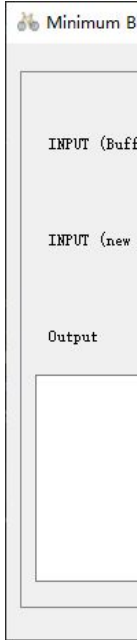
```

Schema and its validation

```
wujialundeMacBook-Air:~ Gallon0529$ python /Users/Gallon0529/Desktop/extension/extensions/validate-extension.py /Users/Gallon0529/Desktop/extension/extensions/test_schema.json
```

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

How to Tool pe



Delft

d
analysis

e)

ntain

Limitation

- The automation of the inspection process needs to be improved (Running speed and ability)
- Fewer data sources and fewer test datasets (especially new Buildings)
- The shape and structure of new buildings is simplified (need the model of the new building from the construction developer, which can be in BIM format or 3D model format)

Contribution and application

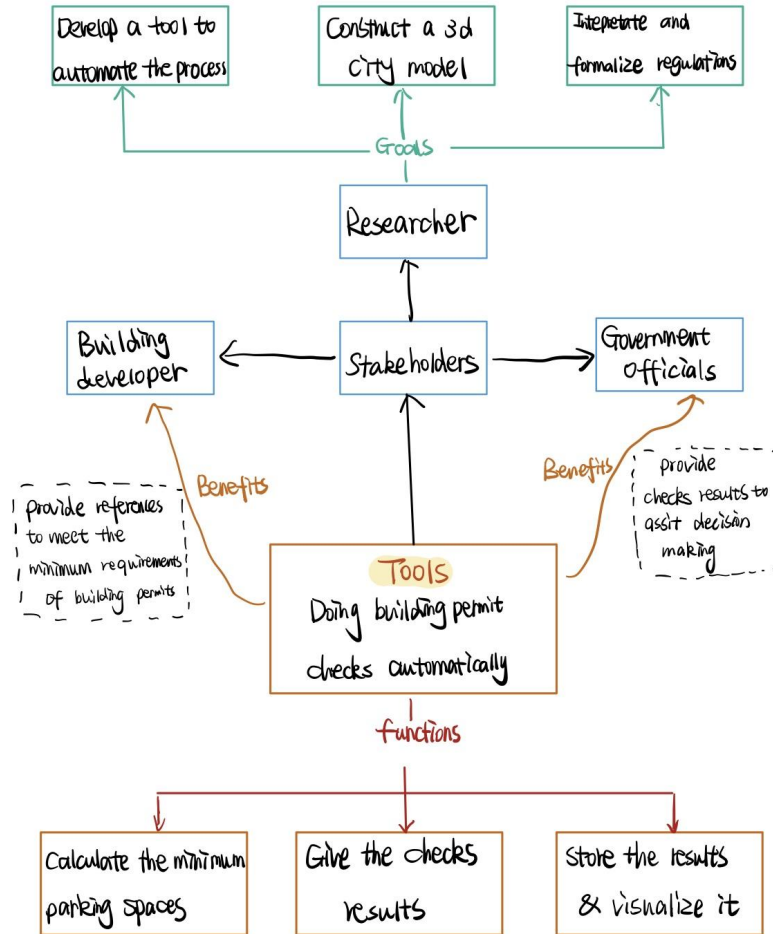
- Construct a 3D city model in form of CityJSON extension
 - Test the whole workflow with use case regulation and design a GUI program tool
 - Improved from previous work
 - Can be applied in other cases
- Extends the scope
 - New datasets for test
 - Better interoperability than BIM

- Use of CityJSON extension to extend the 3d city model
 - Provides convenience for analysis and data processing

*Checked the building permit
through GeoBIM
(Noardo et al., 2020c)*

*Urban planning CityGML ADE
(Ishimaru et al., 2020)*

Application



Conclusion

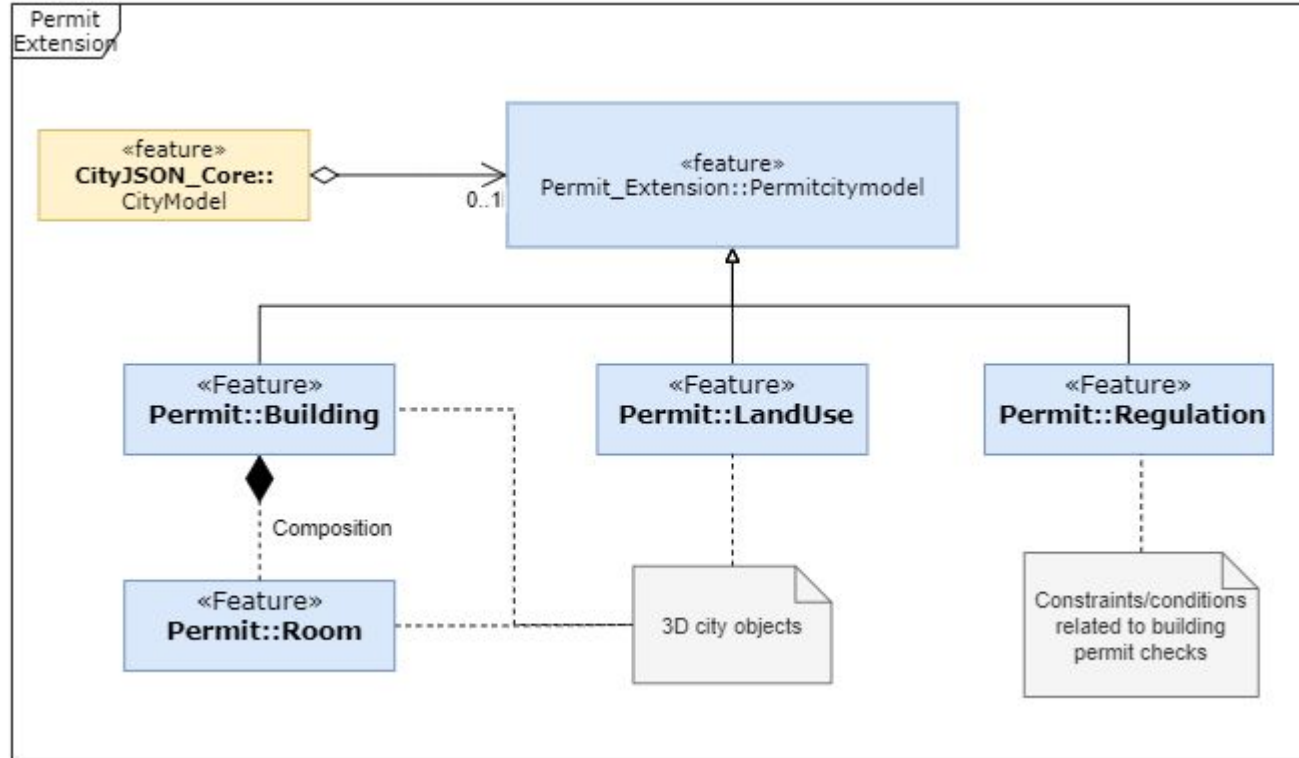
- Overview of research
- Future work

Conclusion

“ How

Select
the in

Interpr
forma
the r



ol

on
is

Future work

Applied to more cases (other regulations and cities)

- Other regulations (e.g environmental aspect)
- expand the test area

Improve the construction of 3D data model

- Test on the consistency and stability
- Data visualization improvement

Improve the performance of the tool

- More powerful computer machine
- Improve the code
- Better pre-processing of the data

THANK YOU!
谢谢

References in this slides:

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

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.

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

B. Dukai. 3D-inputgegevens voor geluidssimulaties gegenereerd uit bestaande landsdekkende datasets. Geo-Info, 15(6):8–12, 2018. ISSN 1572-5464.

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.

<https://limyyj.github.io/cesium-cityjson/>

Does anyone have any questions?