Project Title:
THE FOF-DESIGNER:
DIGITAL DESIGN SKILLS FOR FACTORIES OF THE FUTURE

Project Acronym:
DigiFoF

Grant Agreement number:
2018-2553 / 001-001

Project Nr. 601089-EPP-1-2018-1-RO-EPPKA2-KA

Subject:
ULBS_02 Sibiu – Smart city modelling (ADOxx)

Dissemination Level:
Public

Lead Organisation:
ULBS

Project Coordinator:
ULBS

Contributors:
All Partners

Reviewers:

Revision	Preparation date	Period covered	Project start date	Project duration
V1	January 2019	Month 1-6	01/01/2019	36 Months

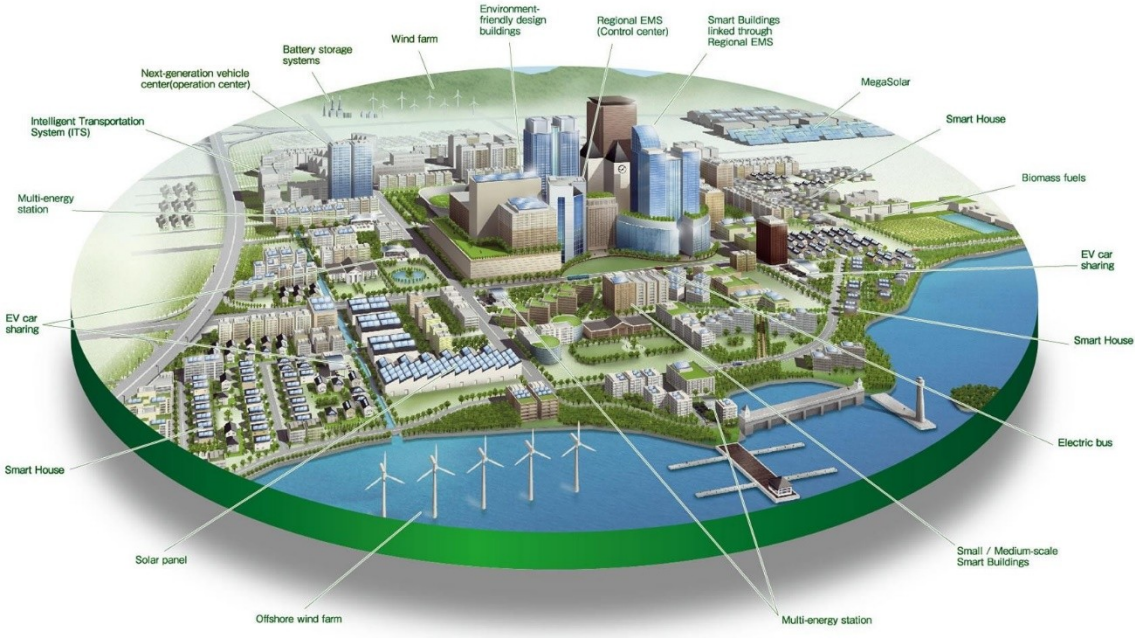
This project has received funding from the European Union's EACEA Erasmus+ Programme Key Action 2 - Knowledge Alliances under the Grant Agreement No 2018-2533 / 001-001

Table of content

1 Smart City

1.1 Exercise I: Smart City Modelling

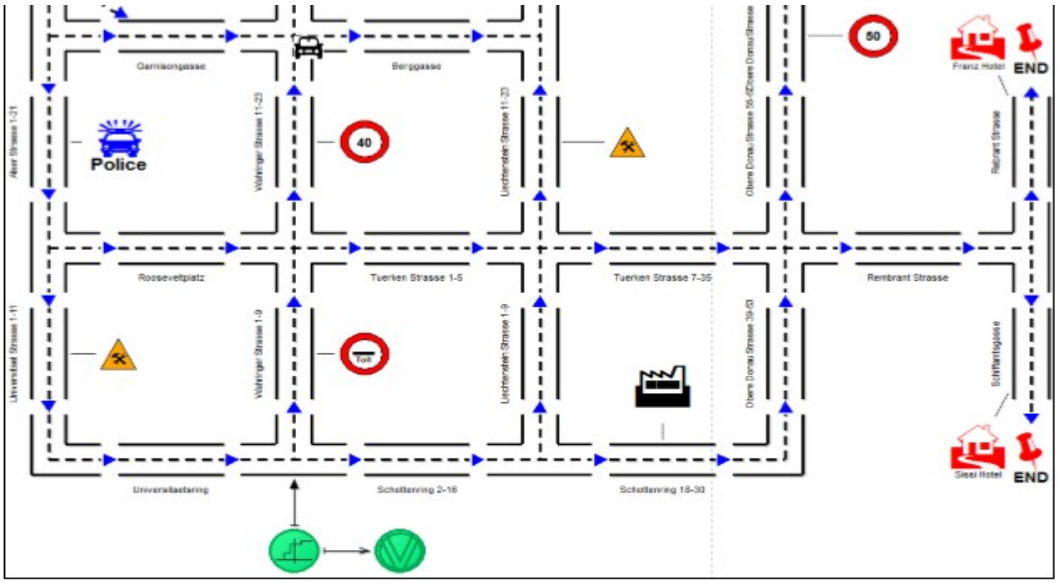
1.1.1 Smart City Scenario



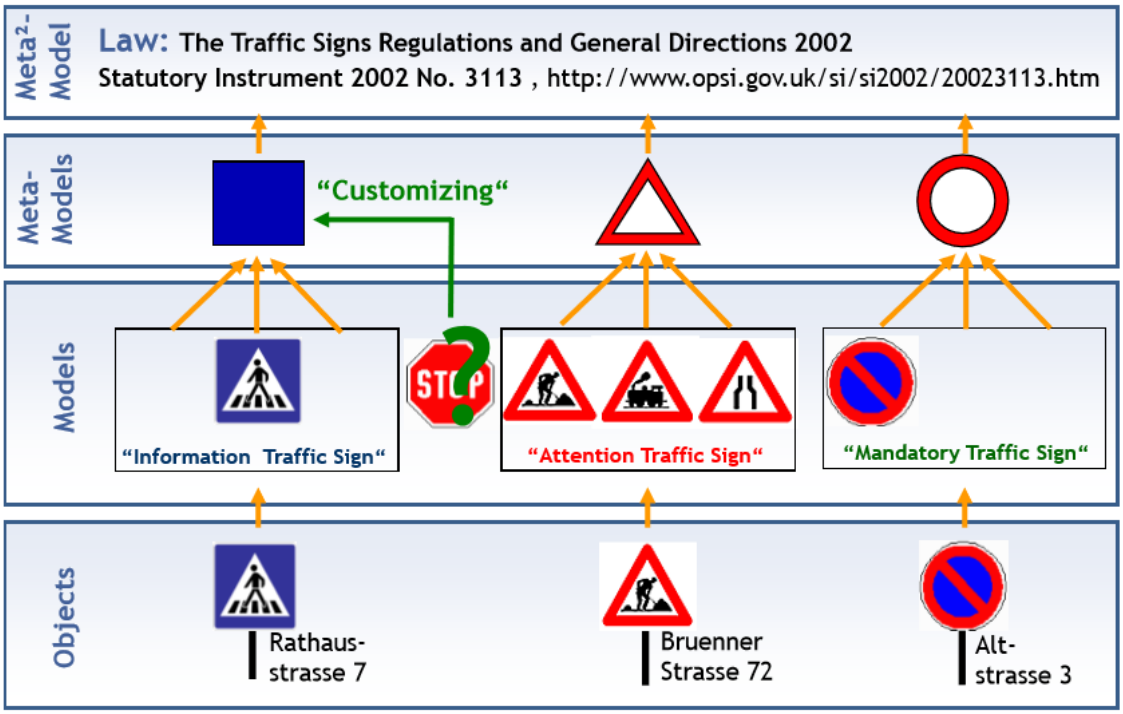
1 Smart Traffic

- Drivers spend 30 hours per year in traffic jams
- 30 percent of the city's traffic congestion is caused by Drivers looking for a place to park
- Annual cost of traffic congestion in the United States alone adds up to $87.2 billion in wasted fuel and lost productivity
- Internet of Things enables:
  - ✓ Establishment of an infrastructure including, mobile devices, cars, LAN, WAN, GPS
  - ✓ Traffic Light Assistants in cars
  - ✓ Smart Traffic Lights: The MARLIN project

Planning a Smart City



What is Metamodeling?



I. How to build SmartCity models?

Find the right abstraction level and use metamodeling to reproduce the reality for a specific purpose

II. How to analyze SmartCity models?

Use the knowledge captured in the models to answer non-trivial questions

III. How to simulate SmartCity models?

Combine static and dynamic SmartCity models to simulate complex scenarios

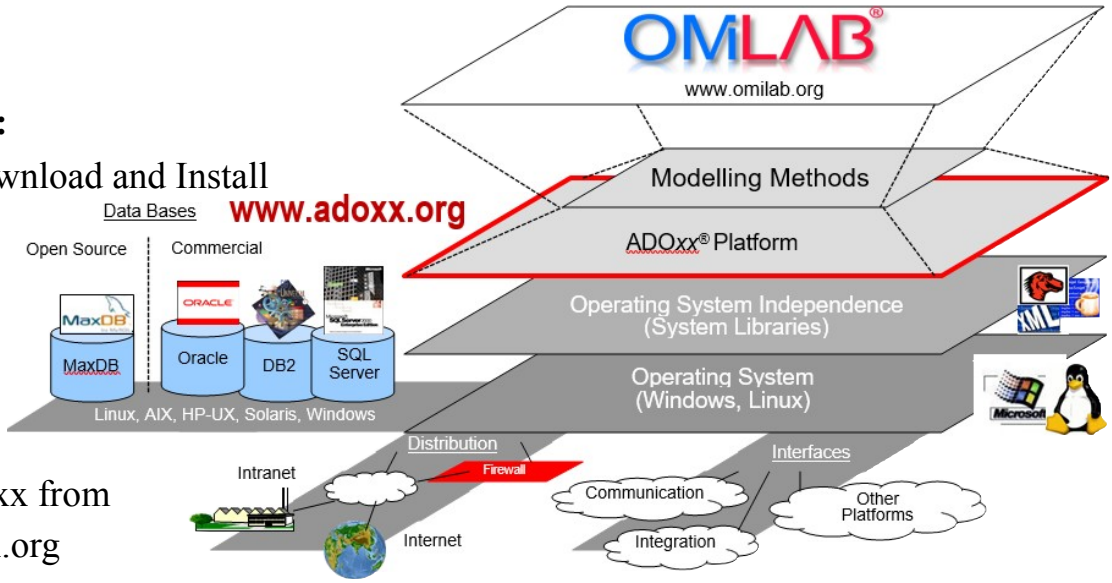
1.1.2 Smart City Modelling Task Overview

- Task 0: Introduction to ADOxx & SmartCity modelling
- Task 1: Create a new Modelling Class
  - A) Create Class „Speed Limit Sign“
  - B) Create a class attribute „speed limit“ of type integer
- Task 2: Realize a static graphical visualization
  - A) „Speed Limit Sign“ with a static value, e.g., 50 km/h
  - B) Create new SmartCity models using the Speed Limit Sign class
- Task 3: Realize a dynamic graphical visualization
  - A) Realize an attribute-dependent visualization for the „Speed Limit Sign“
  - B) Create new SmartCity models using the new Speed Limit Sign
- Task 4: Realize a sensor for the Common Air Quality Index (CAQI)
  - A) Create a class “CAQI“ with an enumeration for the attribute ”index”: Very Low | Low | Medium | High | Very High
  - B) Realize an attribute-dependent visualization based on the current index status
- Task 5: Create a new Relation Class
  - A) ”SensorRelation” class that connects the CAQI sensors with Road
  - B) Create new SmartCity models using the SensorRelation

1.1.3 Introducing to ADOxx and Smart city Modelling

Steps:

1. Download and Install ADOxx from adoxx.org



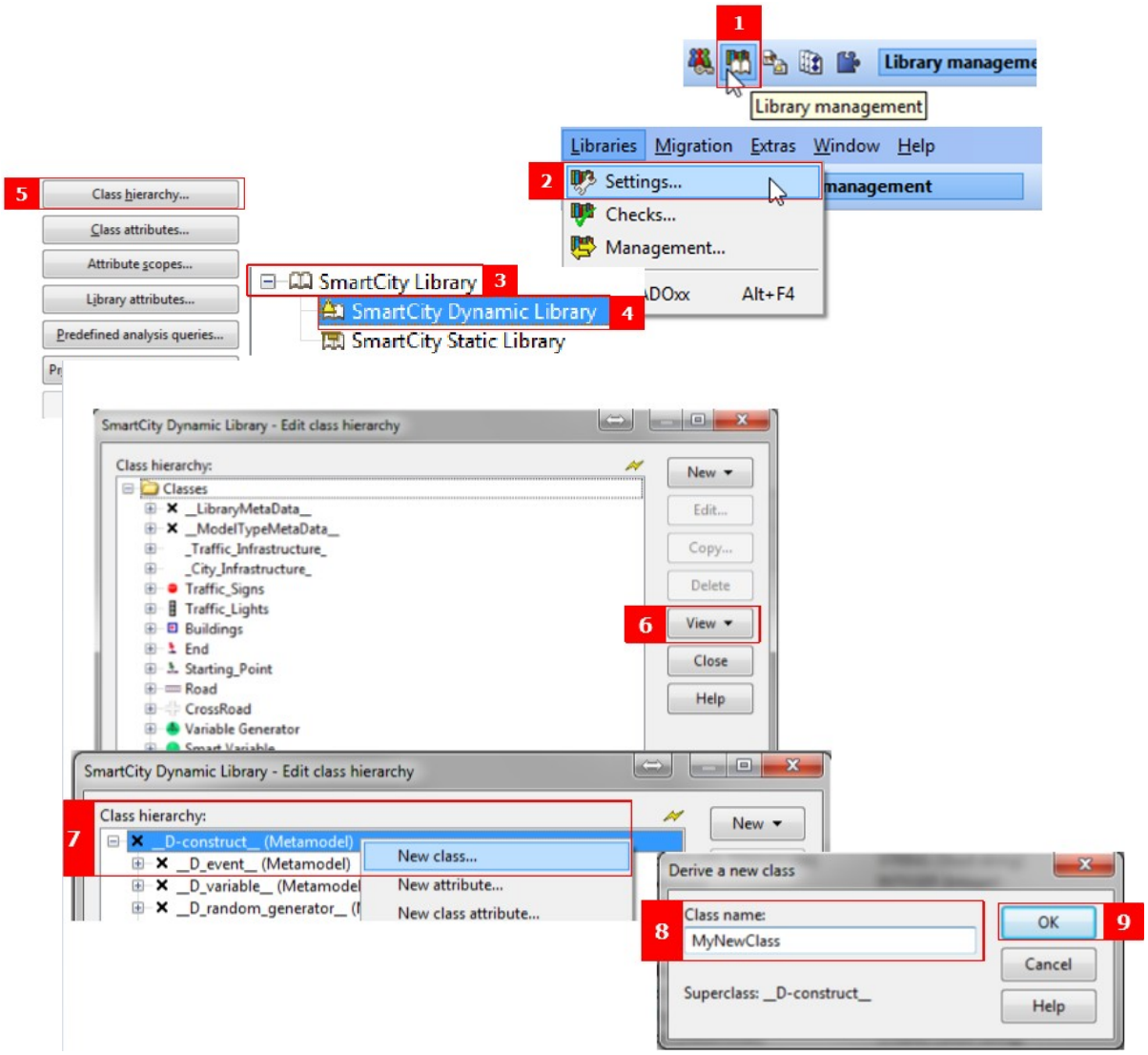
2. Import the empty library from the Getting Started Package
3. Create Classes and Relationclasses
4. Add Attributes to Classes and Relationclasses
5. Define Graphical Representation
6. Define Model Types
7. Define Users and link them to your library
8. Start the Modeling Toolkit with the new user

1.1.4 CREATE A NEW MODELLING CLASS

Login to the Development Toolkit

1 Switch to *Library management* component
2 Menu -> *Libraries* -> *Settings*
3 Click on *SmartCity Library*
4 Click on SmartCity Dynamic Library
5 Click on "Class hierarchy"

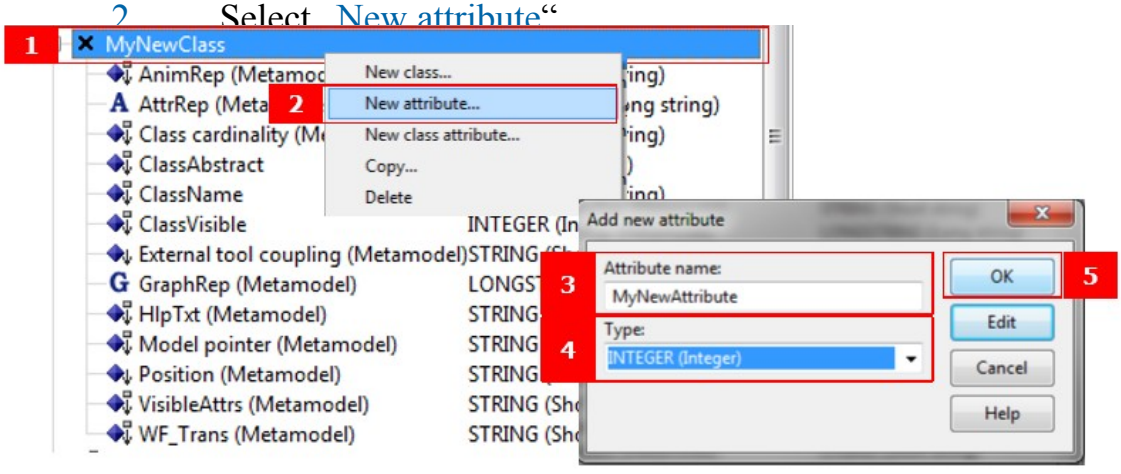
6 Adjust the View; Enable “Metamodel” and “Class hierarchy”
7 Right-click on “D_Construct” -> “New class…”
8 Enter a name for the new class
9 Click on “Ok”



How to create new attributes

The class hierarchy should now comprise your new class

1 Right-click on a class
2 Select „New attribute“

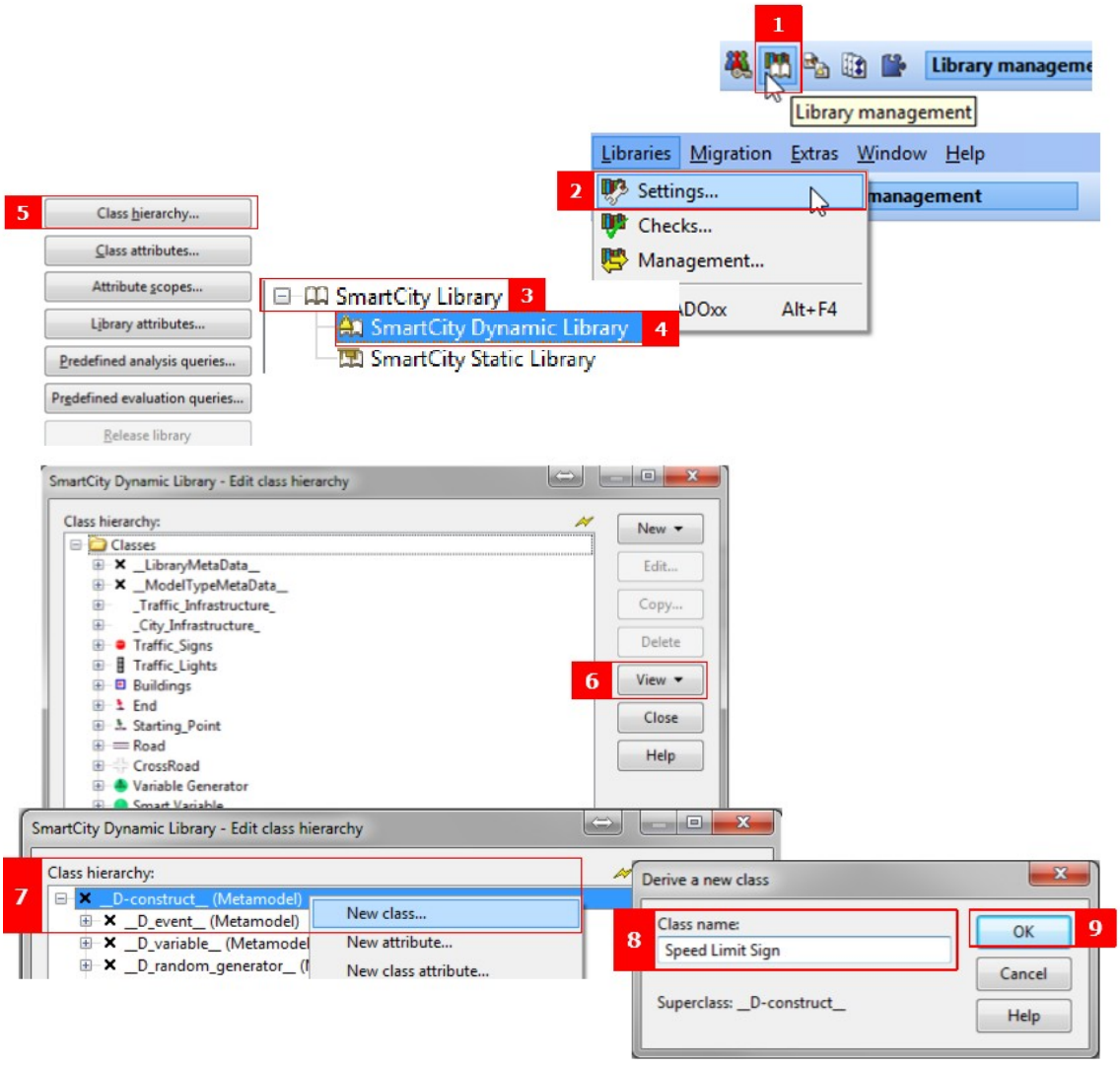


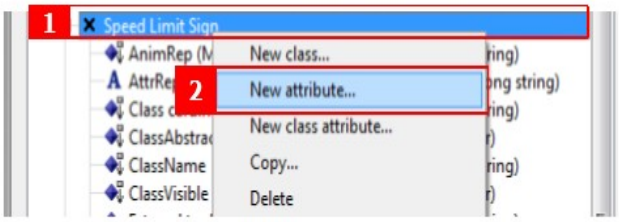
\\

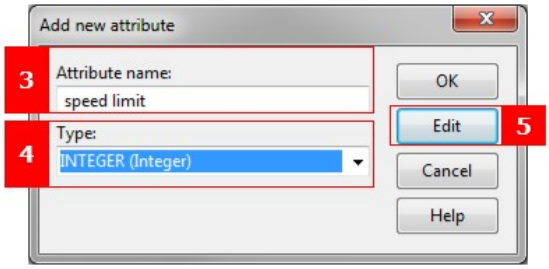
Exercise 1: CREATE A NEW MODELLING CLASS

a Create Class "Speed Limit Sign"

1. Click on „Library Management"
2. Click on „Settings"
3. Click on „SmartCity Library"
4. Select the „SmartCity Dynamic Library",
5. Click on „Class hierarchy"
6. Adjust the view; Enable "Metamodel" and "Class hierarchy"
7. Right-click on "_D_Construct_ -> "New class…"
8. Enter "Speed Limit Sign" as name for the new class
9. Click on "Ok"







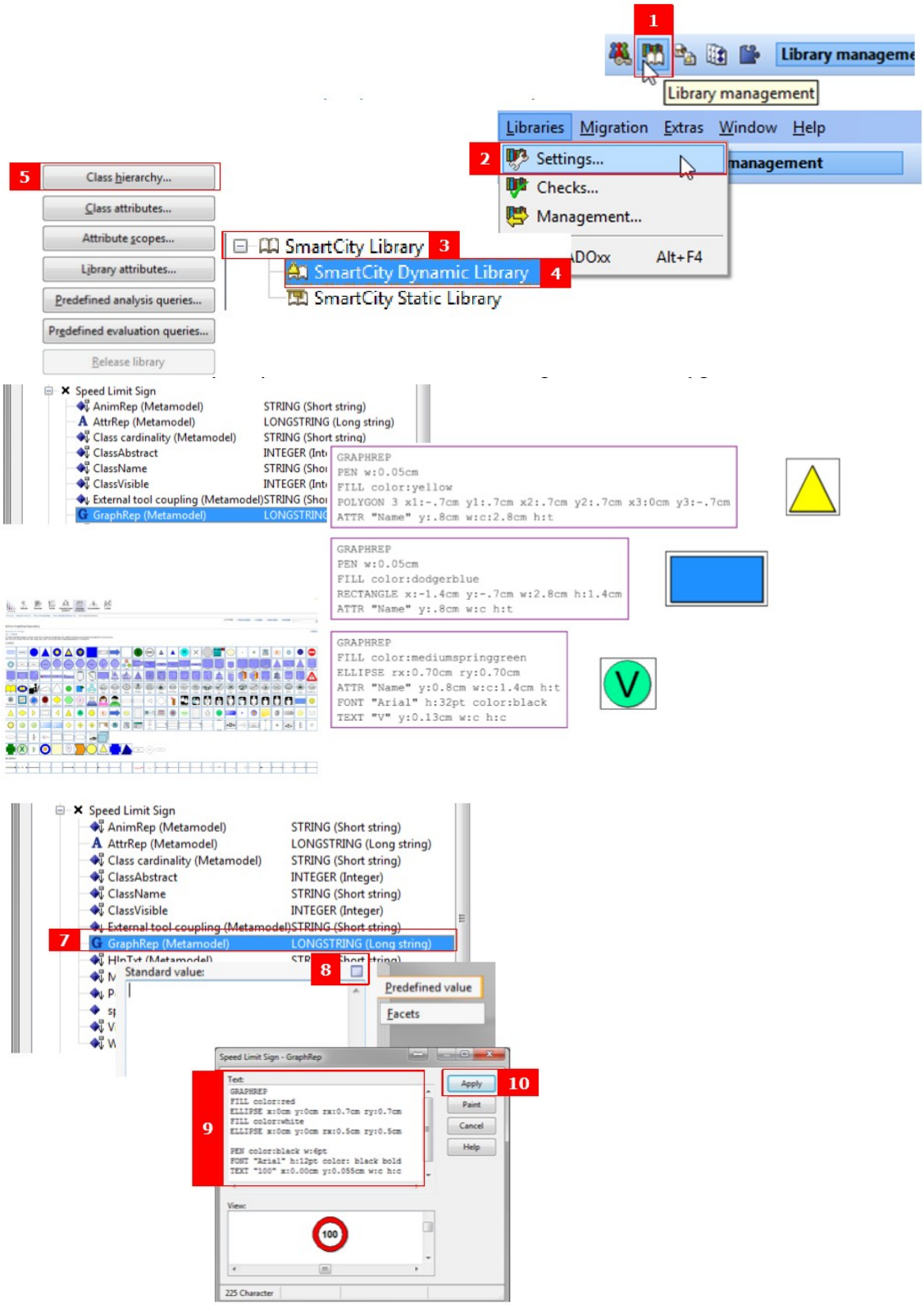
1.2 Exercise 2: REALIZE A STATIC GRAPHICAL VISUALIZATION

a) Firstly:

1 Login to the Development Toolkit
2 Switch to "Library management" component
3 Menu -> "Libraries" -> "Settings"
4 Click on "SmartCity Library"
5 Click on "SmartCity Dynamic" Library
6 Click on "Class hierarchy"
7 Edit the GraphRep attribute to create Rectangles, Lines, Polygons, …
8 Double-Click on "GraphRep" attribute of the class "Speed Limit Sign"
9 Click on "Dialog"
10 Copy this code into the „Text" area:

```
GRAPHREP FILL color:red
ELLIPSE x:0cm y:0cm rx:0.7cm ry:0.7cm
FILL color:white ELLIPSE x:0cm y:0cm rx:0.5cm ry:0.5cm
PEN color:black w:6pt
FONT "Arial" h:12pt color: black bold
TEXT "100" x:0.00cm y:0.055cm w:c h:c
```

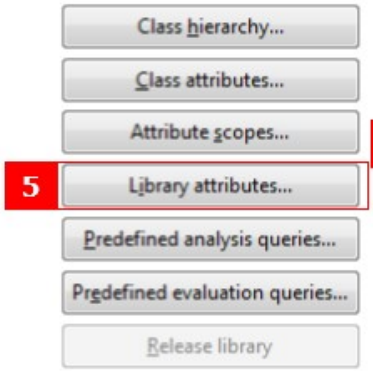
11. Press "Apply", "Close" and "Save changes!"



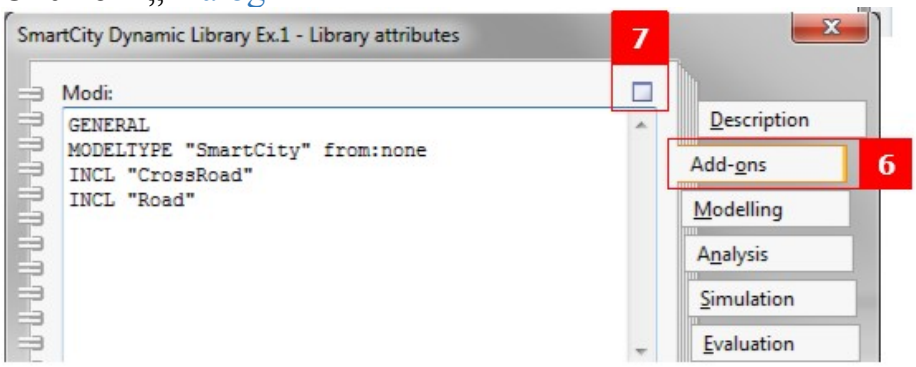
b Create new SmartCity models using the Speed Limit Sign class

Use previous four steps and:

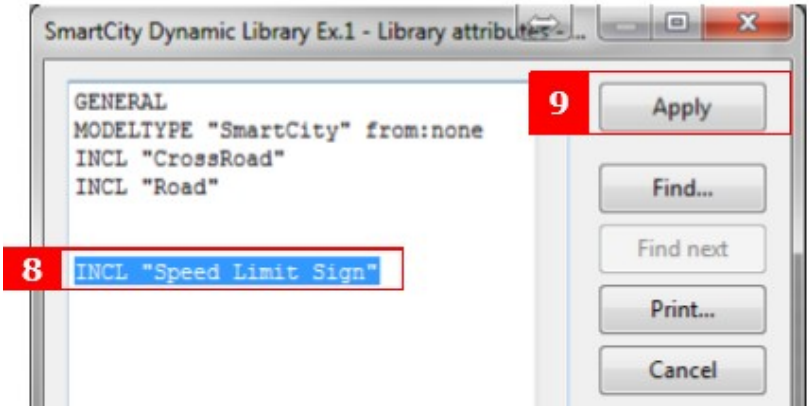
5 Click on „Library attribute



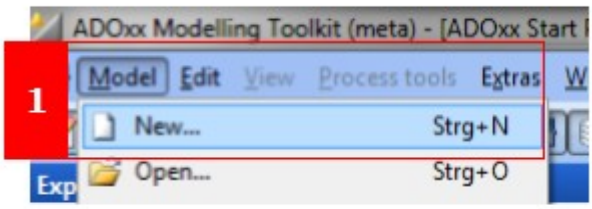
6 Click on „Add-ons“
7 Click on „Dialog“



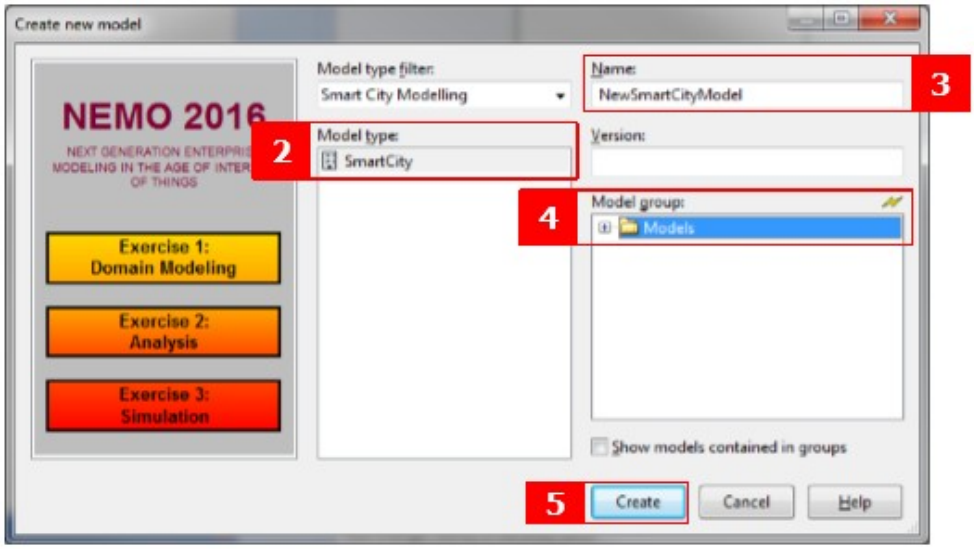
8 Add the text INCL “Speed Limit Sign”
9 Click on „Apply”

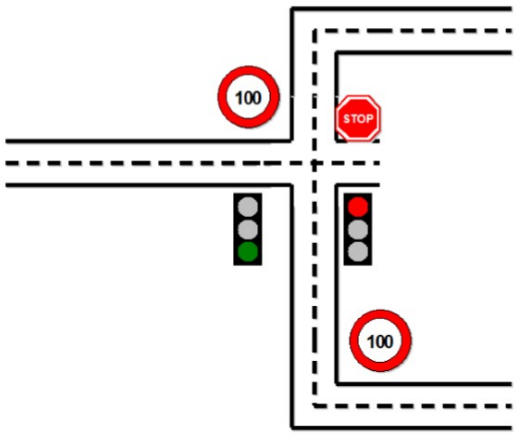


Now open the “ADOxx Modeling Toolkit”
1 Click on „Model“ => “New”



2. Select „SmartCity" model type
3. Define a name for the model
4. Click on „Models"model group
5. Click on "Create"





1.3 Exercise 3: REALIZE A DYNAMIC GRAPHICAL VISUALIZATION

a Realize an attribute-dependent visualization for the „Speed Limit Sign". Depending on the speed limit attribute value, the visualization should adapt

Solution:

I Change the GraphRep code in order to realize attribute-dependent functionality

1. Open the class hierarchy of the library
2. Double-click on „GraphRep“attribute of the Speed Limit Sign class
3. Replace the following lines of code at the end of the existing GraphRep code **Delete** this line:

```
TEXT "100" x:0.00cm y:0.055cm w:c h:c
```

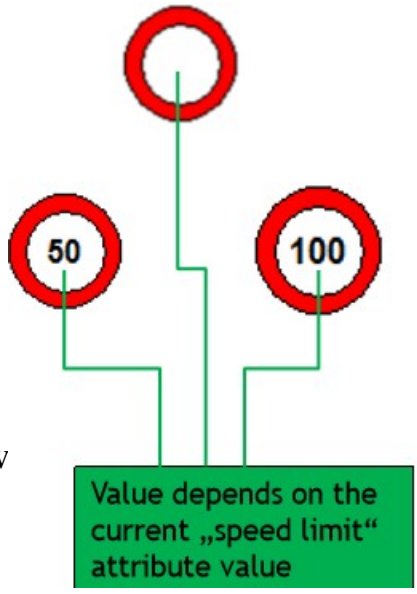
Add this line:

```
ATTR "speed limit" x:0.00cm y:0.055cm w:c h:c
```

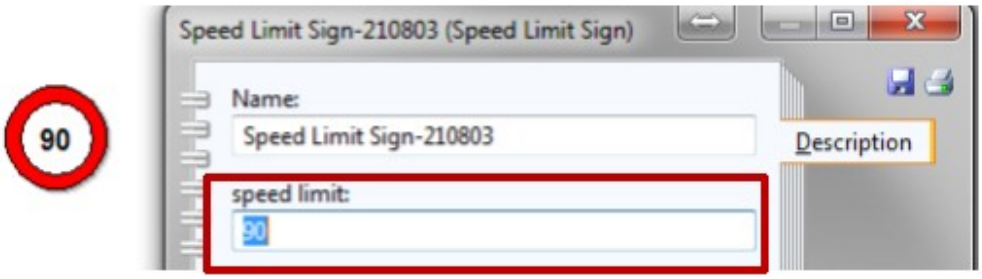
4. Click „Apply“

II Add the attribute „speed limit“ to the Speed Limit Sign Notebook

1 Open the class hierarchy of the library
2 Double-click on „AttrRep“ attribute of the Speed Limit Sign class
3 Add the following lines of code to the end of the existing AttrRep code `ATTR "speed limit"`
4 Click „Apply“
5 Save all changes to the library



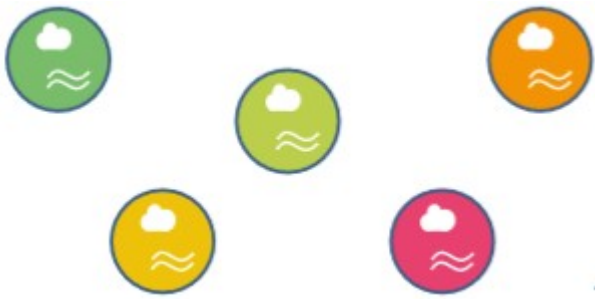
b Create new SmartCity models using the new
1. Open the ADOxx Modelling Toolkit
2. Create a new SmartCity model
3. Create Speed Limit Sign objects
4. Double-Click on a Speed Limit Sign to open it‘s Notebook
5. Change the speed limit attribute and look how the visualization adopts



1.4 Exercise 4: REALIZE A SENSOR FOR THE COMMON AIR QUALITY INDEX (CAQI)

What is Common Air Quality Index (CAQI)?
To present the air quality situation in European cities in a comparable and easily understandable way, all detailed measurements are transformed into a single relative figure: the Common Air Quality Index (CAQI). OMiLAB Smart City has the ability to provide an hourly index, which describes the air quality today, based on hourly values and updated every hour. The CAQI index classifies the air pollution according to the following 5-scale color legend:

Pollution	Color
Very low	
Low	
Medium	
High	
Very high	



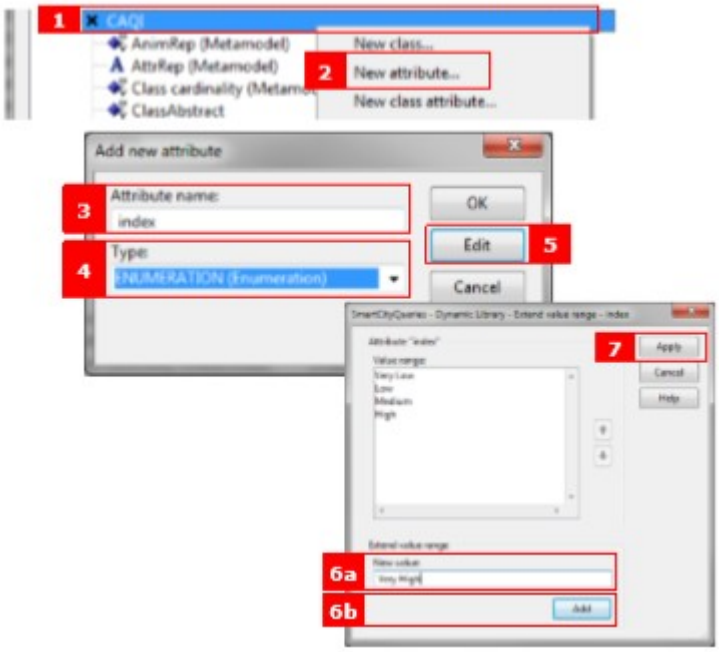
What do you need to do?
Create a new Modelling Class named CAQI with an enumeration attribute ‚index' that determines the current common air quality attribute index. Furthermore, use this attribute to realize an attribute-depended visualization for the CAQI class.

1. Click on „Library Management"
2. Click on „Settings"
3. Click on „SmartCity Library"
4. Select the „SmartCity Dynamic Library"
5. Click on „Class hierarchy"
6. Adjust the View — Enable "Metamodel" and "Class hierarchy"
7. Right-click on "_D_Construct_ -> "New class…"

8. Enter “CAQI” as name for the new class
9. Click on “Ok”

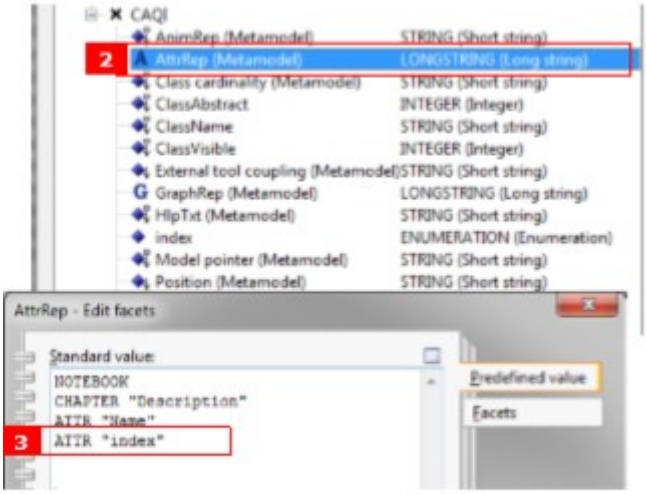
Using the new Modelling Class:

1 Expand the Speed Limit Sign class
2 Right-click on “CAQI” => “New attribute…”
3 Enter “index” as name for the new attributes
4 Select “ENUMERATION” as type
5 Click on “Edit”
6 In Sequence
   a. Add a „New Value“ Very Low | Low | Medium | High | Very High
   b. Press „Add“
7. Press „Apply“, „Close” and „Save Changes“



What is next? You need to add „index” to the CAQI Notebook:

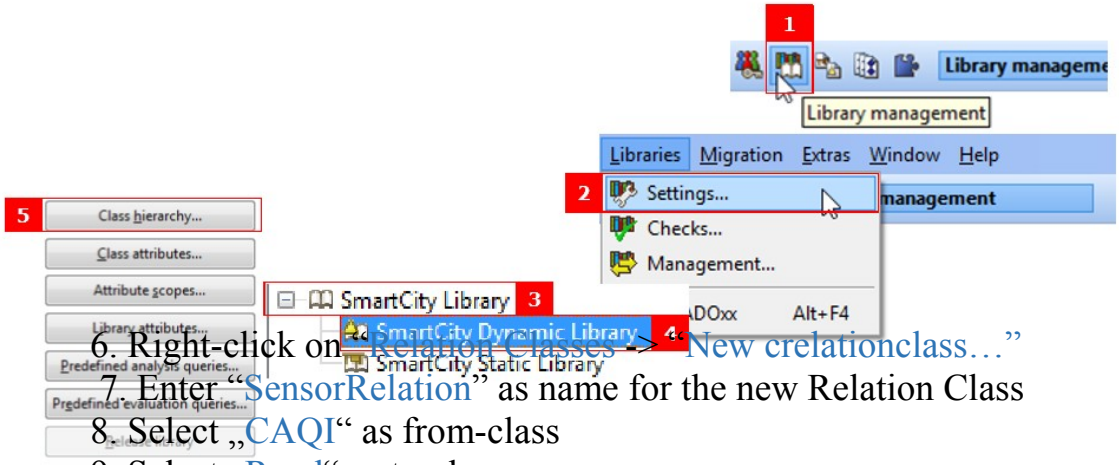
1 Open the class hierarchy of the library
2 Double-click on „AttrRep“ attribute of the CAQI class
3 Add the following lines of code to the end of the existing AttrRep code ATTR “index"
4 Click „Apply“
5 Save all changes to the library



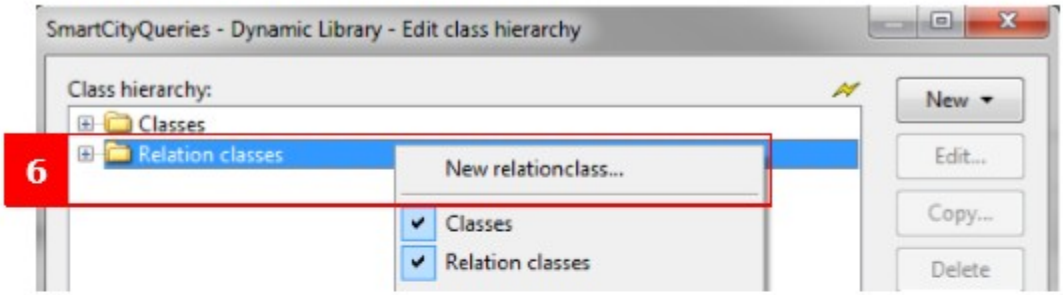
To be continued in the next lesson:
CREATE A NEW RELATION CLASS (Sensor relation)

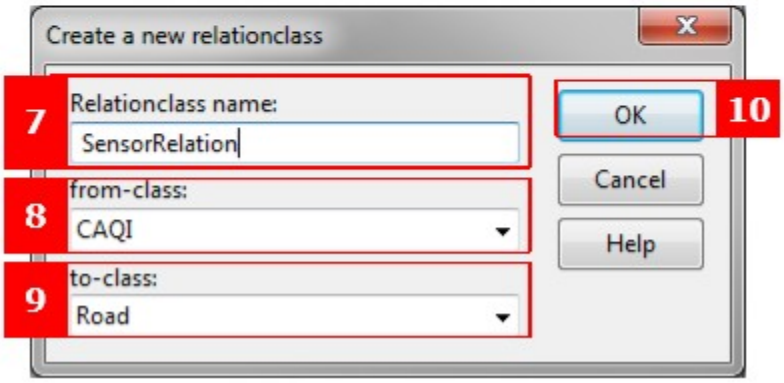
1.5 Task 5: Create a new Relation Class A) SensorRelation class that connects the CAQI sensors with Road

1. Click on „Library Management“
2. Click on „Settings“
3. Click on „SmartCity Library“
4. Select the „SmartCity Dynamic Library“
5. Click on „Class hierarchy“



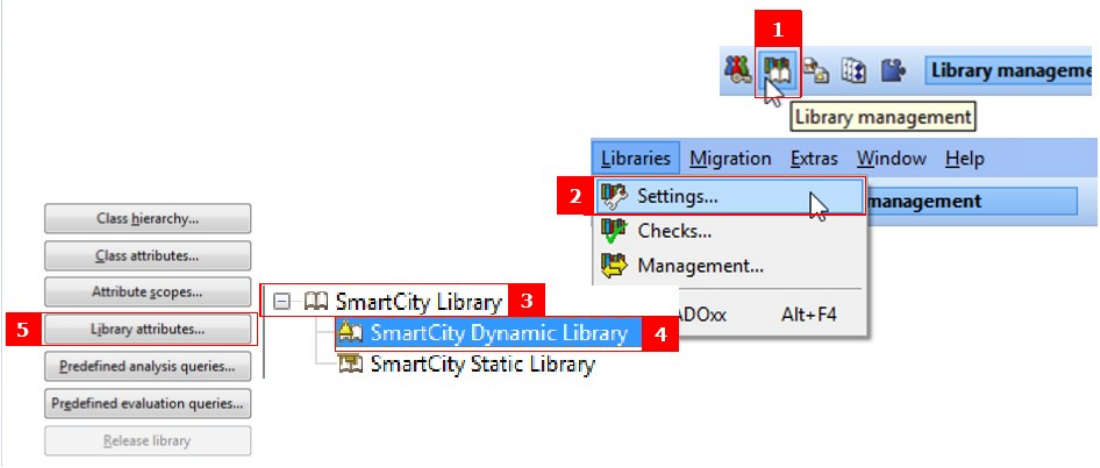
6. Right-click on „Relation Classes -> New crelationclass…“
7. Enter “SensorRelation” as name for the new Relation Class
8. Select „CAQI“ as from-class
9. Select „Road“ as to-class
10. Click on “Ok” , “Close” and “Save Changes”



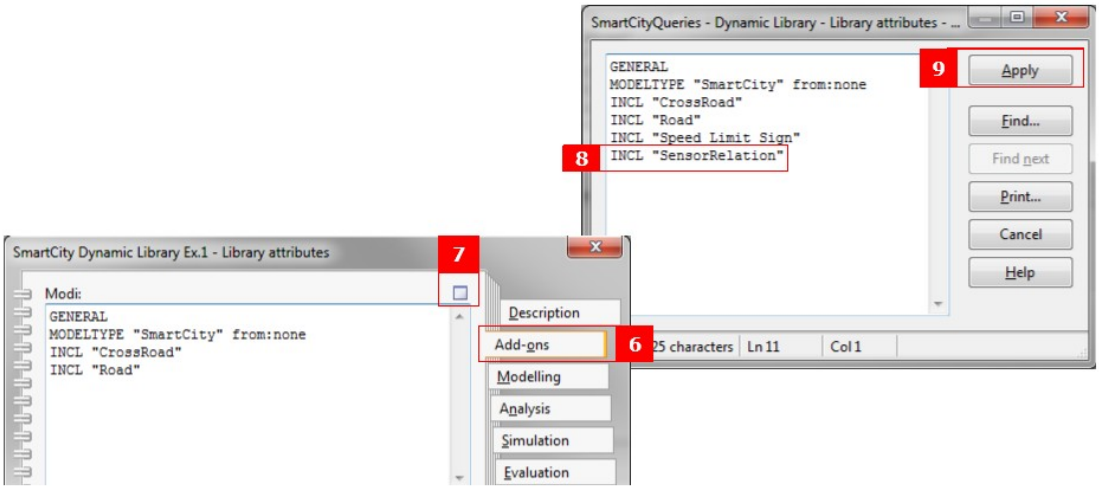


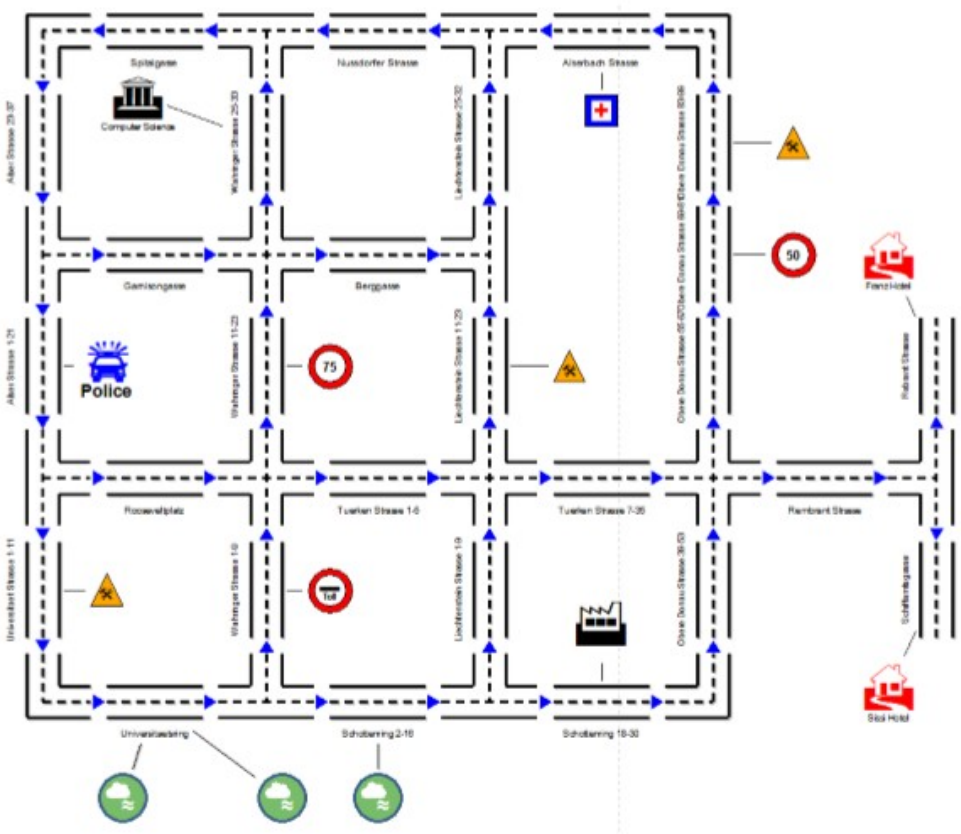
Add the SensorRelation class to the modeltype

1. Click on „Library Management“
2. Click on „Settings“
3. Click on „SmartCity Library“
4. Select the „SmartCity Dynamic Library“,
5. Click on „Library attributes“



6. Click on „Add-ons“
7. Click on „Dialog“
8. Add the text INCL “SensorRelation”
9. Click on „Apply”

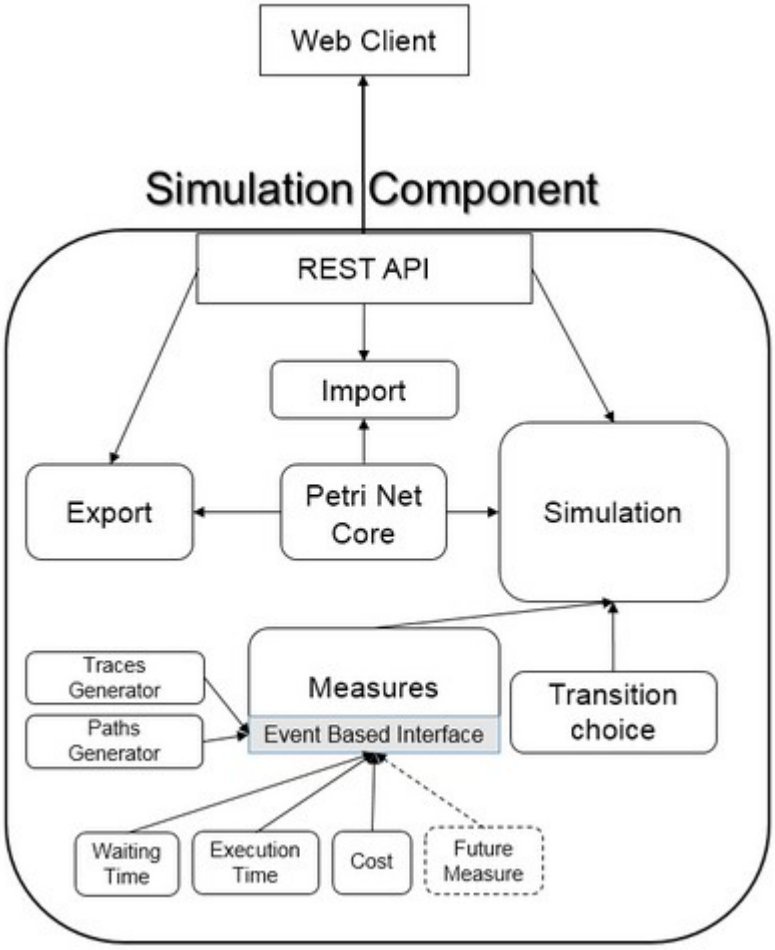




2 ADOxxWEB SIMULATION

The ADOxxWEB Simulation provides a fast and extendible service capable of simulating business process executions. The service is provided through a REST interface that uses the model as input to simulate and add additional simulation parameters. The service returns the result as an XML file. The service is focused on business process models, but is flexible enough to be adapted and to simulate any kind of models structure.

How it Works
The service internally uses a Petri Net structure in order to represent the model to simulate. The semantics of the simulation, in this way, relies on Petri Net strong formalism. In the case of Business Process models, the simulation uses state-of-the-art mapping rules in order to import the model as a Petri Net. This import phase can also be skipped, providing a Petri Net model in PNML standard format directly to the simulator. This gives the possibility to simulate any kind of model that can be expressed as a Petri Net, also generated by external systems.
The services expose REST APIs in order to be easily integrated in every system. Examples of such integration have been provided in ADOScript format for ADOxx and as an MFB template for Adonis NP.
The Simulation service also provides an asynchronous HTML/JavaScript client in order to use and test the service without integrating it in a tool.



The **Petri Net Core Module** is the component that contains the main logic of a Petri Net and manages its semantics. The simulation service uses this component in order to evaluate which transition can be enabled at each step.

The **Import Module** is an easy to extend component that is able to automatically recognize the format of the provided model and converts it into the internal Petri Net structure. It manages separately the logic of document parsing, and of object mapping, in order to reuse the same mapping logic for multiple file formats (like in the case of BPMN and ADOxx BPMN).

The **Export Module** is for diagnostics only. It gives the possibility to export the internal Petri Net structure in the PNML standard format in order to be visualized in any supported editor.

The **Simulation Measures Module** is an easy to extend component that gives the possibility to define listeners for the simulation event. Each listener produces a measure or a result from a single simulation, like a trace, a path, the waiting times or the execution costs. The resulting indexes can then be collected in a special container in order to calculate some final indexes (like average values).

The **Simulation Transition Choice Module** is the component that performs the choice of the transition to execute between the available one. The module provides a base

mechanism that performs a fear choice between parallel transitions and a user defined probabilistic choice between concurrent transitions. The base mechanism has also been extended in order to support dynamic probability evaluation using a scripting system.

The **Simulation Module** is the component that manages all the simulations, invoking the functionalities of the measures module and of the transition choice. It is also responsible for the generation of the simulation output in a structured XML format.

Workshop: Create your own scenario for a known street or neighborhood and implement it in AdoXX.

3 Smart Parking:

- **Project presentation**
- **Integration with/in Smart City project**
- **Implement Modelling Language**
- **Classes**
- **Testing the Modelling Language**

4 References