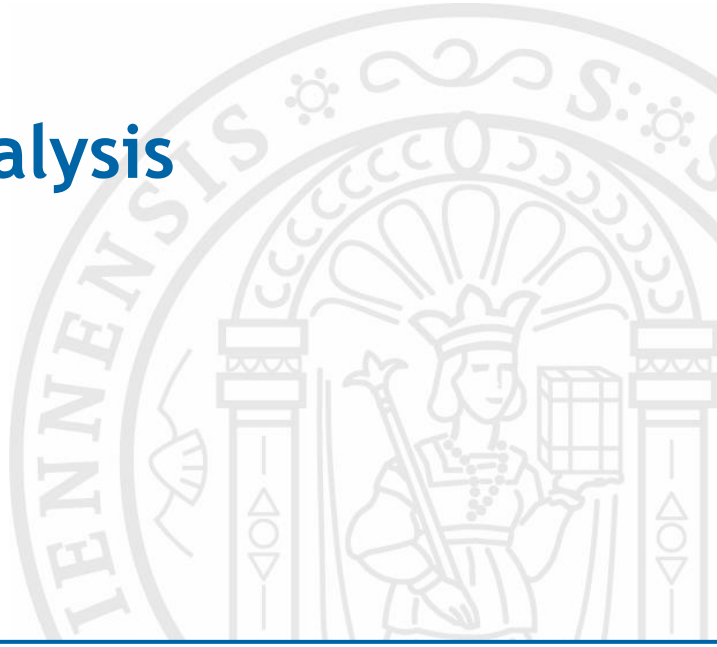
Smart City Exercises

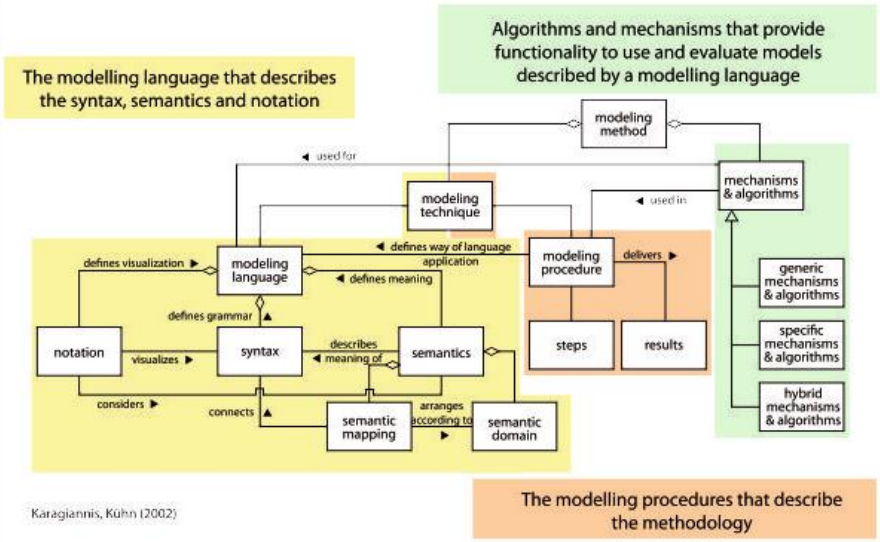
Exercise II: Smart City Analysis



INTRODUCTION

Modeling Method – Mechanisms and Algorithms

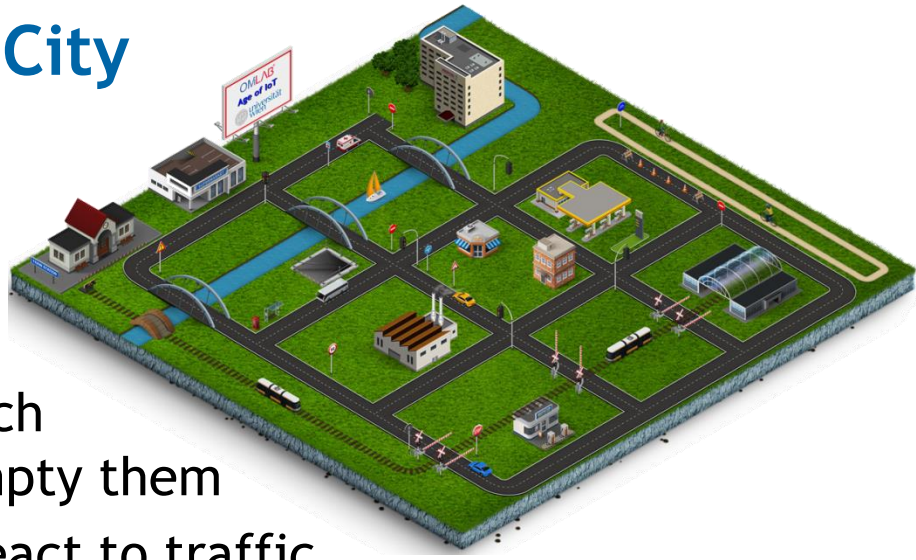
- **Generic** mechanisms and algorithms can be applied to **arbitrary modelling languages**
- **Specific** mechanisms and algorithms can be applied to **particular modelling languages**
- Both ensure an adequate **user experience**



Mechanisms and Algorithms - Analysis

- Analysis assists in utilizing the complex relationships within a model or between multiple models
- Manual model analysis can be costly
- An analysis technique is chosen for a certain purpose
  - Analysis supports the decision making process
  - Analysis assists in understanding, visualizing and communicating structured and unstructured data
  - Contextualization, verification, validation, conformance, and consistency checking

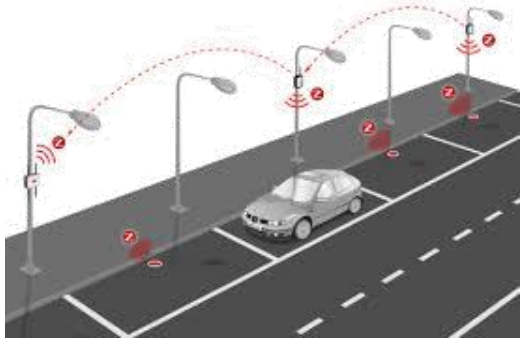
Analysis Case Study: Smart City



- Smart City Scenarios
  - **Smart waste management:** **Find** trash cans before they reach their maximum capacity and empty them
  - **Smart street lights:** **Find** and react to traffic participants and to the environment by changing lighting
  - **Smart open space:** **Find** and cultivate micro agricultures
  - **Smart public transportation:** **Find** and register stops based on public demand

- Two quick introductory scenarios in more detail
  - **Parking Scenario**
  - **Emergency Scenario**

Parking Scenario



www.libelium.com

- Searching for parking spots is an inefficient task. *"Many drivers spend on average 20 minutes while looking for a parking spot. This increases CO2 emissions and most important of all; people waste their time."* [1]

- Smart Parking
  - Improves the urban environment
  - Reduces pollution
  - Generates revenue
  - Optimizes human resources
  - Creates happiness
  - Is vandalism proof
  - Easy to maintain
  - Reduces accidents

[1] Amsterdam Smart City : http://amsterdamsmartcity.com/projects/detail/id/64/slug/smart-parking?lang=en

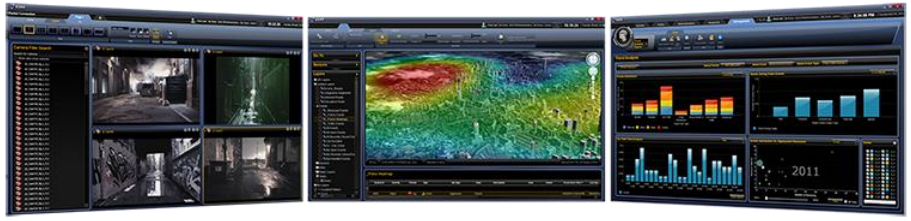
Parking Scenario: Traditional Approach (animated)

Parking Scenario: Smart Approach (animated)

Emergency Scenario

www.psfk.com

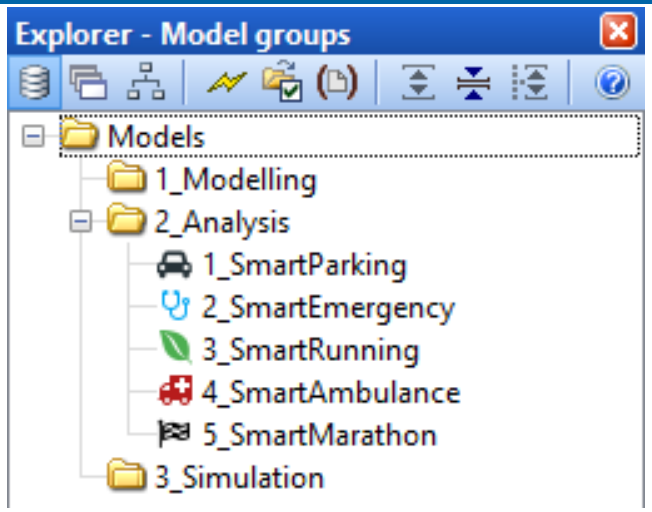
- There is a building on fire!
- Smart City response:
  - Notify the emergency services
  - Block public traffic into that area
  - Find people in danger by locating their smart devices
  - Configure smart buildings to guide people out of danger zones
  - Notify relevant people in proximity for quick first response (firefighters, doctors, policemen, …)



Emergency Scenario: Block Public Traffic (animated)

Emergency Scenario: Medical Response (animated)

Analysis Case Study: Smart City



- Exercise scenarios:
  - **Smart Parking:** Increase the quality of living with queries, e.g., find and book open parking spots
  - **Smart Emergency:** Save lives with queries, e.g., lower first aid response times
  - **Smart Track**: Increase public health with queries, e.g., find save and healthy running tracks
  - **Smart Ambulance**: Save lives with queries, e.g., find fast routes and adapt smart city infrastructure
  - **Smart Marathon**: Support runners and organizers of the Vienna City Marathon with queries, e.g., improve relay race coordination
  - And many more

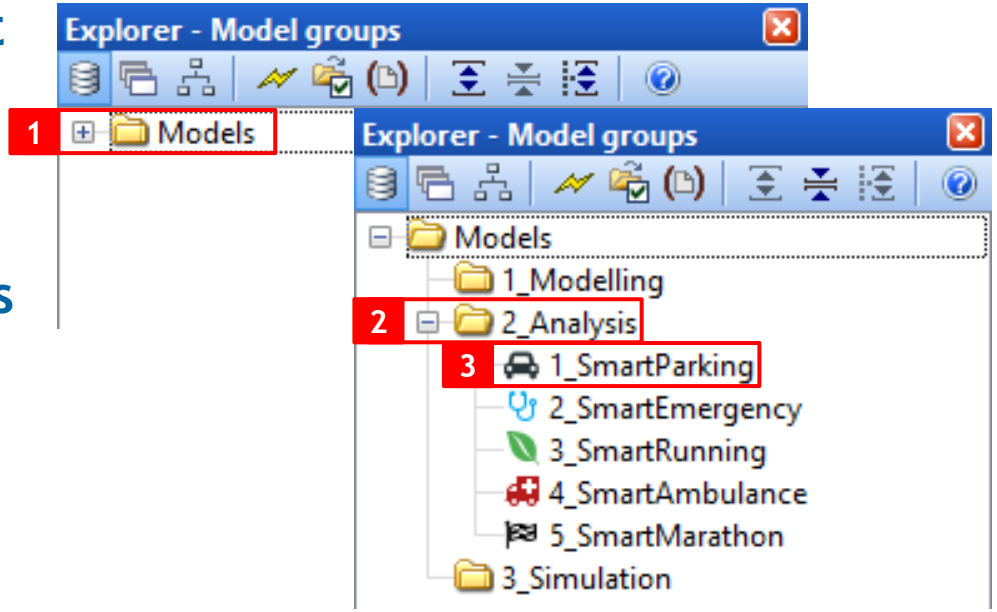
CLASSROOM EXERCISES

Exercise Structure

1. **Standardized queries**
   - **Basics: The query dialog and the query result window | Smart Traffic Signs**
   - **Demo: Querying classes and attributes | Smart Parking**
   - **Hands-on: Operators, wildcards, and changing values | Smart Parking**
2. Combination of standardized queries using set operators
   - Basics: Construction of combined queries
   - Demo: Intersection of query result sets | Smart Emergency Response
   - Hands-on: Union of query result sets | Smart Running Track
3. User-defined queries and advanced query combination
   - Basics: Relations and relationship operators, AQL
   - Demo: AQL, transitive relations | Smart Traffic Planning
   - Hands-on: AQL, transitive relations | Smart Ambulance Navigation
4. Pre-defined analysis queries
   - Basics: Extending Mechanisms and Algorithms of Modeling Methods
   - Demo: Pre-defined queries | Smart Relay Marathon Runner
   - Hands-on: Pre-defined queries | Smart Relay Marathon Organization

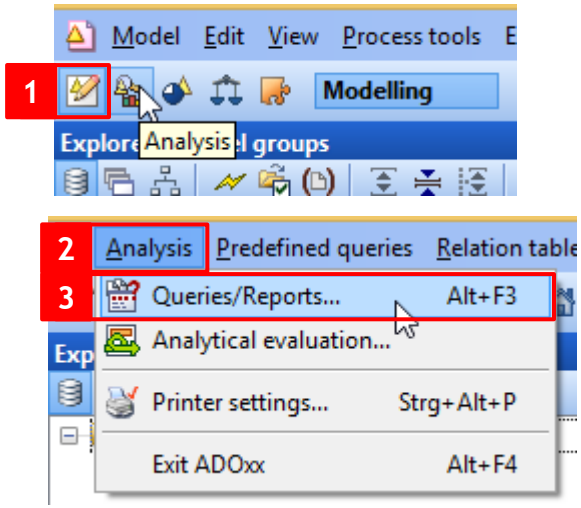
Execute a very first Query

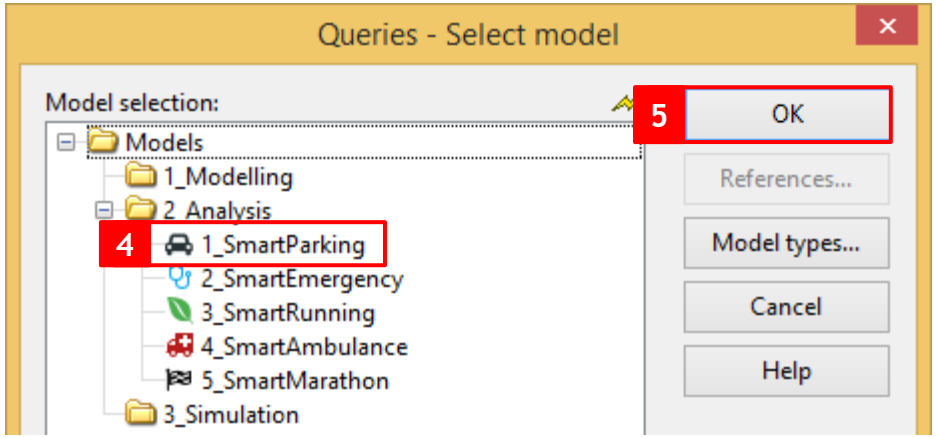
- The city of Vienna wants to change its smart traffic signs for the Eurovision Song Contest. You are in charge! The first step is to find the relevant signs via queries
  - Open the **Modelling Toolkit**
    - Log in
  - Open the **1_SmartParking** model (doubleclick) within the model group **2_Analysis**



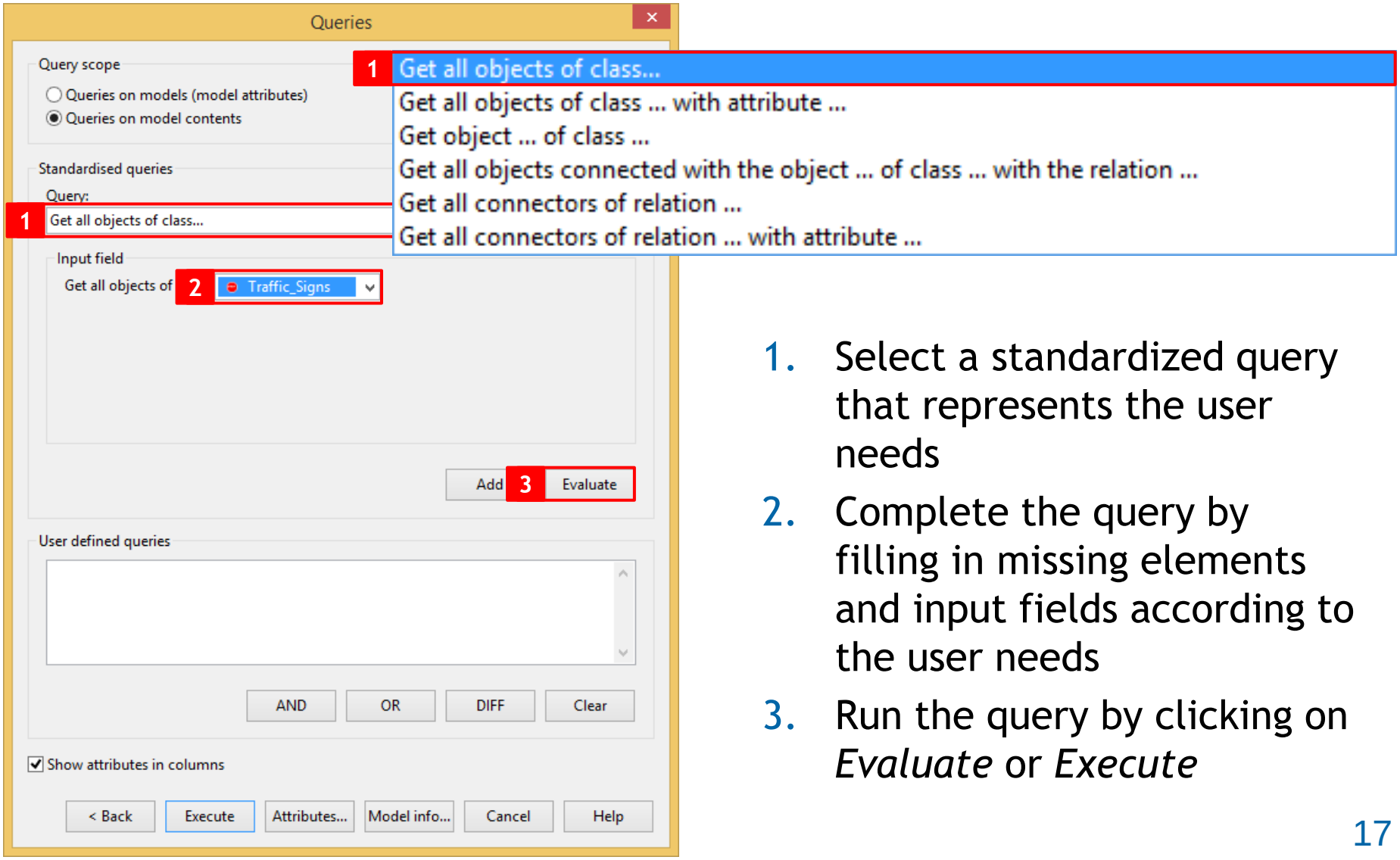
Basics: Construction of Standardized Queries (1/2)

- Queries are found in the *Analysis* component (in the toolbar)
  - **Standardized queries:**
    - > Standardized queries as "fill-in text", to be completed by the user.



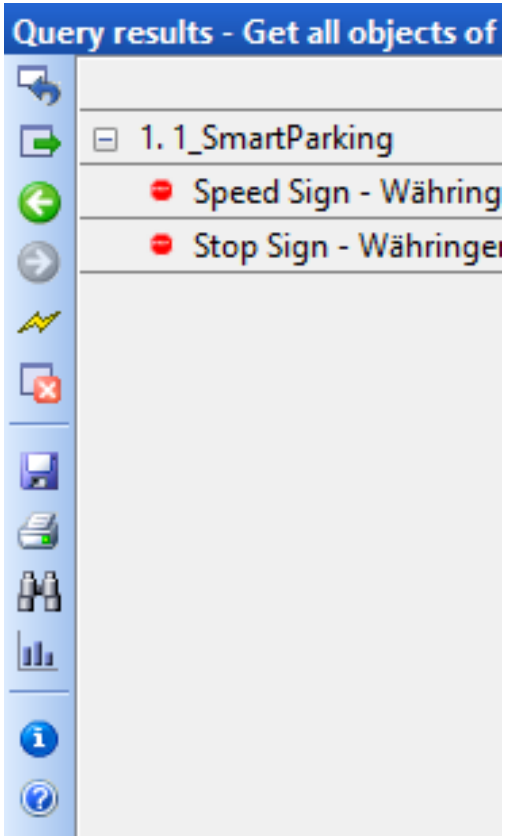


Basics: Construction of Standardized Queries (2/2)



1. Select a standardized query that represents the user needs
2. Complete the query by filling in missing elements and input fields according to the user needs
3. Run the query by clicking on *Evaluate* or *Execute*

Basics: Query Result Window (1/2)



- *Back to Query Definition*
- *Continuous Query*
- *Previous/Next Query Result*
- *Close Query Result*
- *Refresh*
- *Save/Print/Search*
- *Diagram*
- *Query Info*
- *Help*

Basics: Query Result Window (2/2)

- You can **click** the query result **to highlight the model object**
- You can **edit model attributes** in the *Query results* window
- You can **configure** the *Query results* window by right-clicking in the grid (*Select Attributes*)
  - Hint: Press *Ctrl* / *Strg* to select multiple attributes

Demo 1: Smart Parking Spot in Sensengasse (1/2)

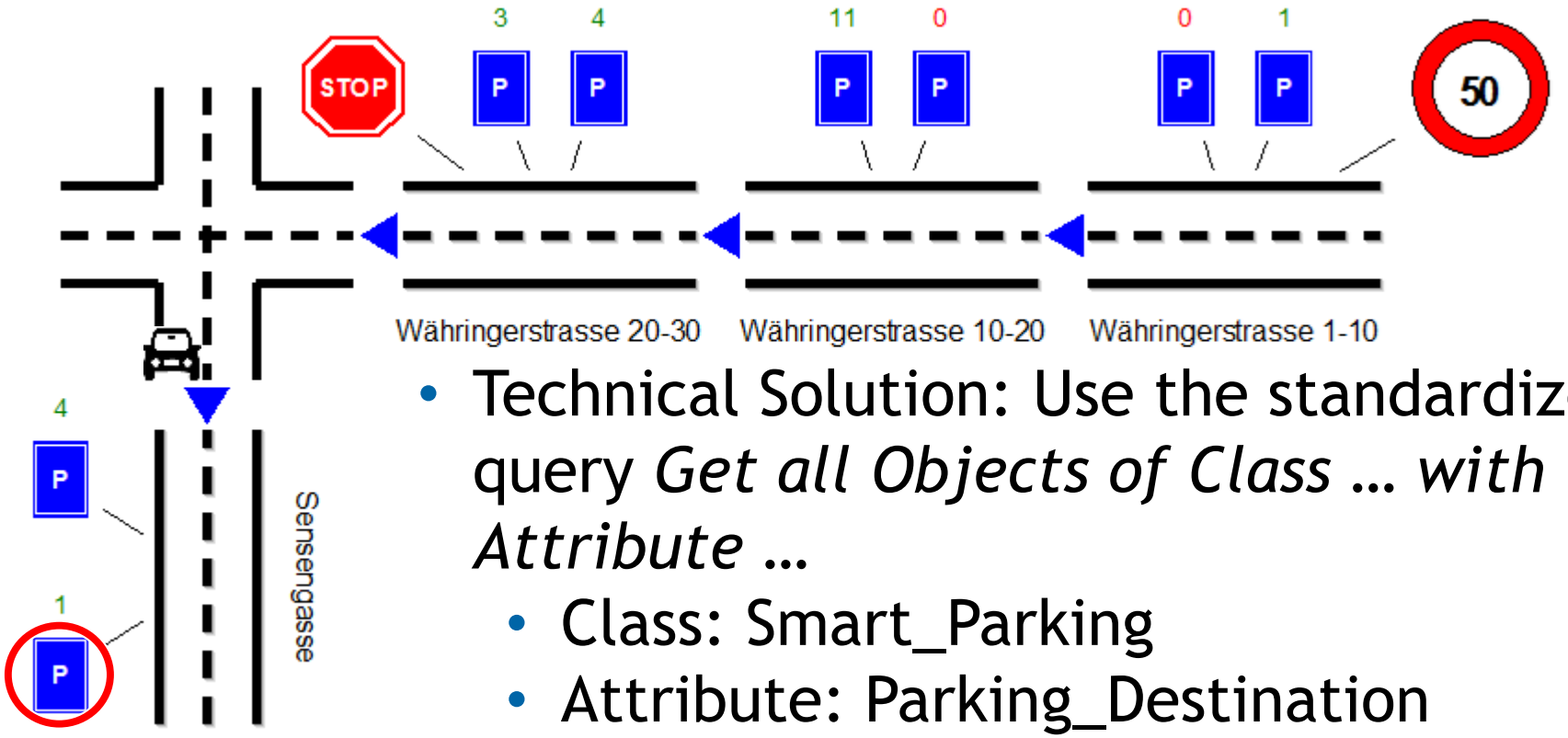
- *"Parking your car in big cities is becoming more and more difficult. Many drivers spend on average 20 minutes while looking for a parking spot. This increases CO2 emissions and most important of all; people waste their time."* [1]
- Envision an app that searches for smart and open parking spots, that can reserve a parking spot for you, and that can access the smart parking meter to automatically pay for tickets
- A Smart City model can provide real time data for a smart parking app

www.theparkerapp.com

[1] Amsterdam Smart City : http://amsterdamsmartcity.com/projects/detail/id/64/slug/smart-parking?lang=en

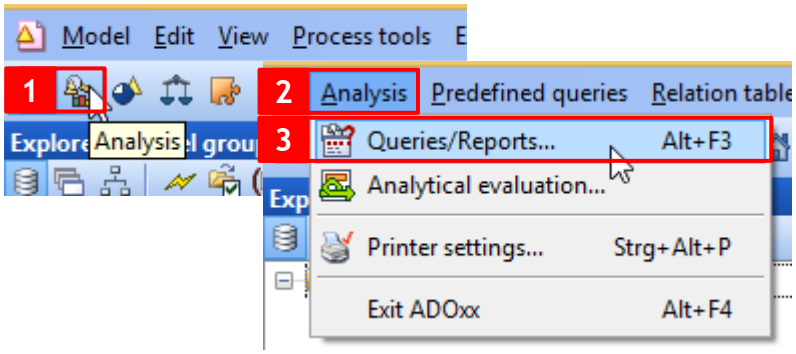
Demo 1: Smart Parking Spot in Sensengasse (2/2)

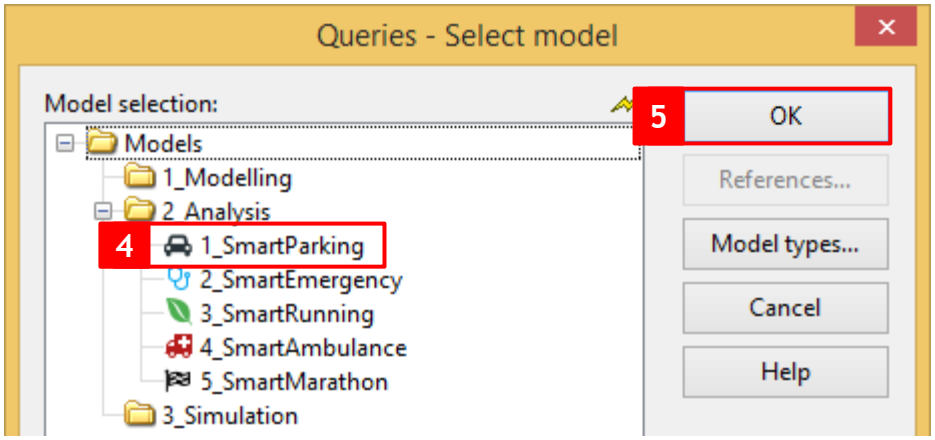
- Envision that you drive home from work and that you want to search and reserve a smart parking spot at your home address near your front entrance

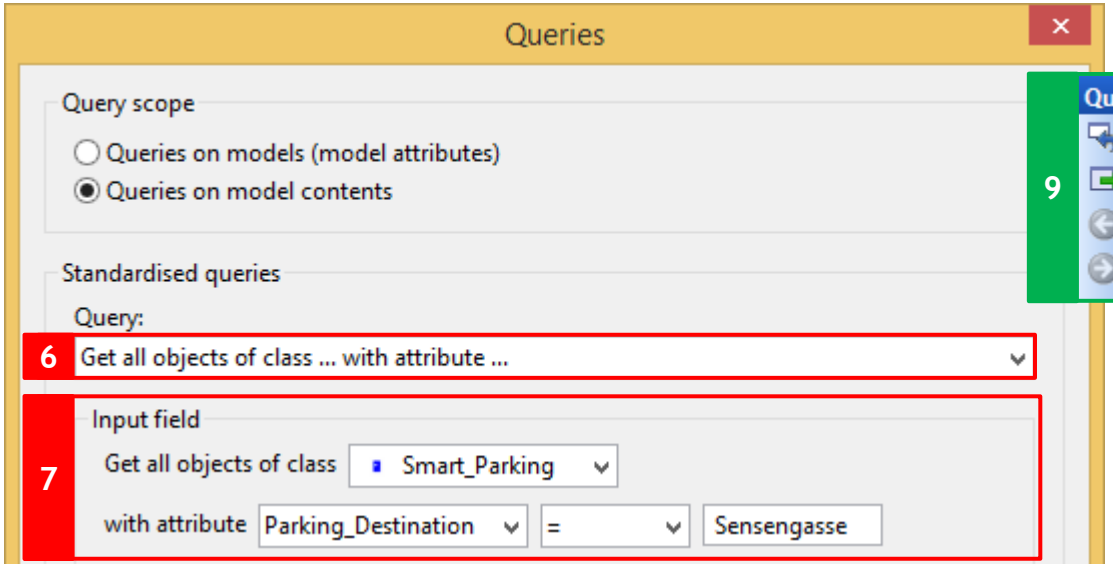


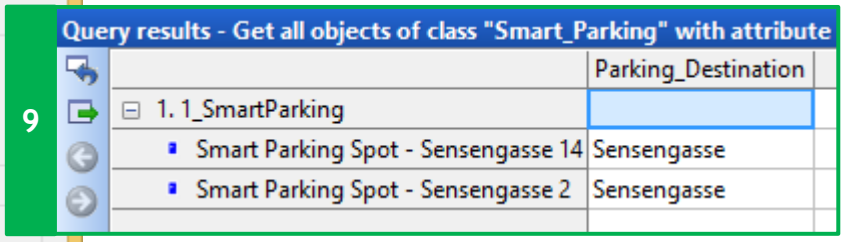
- Technical Solution: Use the standardized query *Get all Objects of Class … with Attribute …*
  - Class: Smart_Parking
  - Attribute: Parking_Destination

Demo 1: Solution



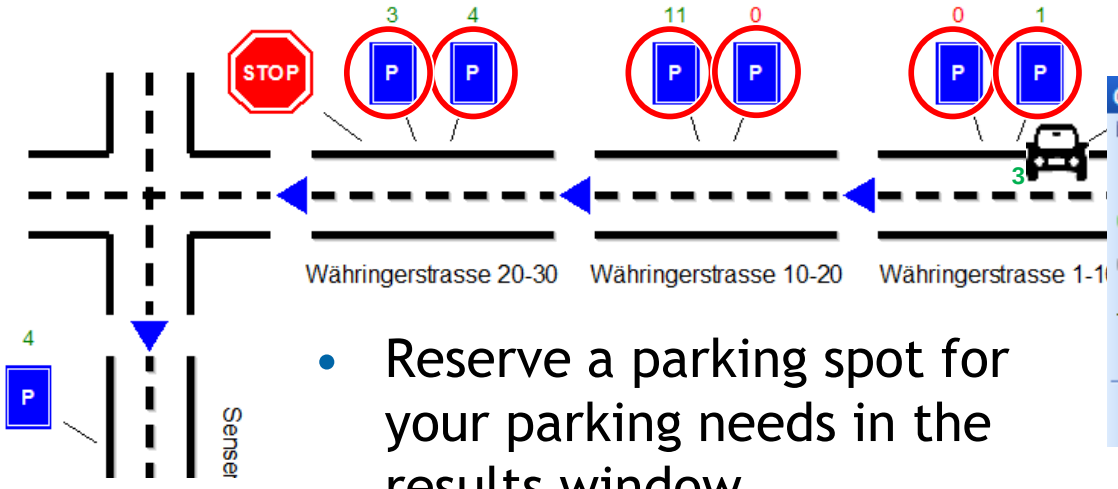


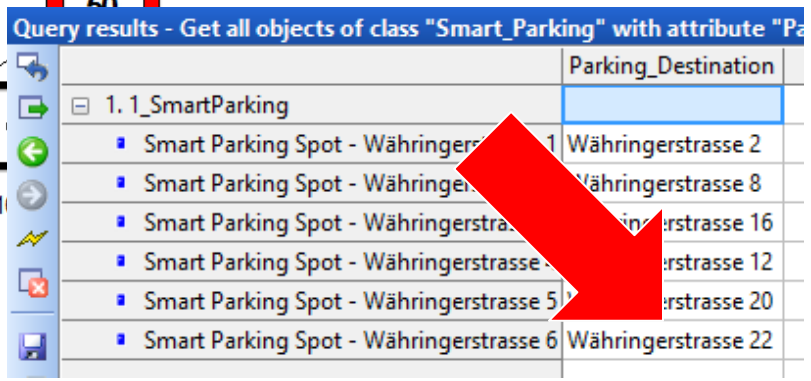




Hands-On 1: Reserve a Smart Parking Spot in Währingerstrasse

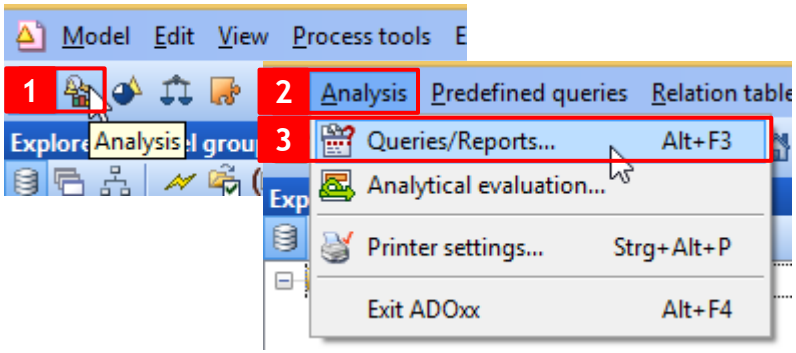
- You need a parking spot in *Währingerstrasse*. Find all parking spots in *Währingerstrasse* and display their attributes in the results window
  - Hint:
    - > Use the *like* operator to compare attributes and '?' or '*' as wildcard
    - > If you cannot find the 'ä' key you can use a wildcard

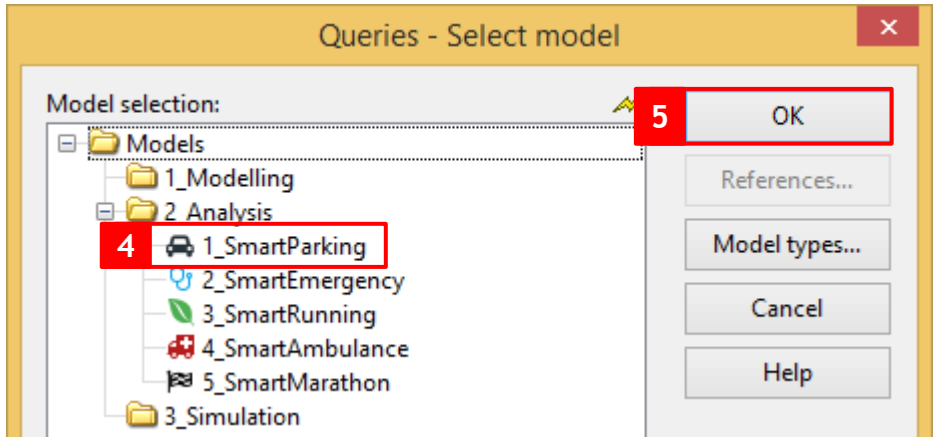


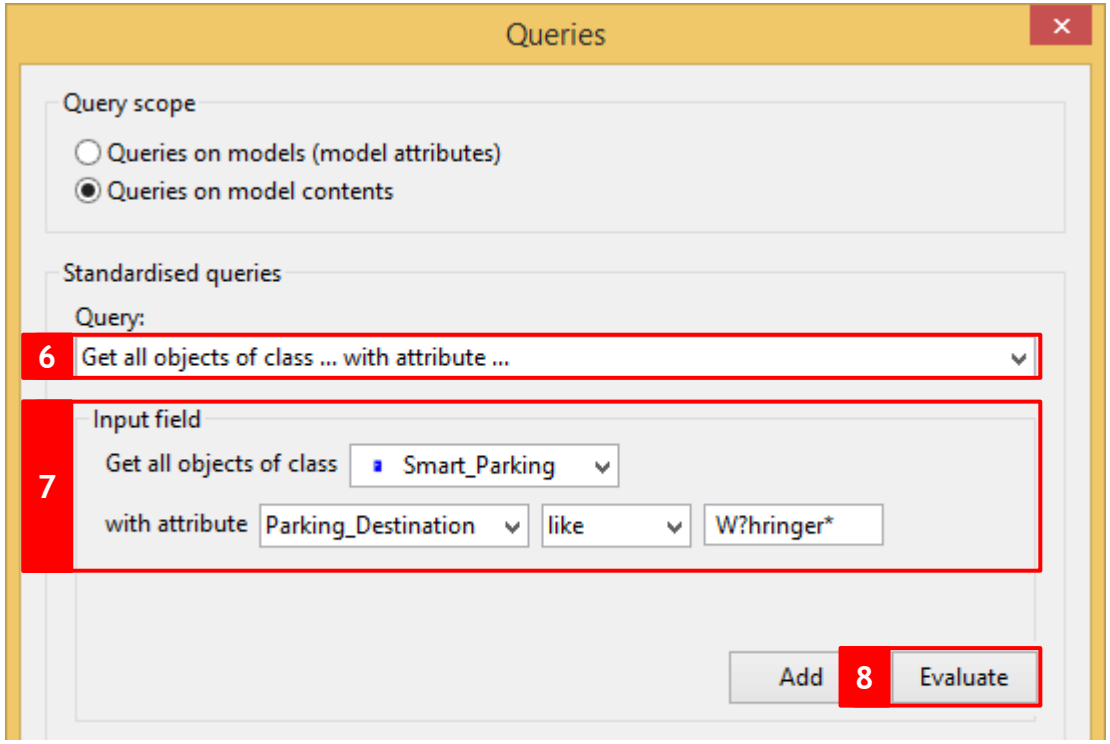


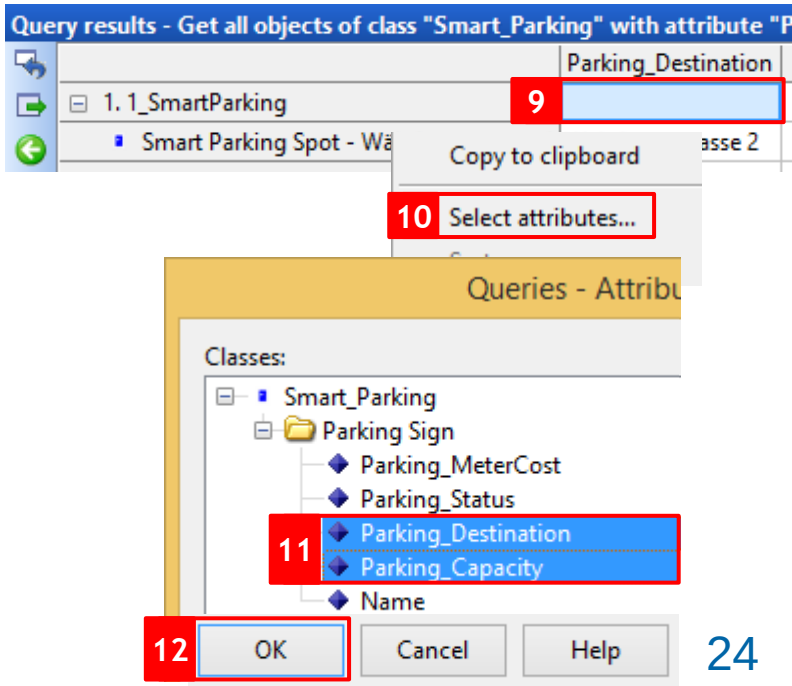
- Reserve a parking spot for your parking needs in the results window

Hands-On 1: Solution (1/2)





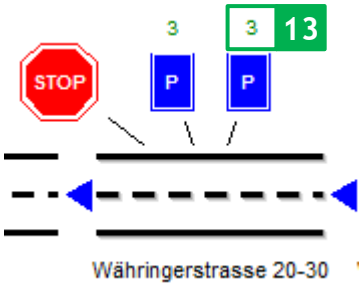




Hands-On 1: Solution (2/2)

Query results - Get all objects of class "Smart_Parking" with attribute "Parking_Destination" like "W?h*"

	Parking_Destination	Parking_Capacity
1. 1_SmartParking		
Smart Parking Spot - Währingerstrasse 1	Währingerstrasse 2	1
Smart Parking Spot - Währingerstrasse 2	Währingerstrasse 8	0
Smart Parking Spot - Währingerstrasse 3	Währingerstrasse 16	11
Smart Parking Spot - Währingerstrasse 4	Währingerstrasse 12	0
Smart Parking Spot - Währingerstrasse 5	Währingerstrass 13	3
Smart Parking Spot - Währingerstrasse 6	Währingerstrasse 22	3

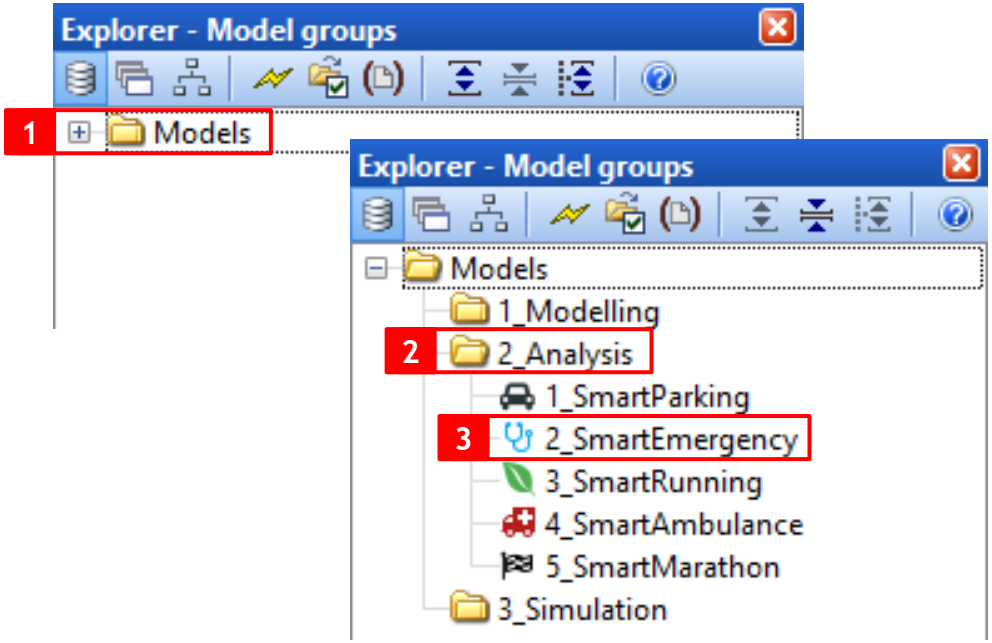


Exercise Structure

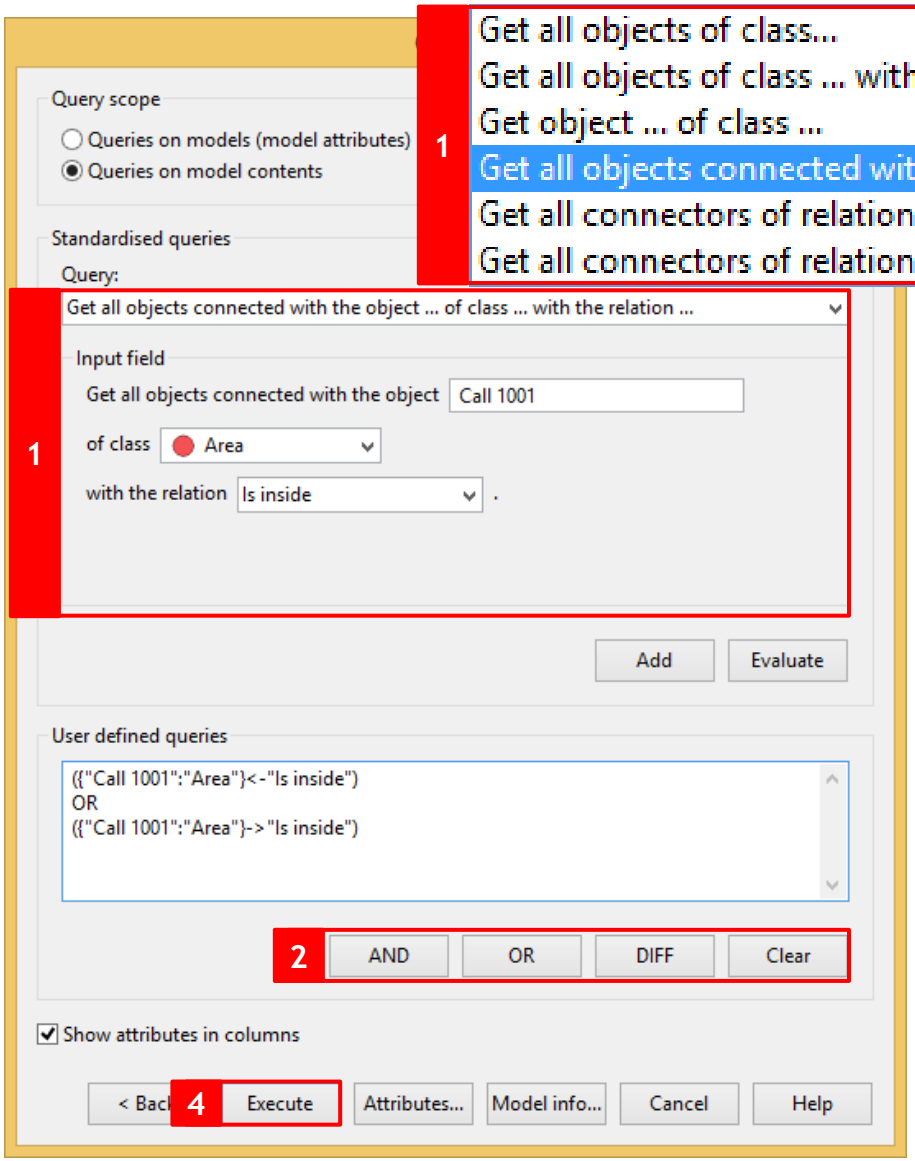
1. Standardized queries
   - Basics: The query dialog and the query result window | Smart Traffic Signs
   - Demo: Querying classes and attributes | Smart Parking
   - Hands-on: Operators, wildcards, and changing values | Smart Parking
2. **Combination of standardized queries using set operators**
   - **Basics: Construction of combined queries**
   - **Demo: Intersection of query result sets | Smart Emergency Response**
   - **Hands-on: Union of query result sets | Smart Running Track**
3. User-defined queries and advanced query combination
   - Basics: Relations and relationship operators, AQL
   - Demo: AQL, transitive relations | Smart Traffic Planning
   - Hands-on: AQL, transitive relations | Smart Ambulance Navigation
4. Pre-defined analysis queries
   - Basics: Extending Mechanisms and Algorithms of Modeling Methods
   - Demo: Pre-defined queries | Smart Relay Marathon Runner
   - Hands-on: Pre-defined queries | Smart Relay Marathon Organization

Combination of standardized queries using set operators

- Open the 2_SmartEmergency model within the model group 2_Analysis in the Modeling Toolkit.



Basics: Construction of Combined Queries

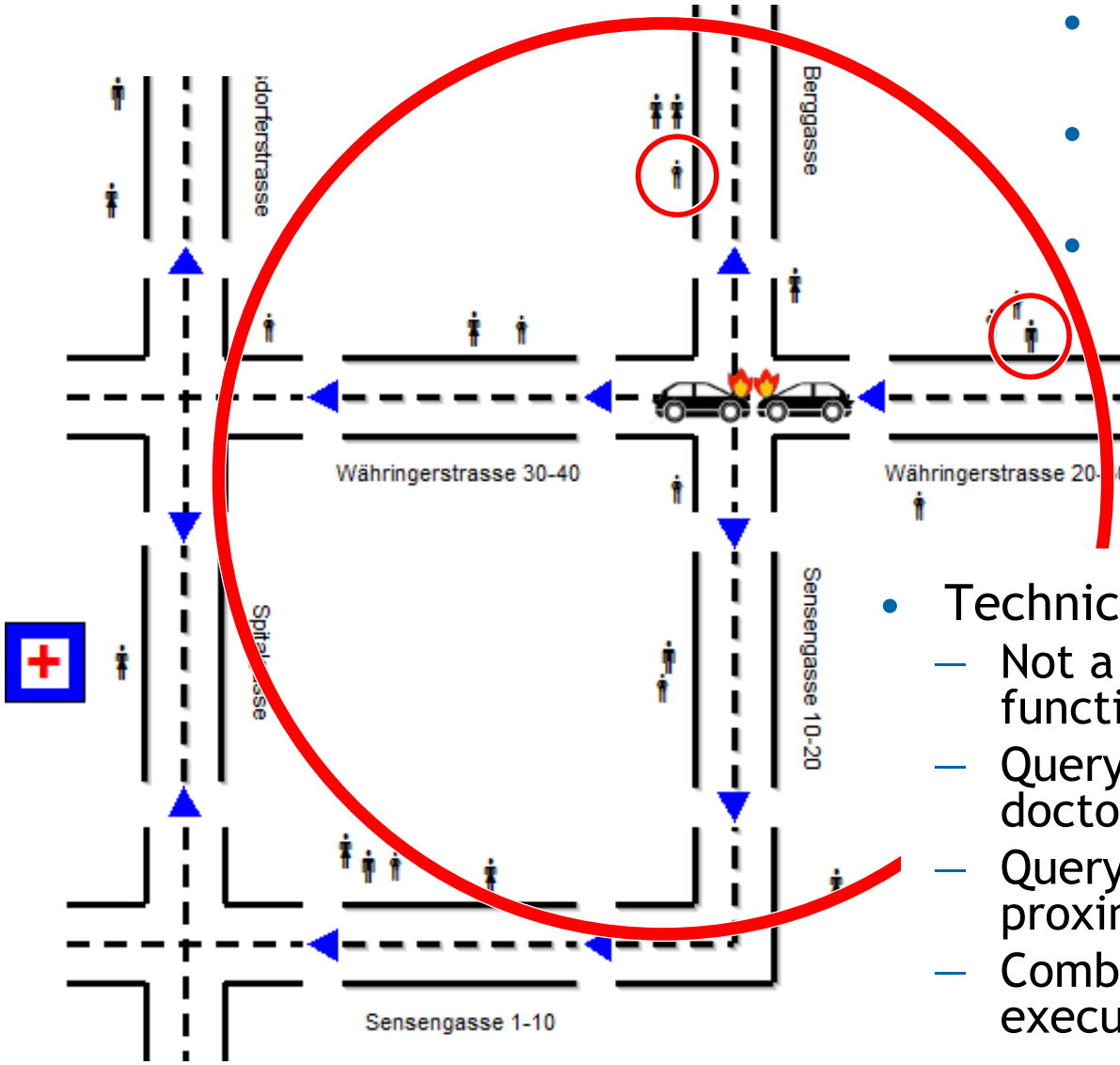


1. Select a standardized query that represents the user needs and complete the query by filling in missing elements and input fields
2. Standardized queries can be combined using the ***Add*** button and the logical operator such as ***AND***, ***OR***, ***DIFF***.
3. Repeat 1 and 2 as necessary
4. Run the query by clicking on *Execute*

Demo 2: There has been an Accident! (1/2)

- The emergency number 122 has been called by a concerned citizen. People have been injured
- The dispatcher informs emergency services immediately
- Additionally, the dispatcher is looking in the Smart City model for registered doctors in proximity of the accident
- The dispatcher informs relevant persons in the area via smartphone and dedicated apps

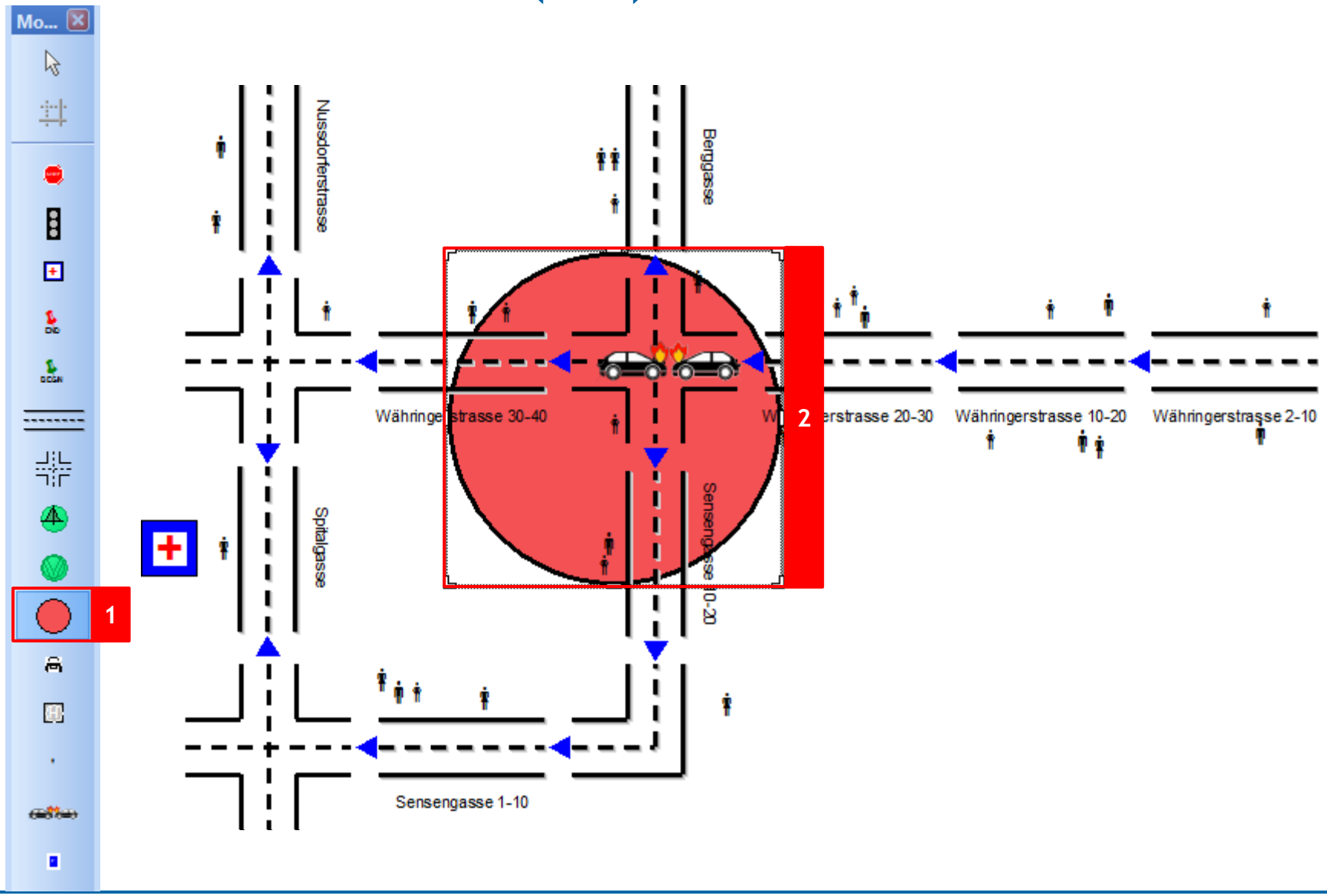
Demo 2: There has been an Accident! (2/2)



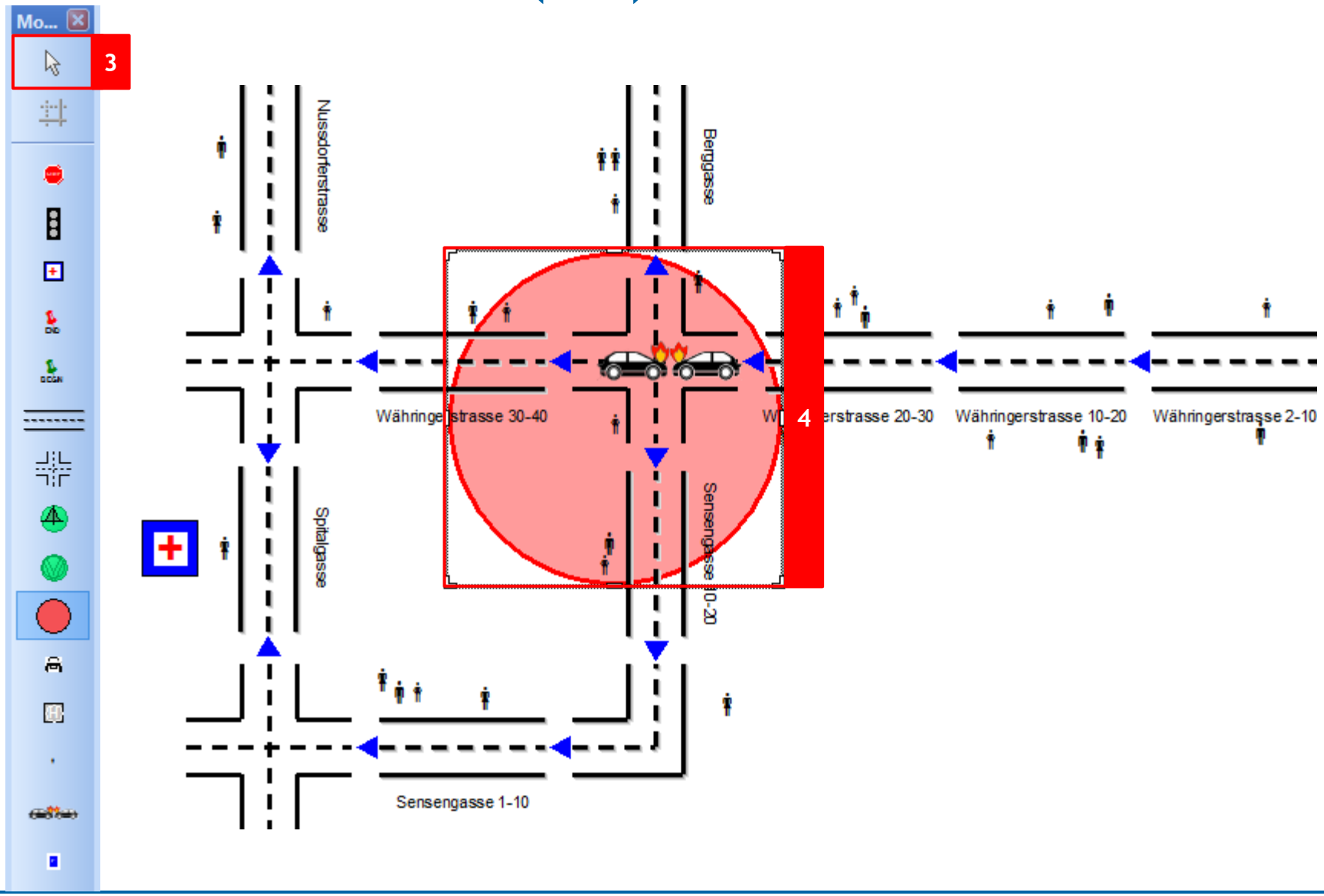
- A 122 call has been registered
- Search for doctors in proximity
- Inform them of the accident

- Technical solution:
  - Not a standardized query functionality
  - Query for all pedestrians that are doctors
  - Query for all pedestrians in close proximity
  - Combine the query result sets and execute the combined query

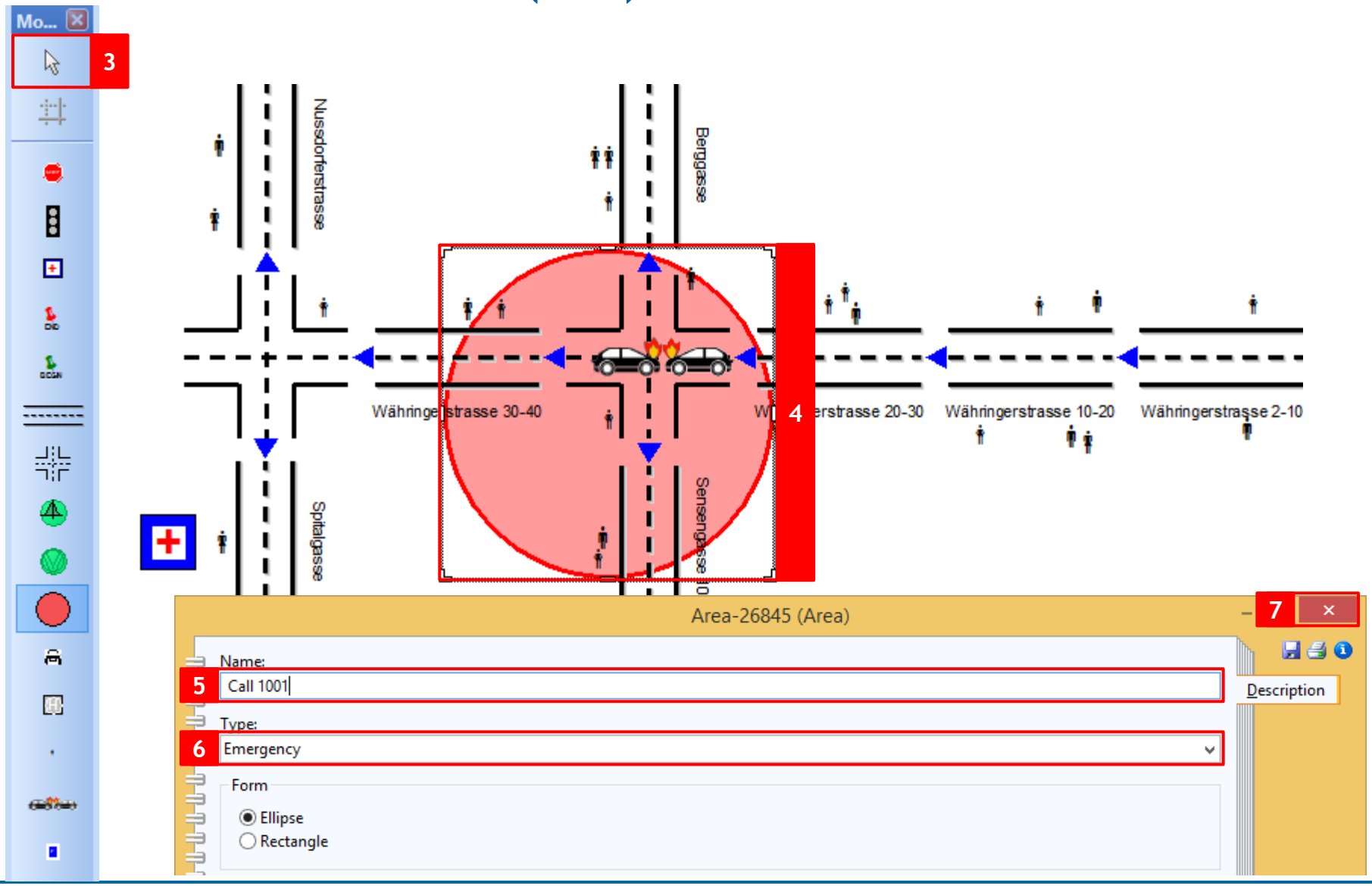
Demo 2: Solution (1/5)



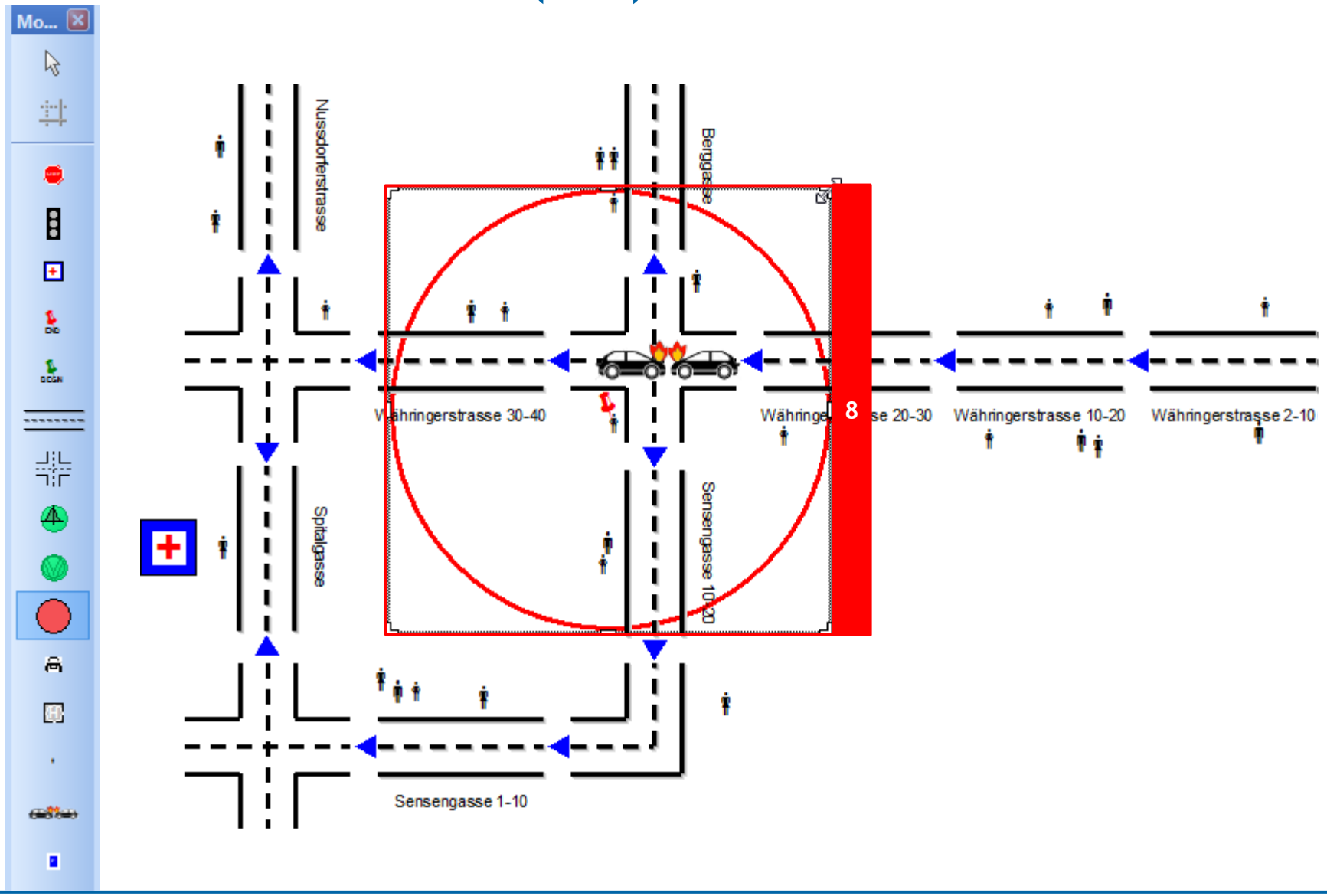
Demo 2: Solution (2/5)



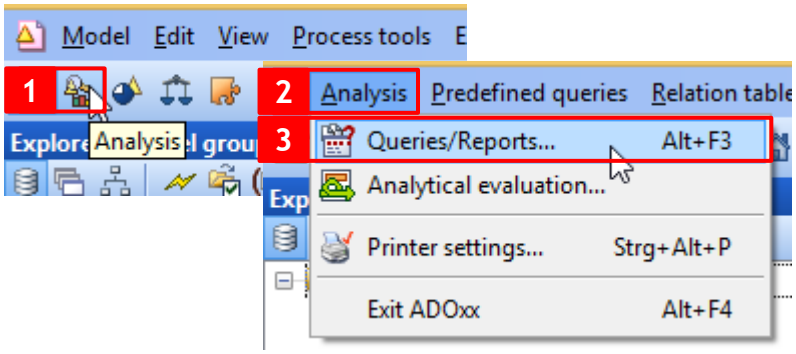
Demo 2: Solution (3/5)

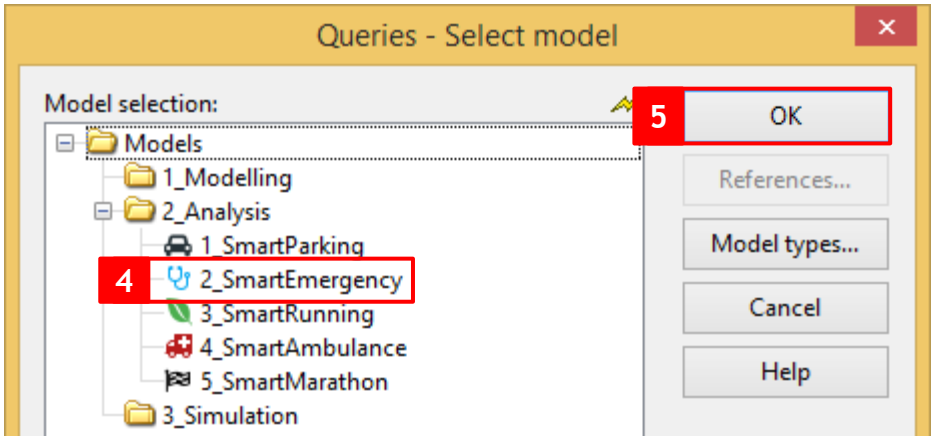


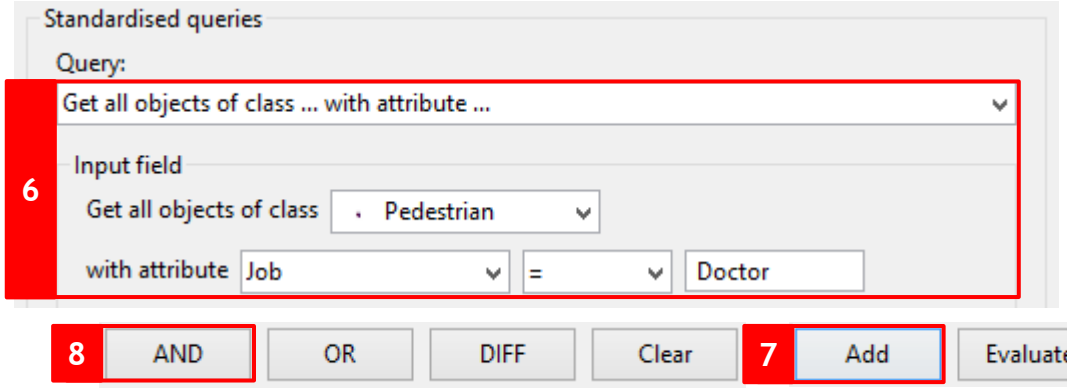
Demo 2: Solution (4/5)

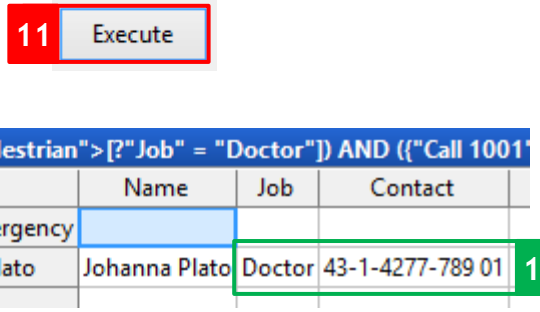


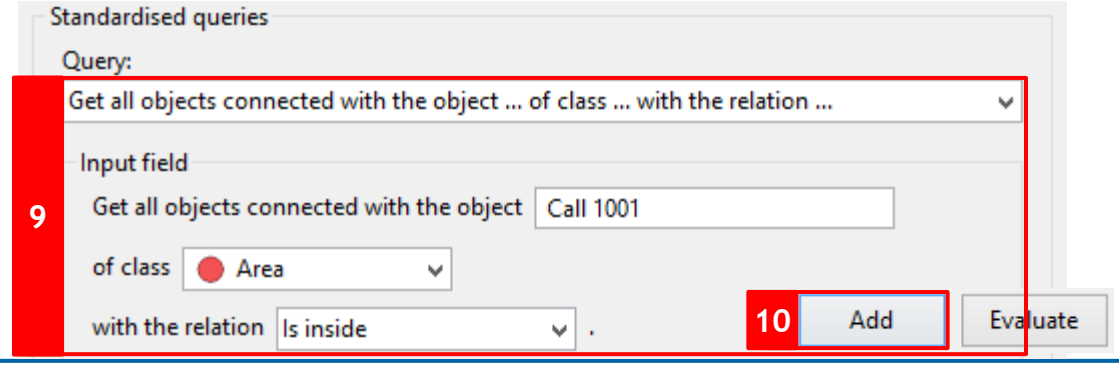
Demo 2: Solution (5/5)

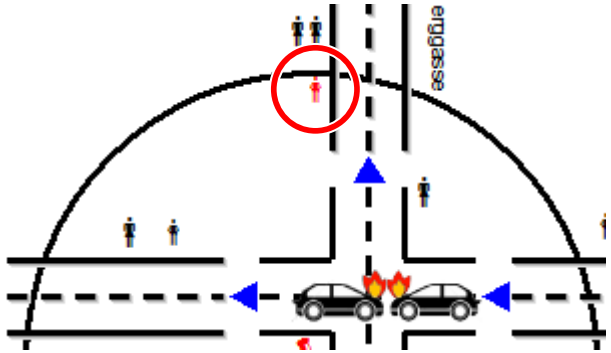












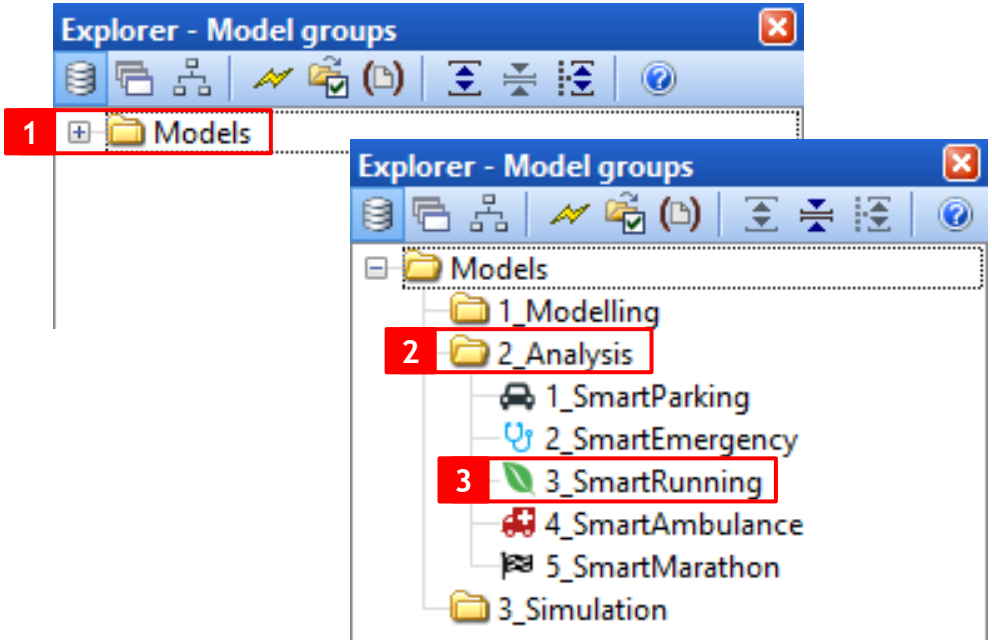
universität wien

OMILAB®
www.omilab.org

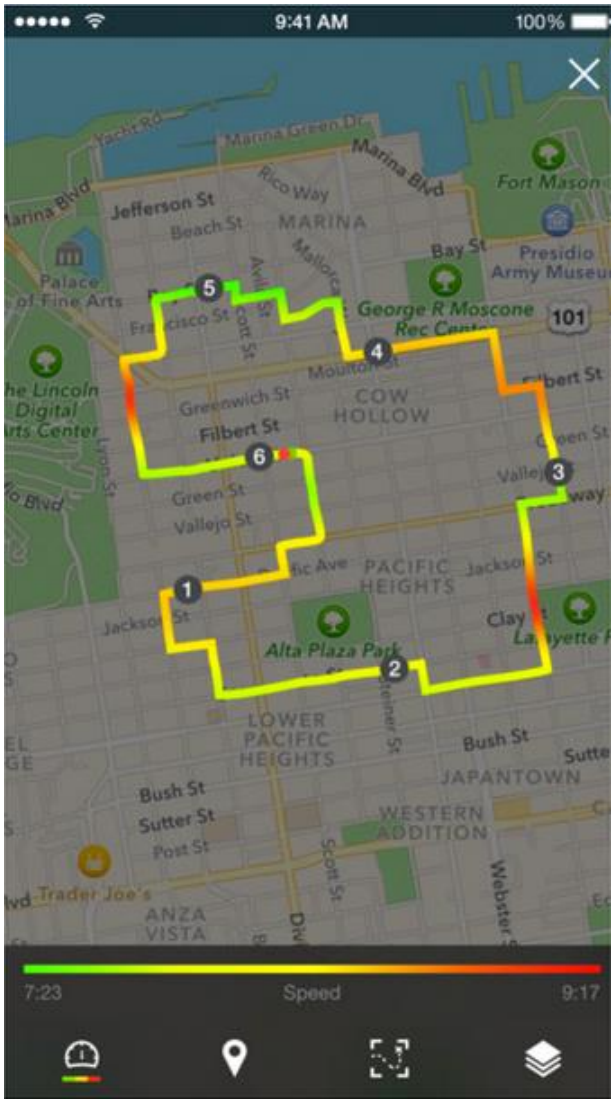
Erasmus+

Combination of standardized queries using set operators

- Open the 3_SmartRunning model within the model group 2_Analysis in the Modeling Toolkit.



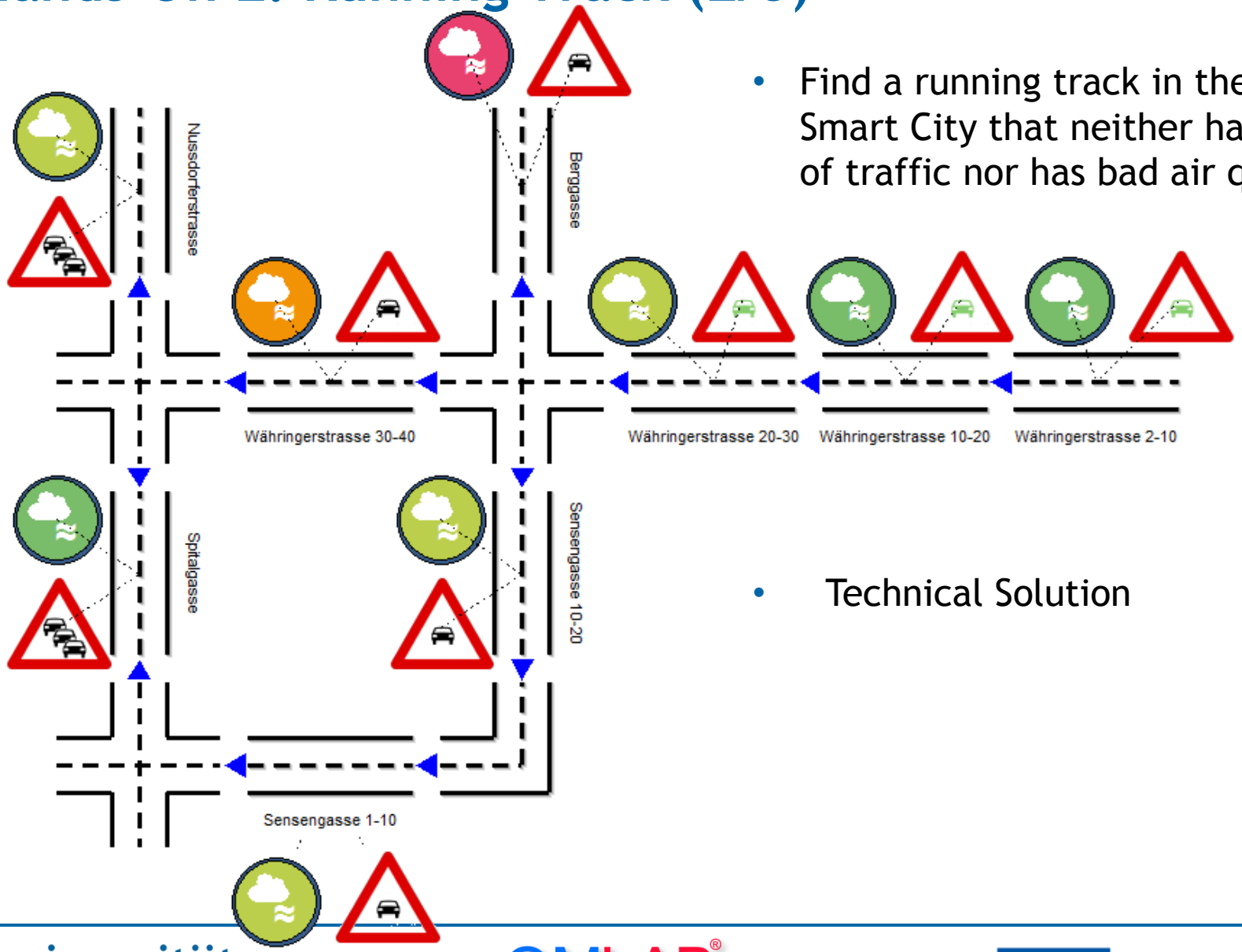
Hands-On 2: Smart Fitness Track (1/5)



www.runtastic.com

- Healthy citizens are cheap and happy citizens.
- Fitness, e.g., running, walking or cycling, is an essential part of a healthy lifestyle.
- To keep citizens save and healthy, fitness tracks should neither have bad air quality nor a lot of traffic.
- A Smart City model can provide the data to create a dynamic running track based on real time data and user requirements.

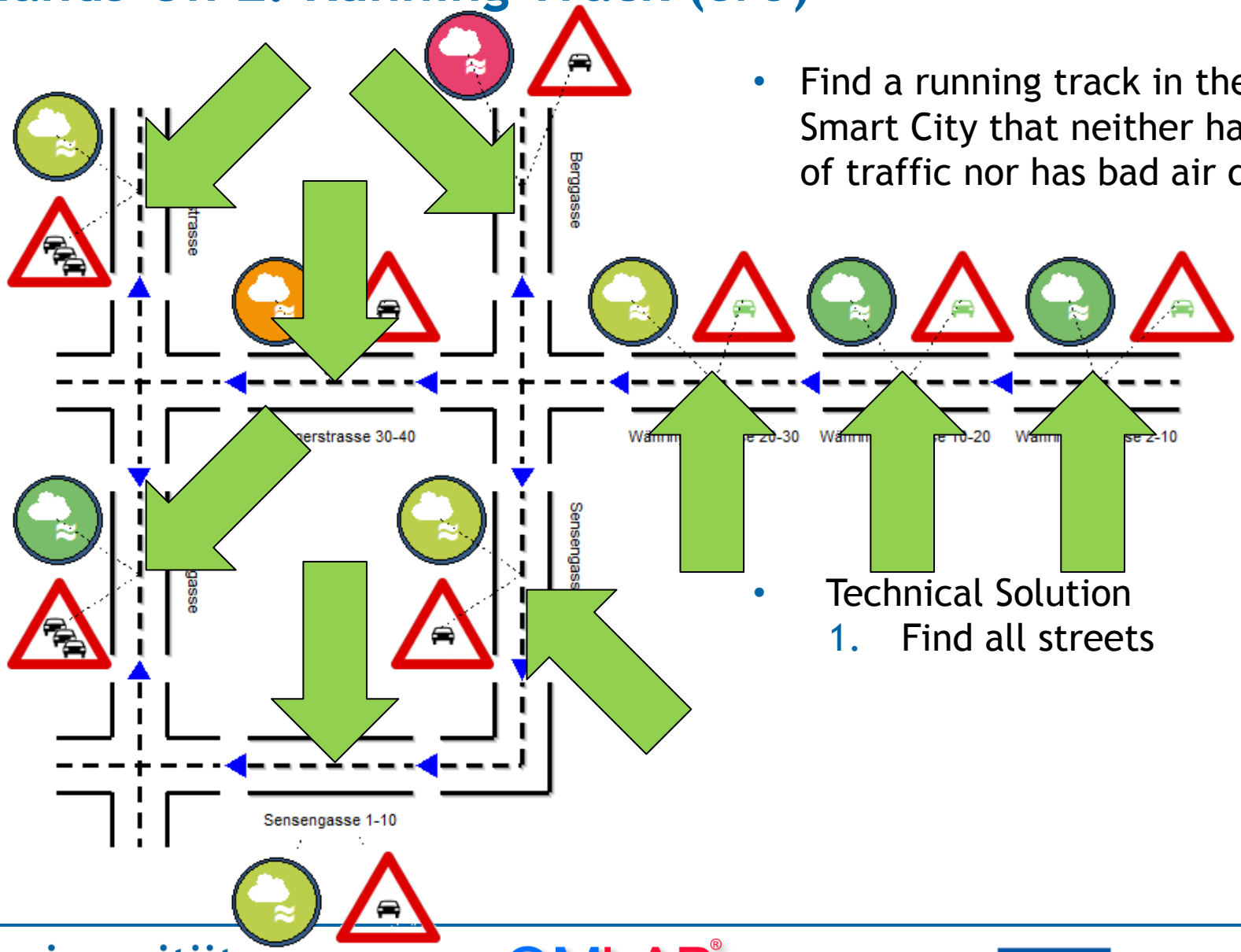
Hands-On 2: Running Track (2/5)



- Find a running track in the Smart City that neither has a lot of traffic nor has bad air quality

- Technical Solution

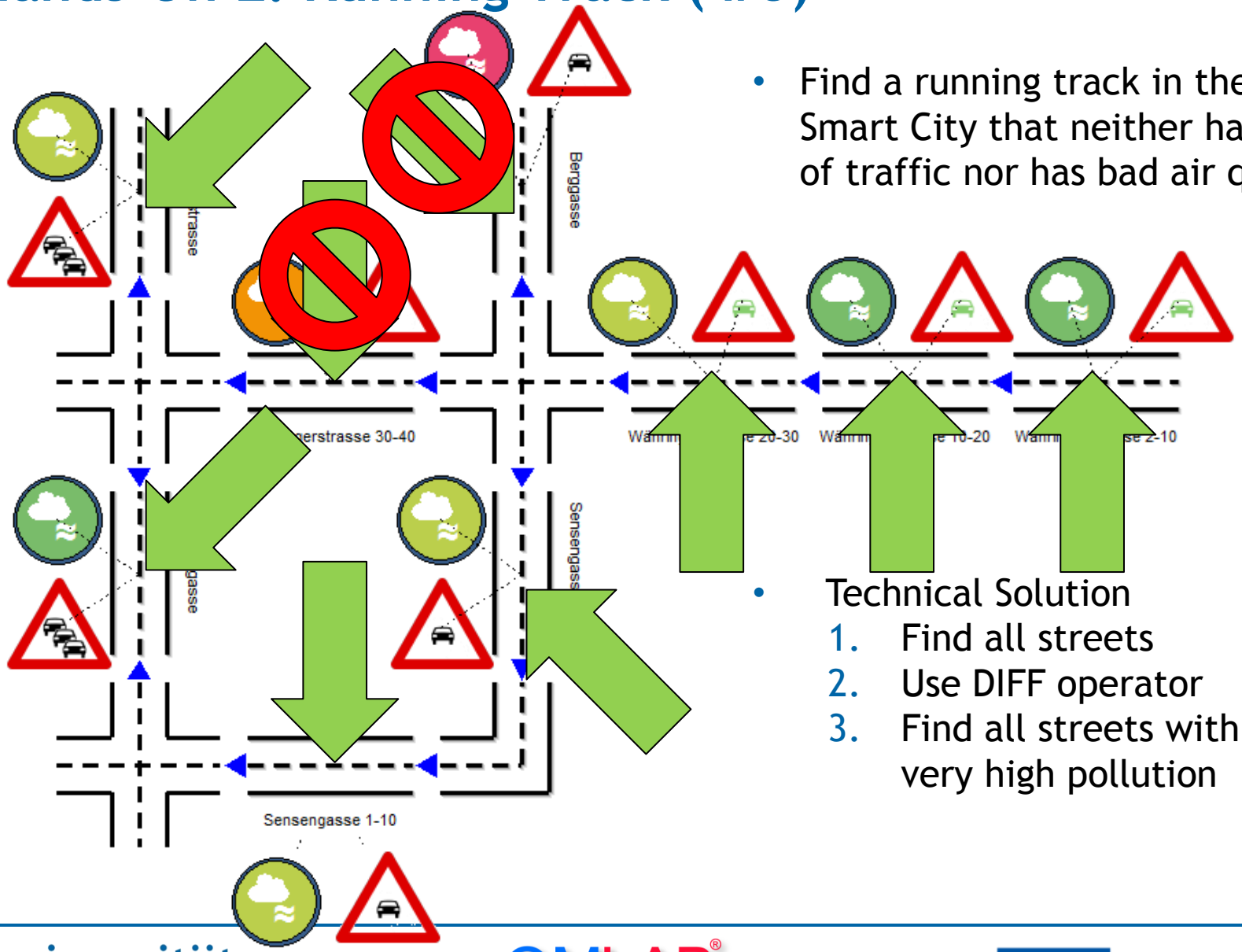
Hands-On 2: Running Track (3/5)



- Find a running track in the Smart City that neither has a lot of traffic nor has bad air quality

- Technical Solution
  1. Find all streets

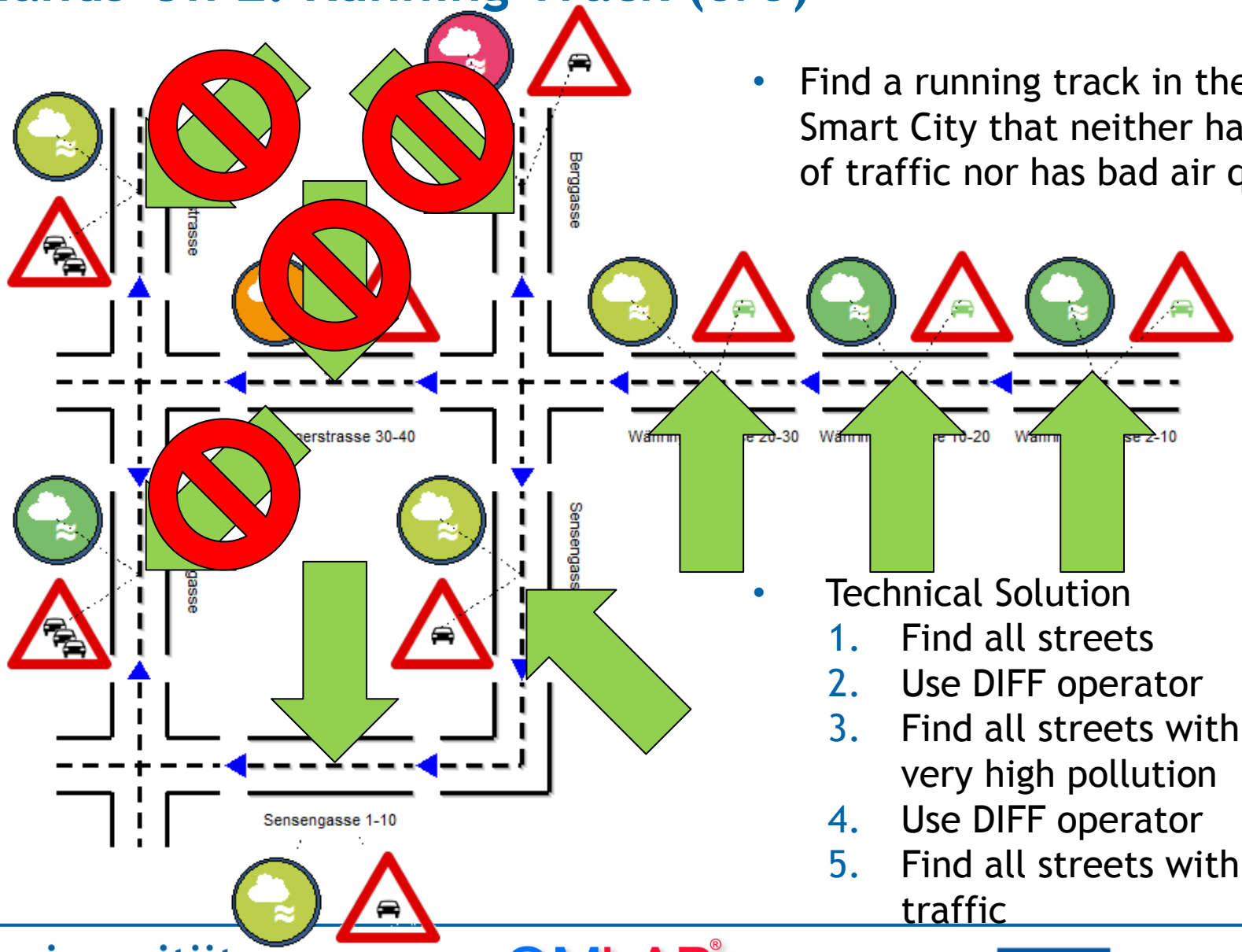
Hands-On 2: Running Track (4/5)



- Find a running track in the Smart City that neither has a lot of traffic nor has bad air quality

- Technical Solution
  1. Find all streets
  2. Use DIFF operator
  3. Find all streets with high or very high pollution

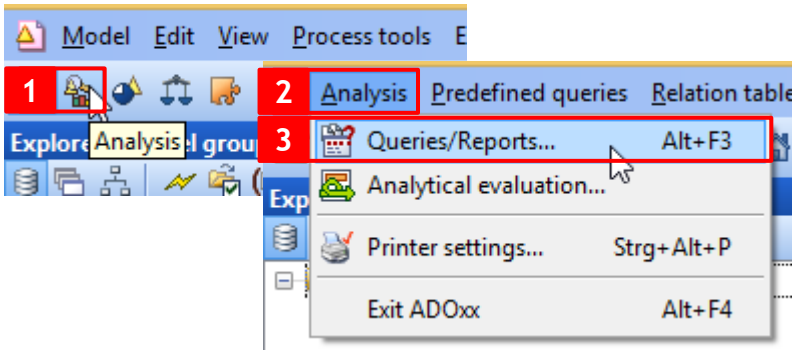
Hands-On 2: Running Track (5/5)

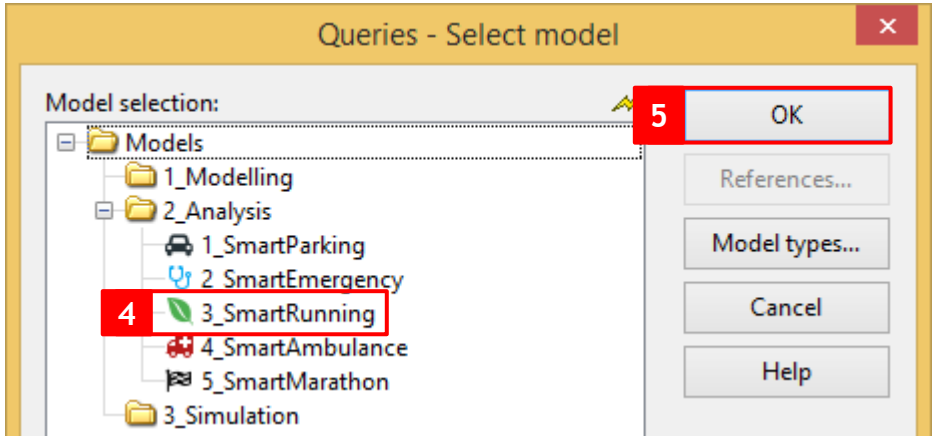


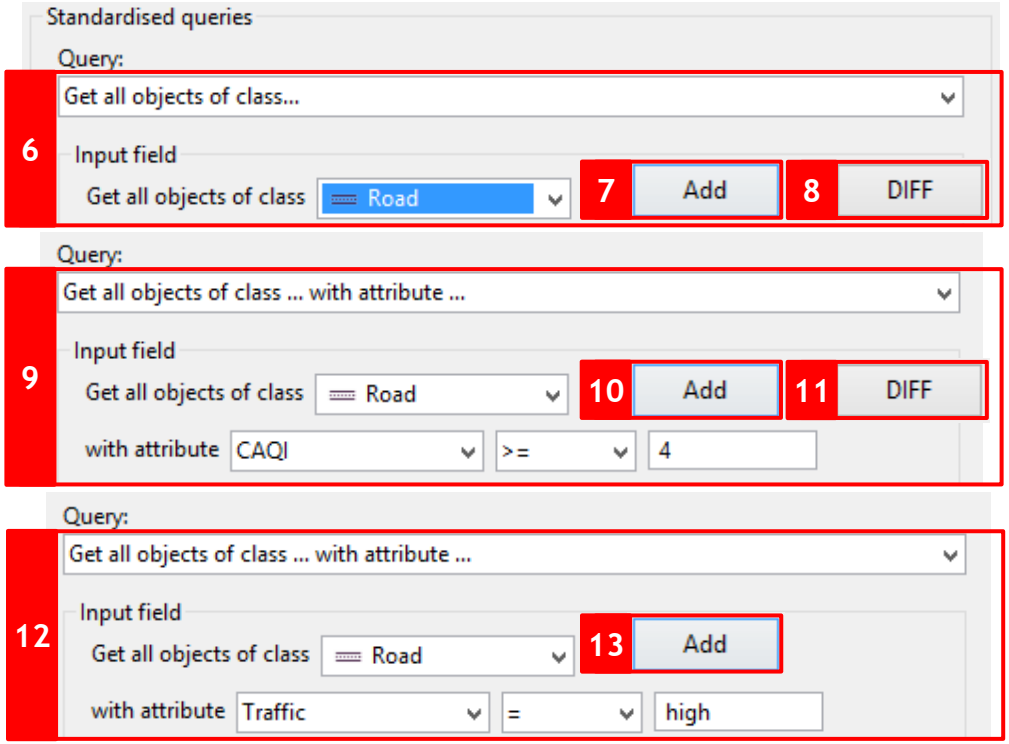
- Find a running track in the Smart City that neither has a lot of traffic nor has bad air quality

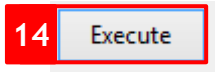
- Technical Solution
  1. Find all streets
  2. Use DIFF operator
  3. Find all streets with high or very high pollution
  4. Use DIFF operator
  5. Find all streets with high traffic

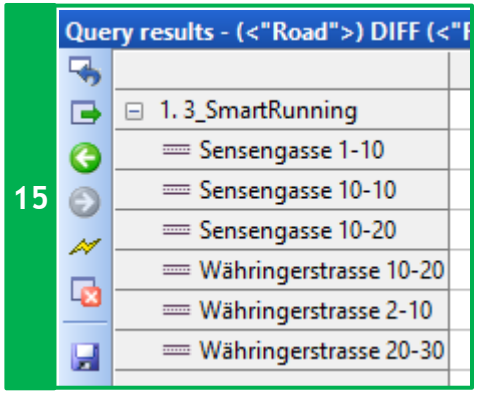
Hands-On 2: Solution









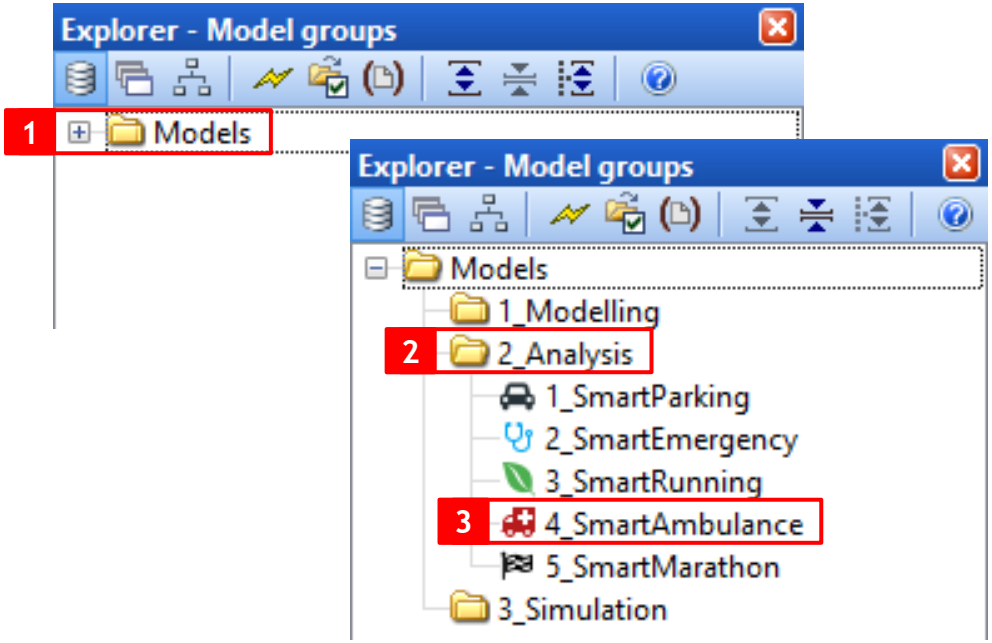


Exercise Structure

1. Standardized queries
   - Basics: The query dialog and the query result window | Smart Traffic Signs
   - Demo: Querying classes and attributes | Smart Parking
   - Hands-on: Operators, wildcards, and changing values | Smart Parking
2. Combination of standardized queries using set operators
   - Basics: Construction of combined queries
   - Demo: Intersection of query result sets | Smart Emergency Response
   - Hands-on: Union of query result sets | Smart Running Track
3. **User-defined queries and advanced query combination**
   - **Basics: Relations and relationship operators, AQL**
   - **Demo: AQL, transitive relations | Smart Traffic Planning**
   - **Hands-on: AQL, transitive relations | Smart Ambulance Navigation**
4. Pre-defined analysis queries
   - Basics: Extending Mechanisms and Algorithms of Modeling Methods
   - Demo: Pre-defined queries | Smart Relay Marathon Runner
   - Hands-on: Pre-defined queries | Smart Relay Marathon Organization

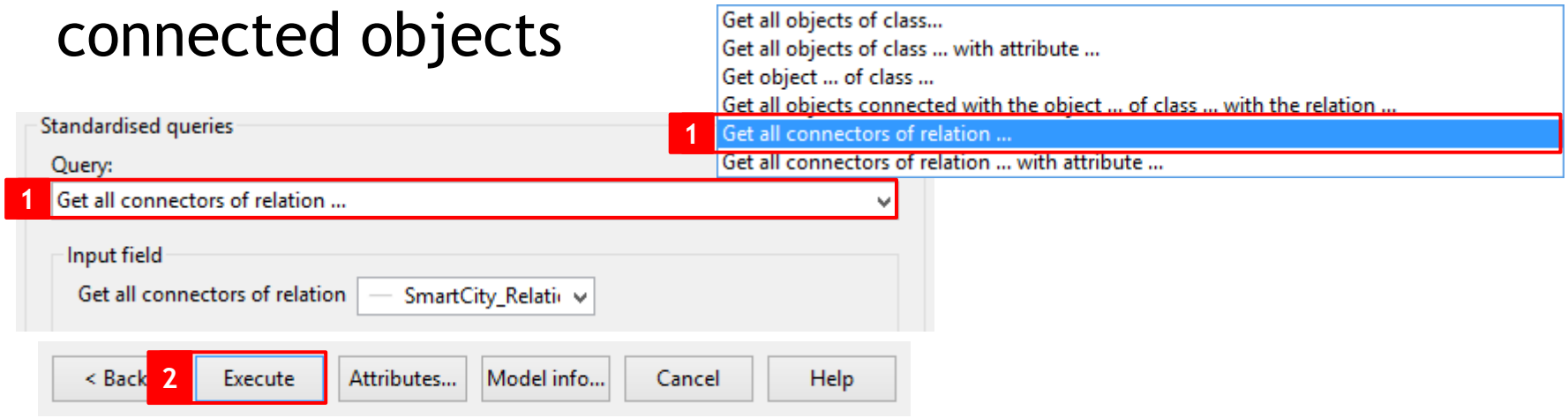
User defined queries and advanced query combination

- Open the **SmartAmbulance** model within the model group **2_Analysis** in the **Modeling Toolkit**.

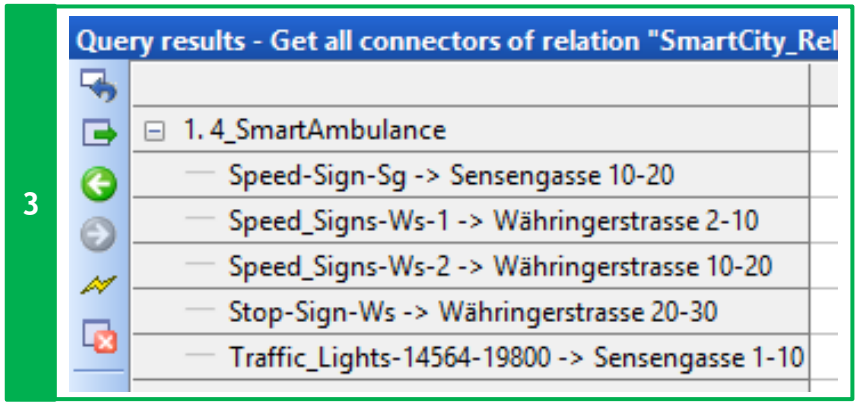


Basics: Relations

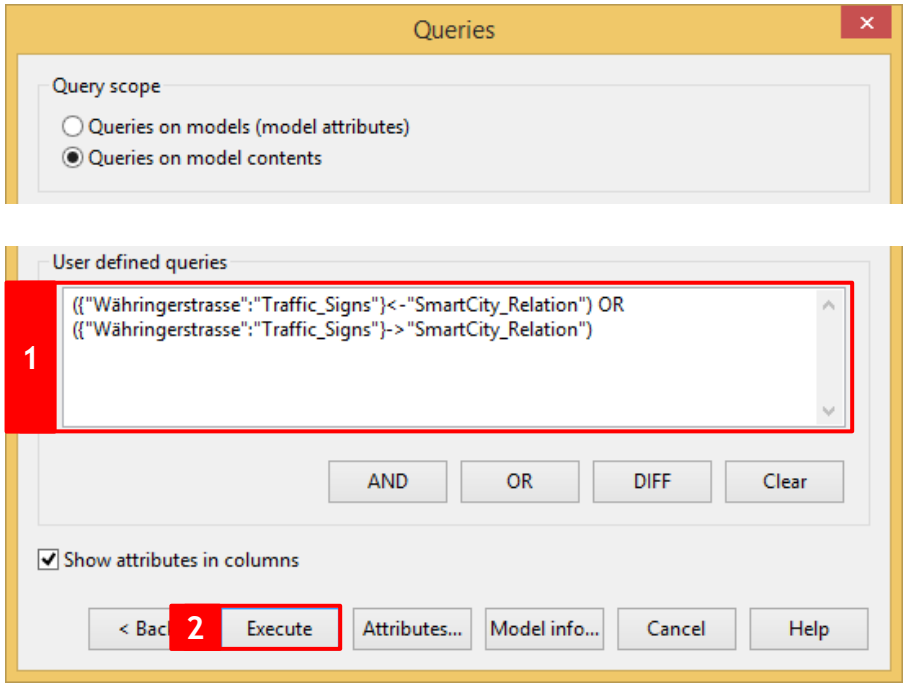
- A Query for relations returns connectors and connected objects



- The result contains all connectors of the specified relation and their start and target object



Basics: User-defined Queries & AQL

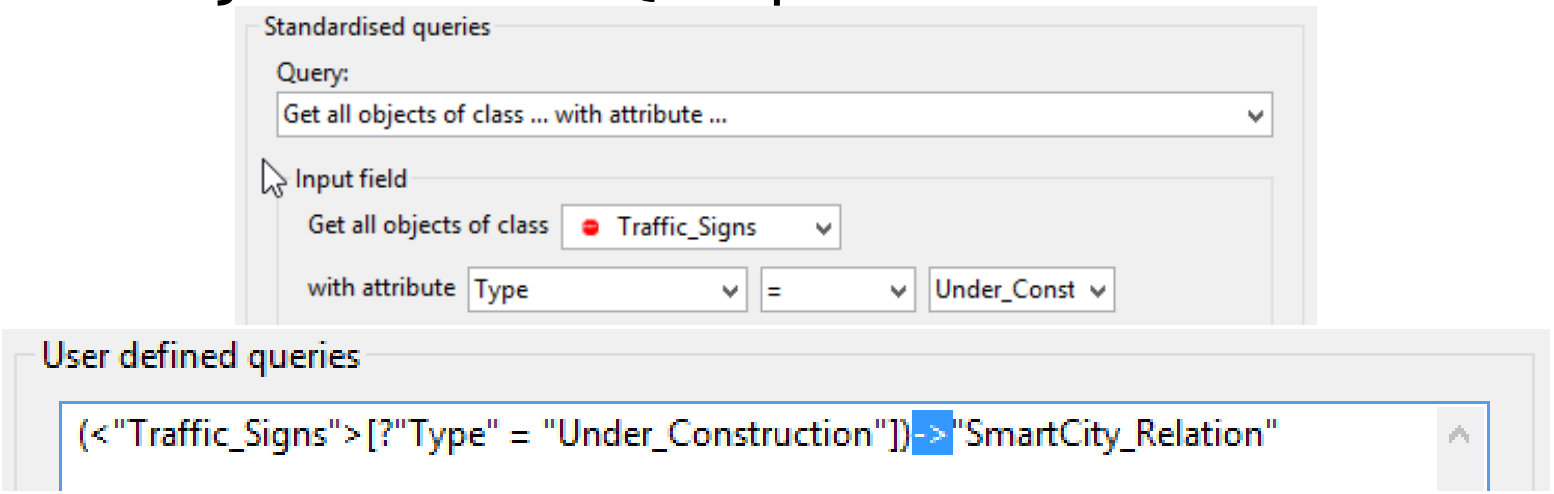


- User-defined queries:
  - Queries which are defined by the user through standardized queries in AQL syntax. For execution AQL knowledge is required.

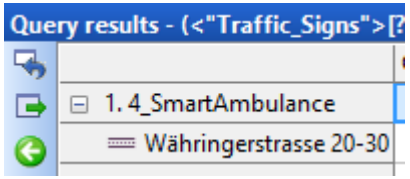
- Queries, can be completely defined by the user by using AQL

Basics: Combining Query Result Sets with the Relation Set using Links <- | ->

- '**->**' returns all **Connectors / connected Objects** outgoing from the objects of the AQL expression
- '**<-**' returns all **Connectors / connected Objects** incoming in the objects of the AQL expression



- The result contains all objects connected with construction sites

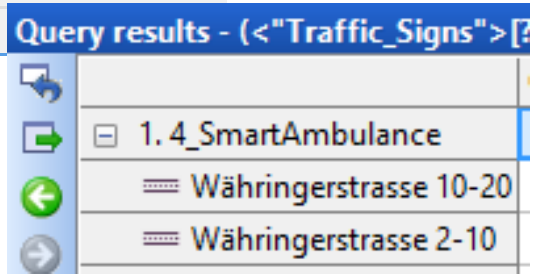


Basics: Combining a Query Result Set with the Relation Set using Transitive Links <<- | ->>

- Cascading links and transitive reference
  - The '**->>**' and '**<<-**' operators return all **Connectors / connected Objects** which are transitively referenced

User defined queries

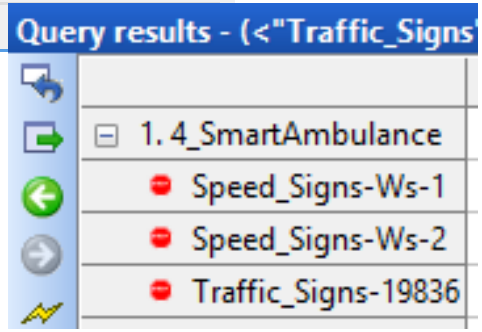
```
(<"Traffic_Signs">[?"Type" = "Under_Construction"])->"SmartCity_Relation"
<<-"Subsequent"
```



- The result lists all streets preceding the construction site

User defined queries

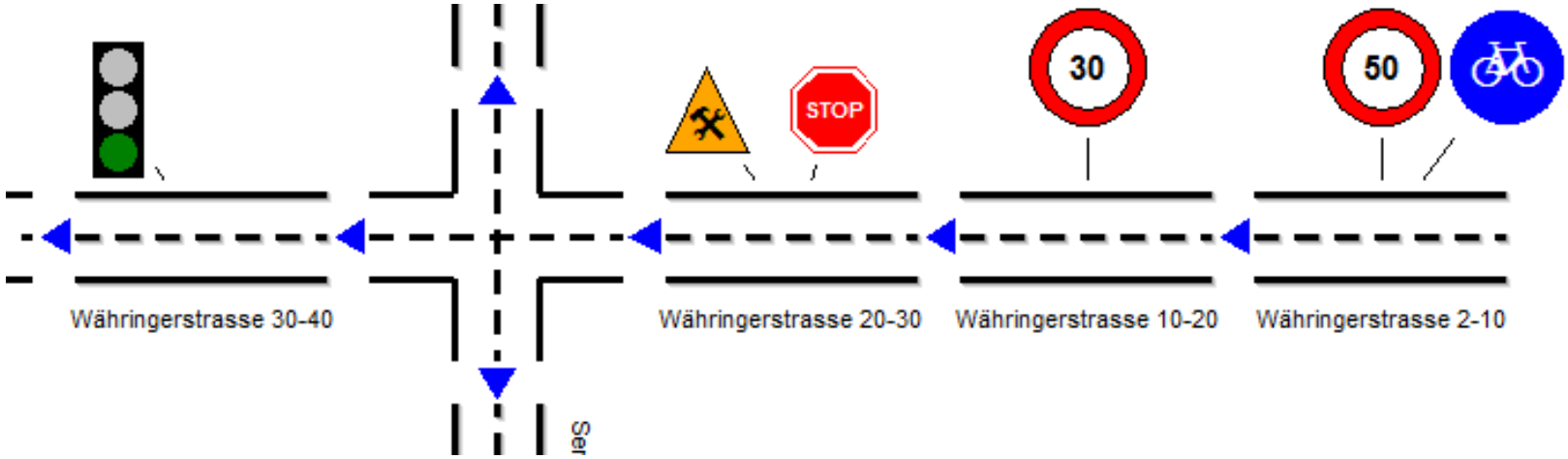
```
(<"Traffic_Signs">[?"Type" = "Under_Construction"])->"SmartCity_Relation"
<<-"Subsequent"
<-"SmartCity_Relation"
```



- The result lists all signs on preceding streets

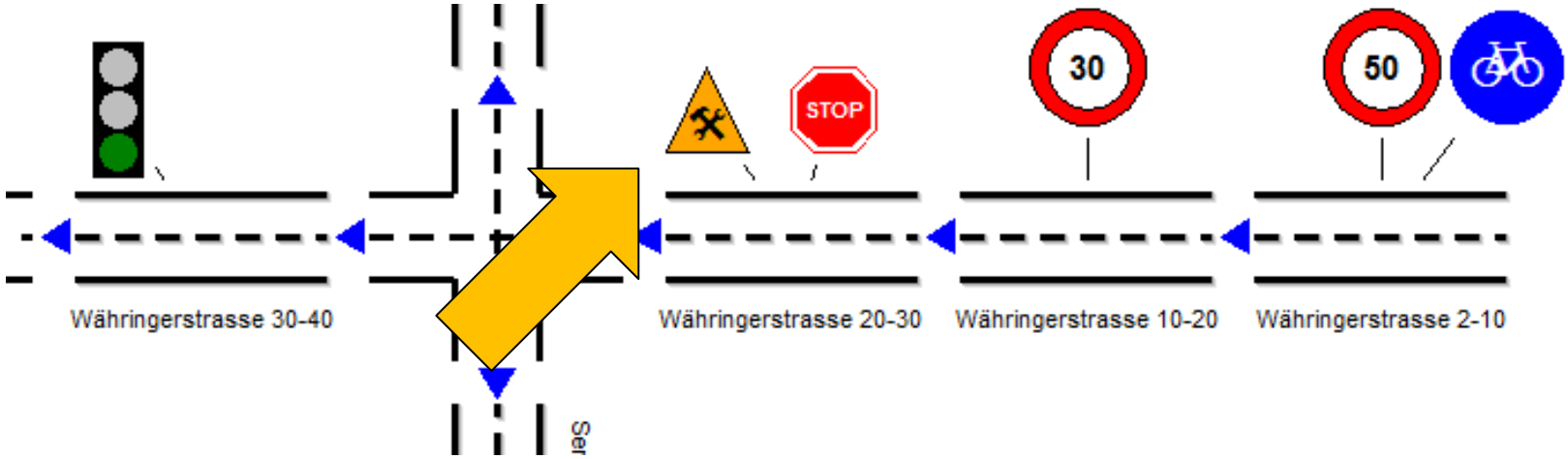
Demo 3: Construction Sites (1/4)

- The urban traffic planner needs to find and adjust all Smart City speed limit signs preceding a construction site
- Technical Solution:



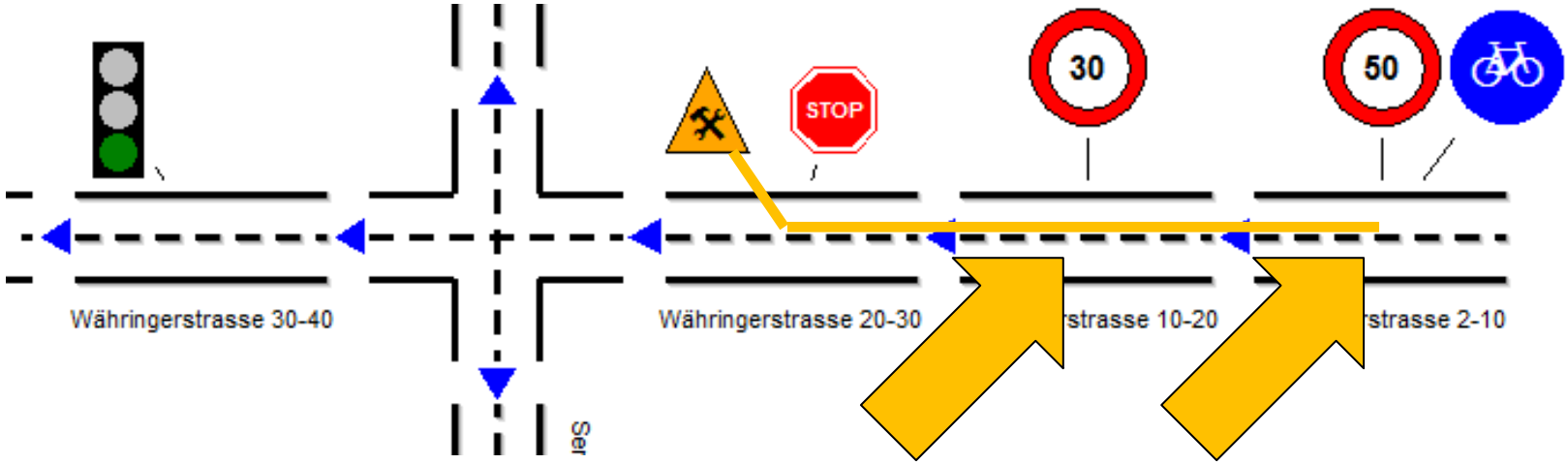
Demo 3: Construction Sites (2/4)

- The urban traffic planner needs to find and adjust all Smart City speed limit signs preceding a construction site
- Technical Solution:
  - Query for construction sites



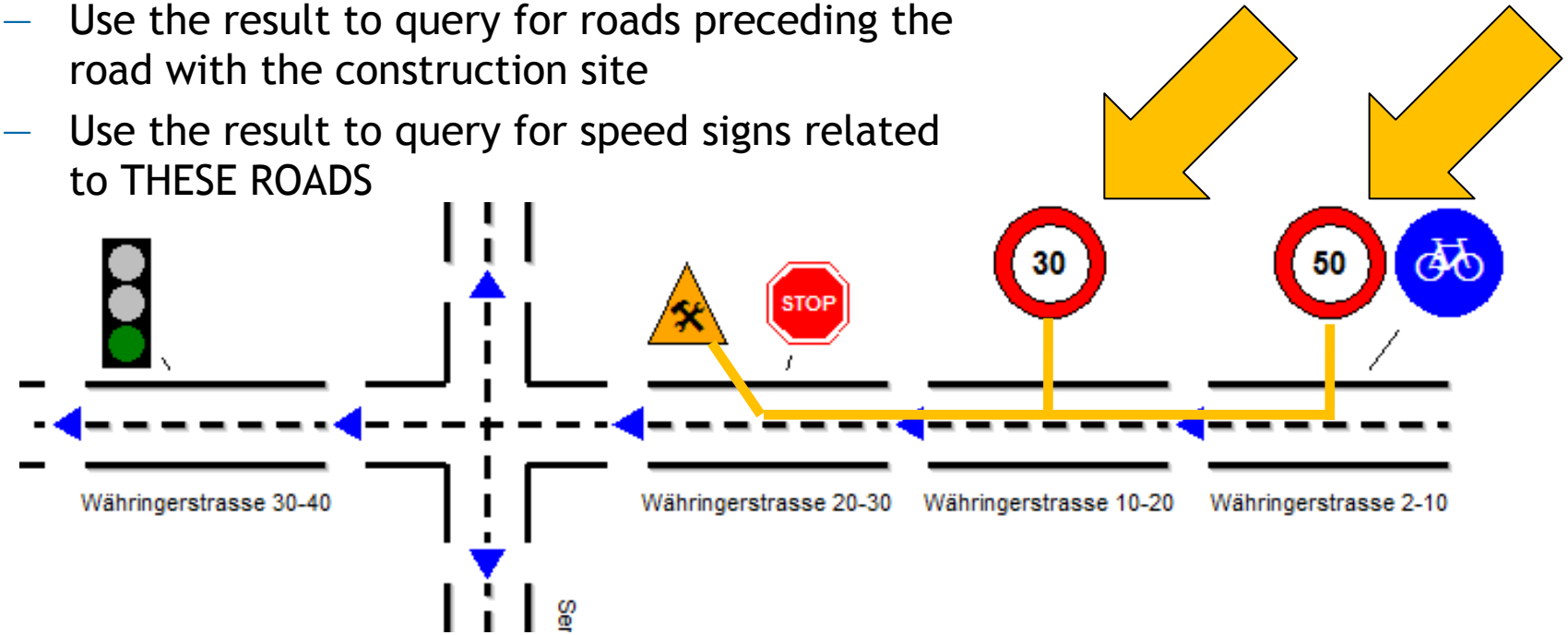
Demo 3: Construction Sites (3/4)

- The urban traffic planner needs to find and adjust all Smart City speed limit signs preceding a construction site
- Technical Solution:
  - Query for construction sites
  - Use the result to query for roads preceding the road with the construction site

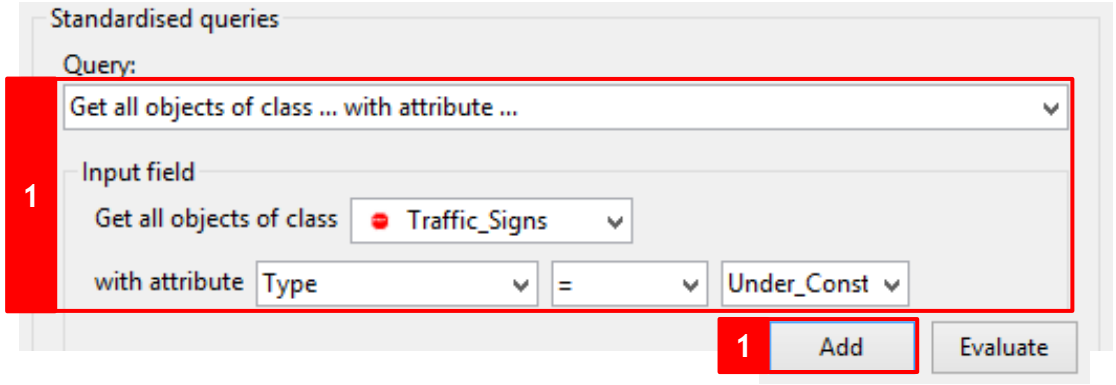


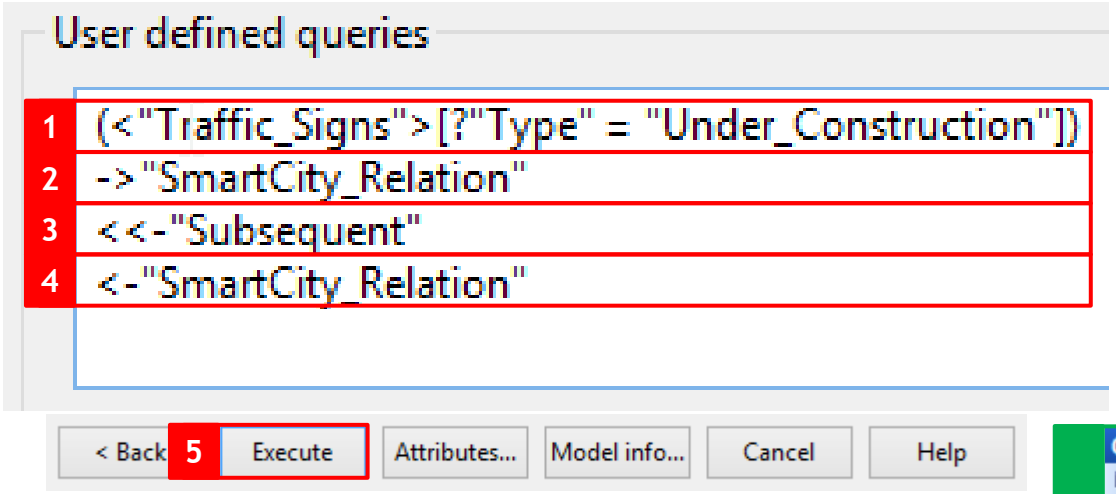
Demo 3: Construction Sites (4/4)

- The urban traffic planner needs to find and adjust all Smart City speed limit signs preceding a construction site
- Technical Solution:
  - Query for construction sites
  - Use the result to query for roads preceding the road with the construction site
  - Use the result to query for speed signs related to THESE ROADS

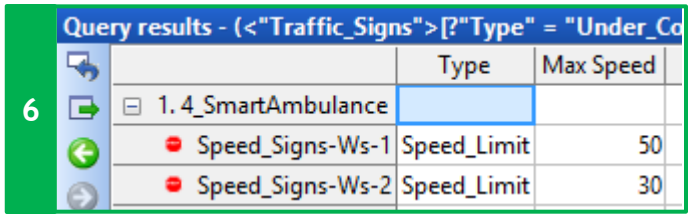


Demo 3: Solution



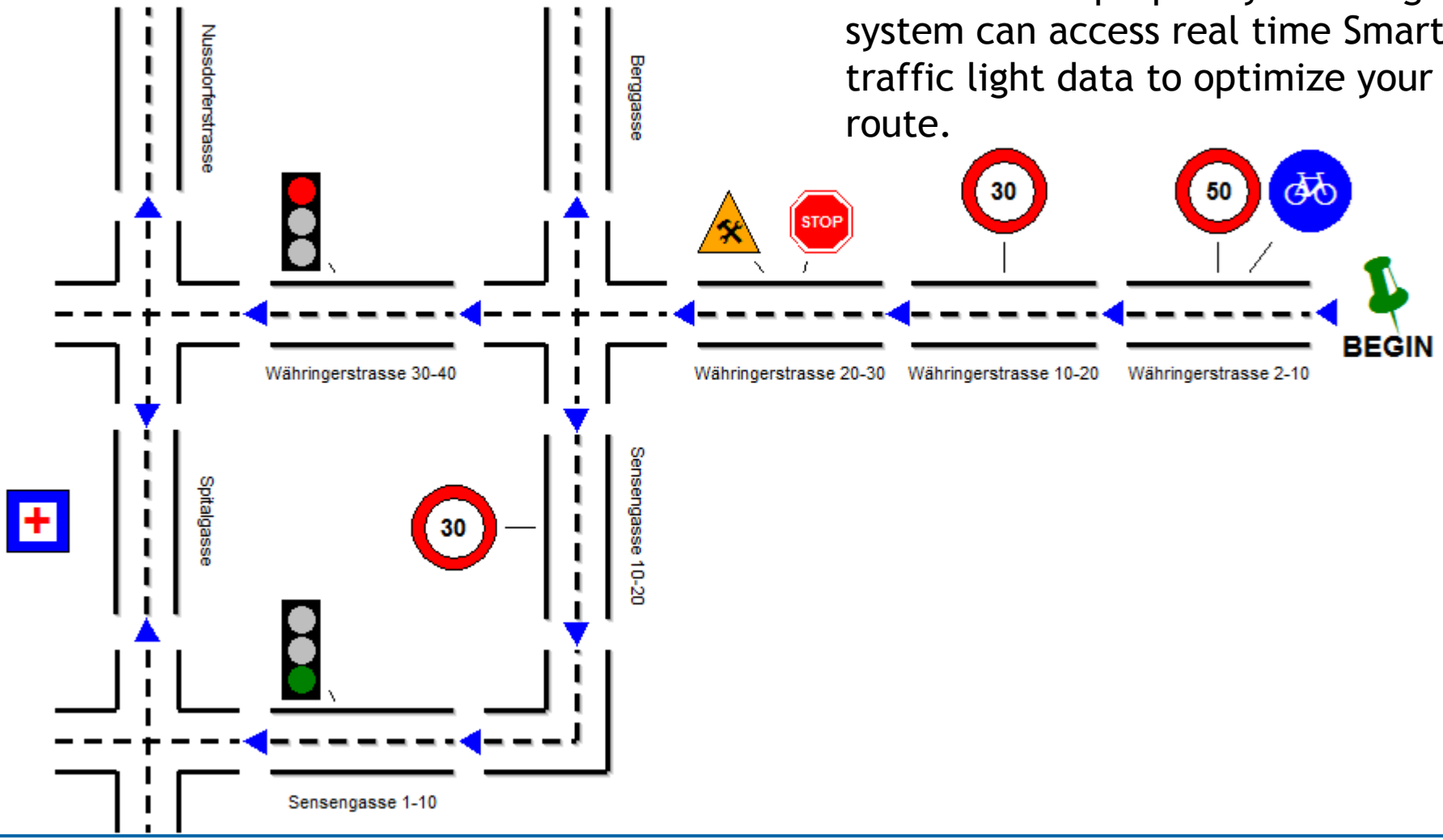


1) Query for construction sites
2) Use the result set to query for the road related to the construction site
3) Use the result set to query for preceeding roads
4) Use the result set to query for signs related to the roads



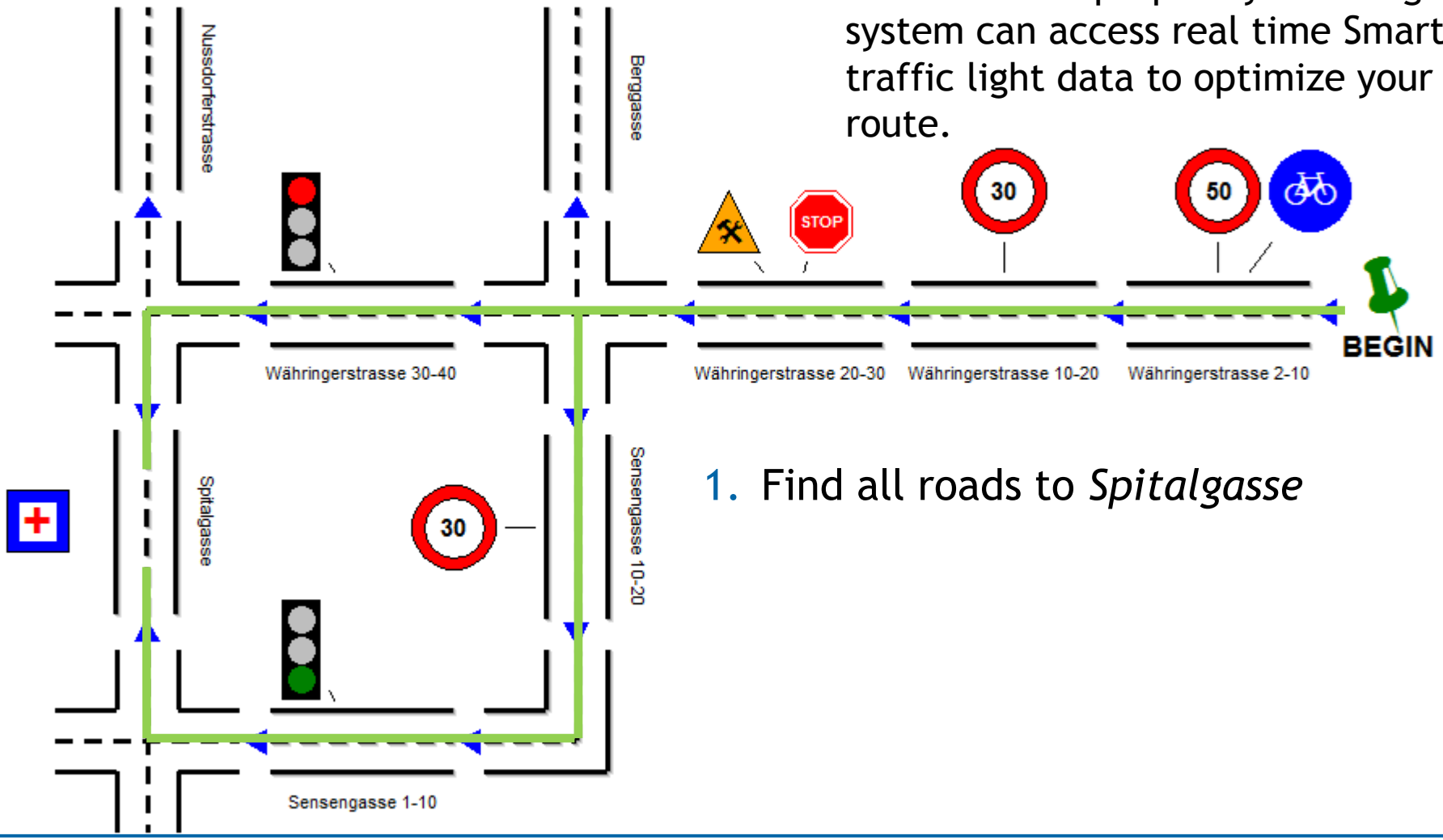
Hands-On 3: Emergency Service in a Smart City (1/5)

You transport an injured person in an ambulance. The condition is not critical, but you want to reduce waiting time. For this purpose your navigation system can access real time Smart City traffic light data to optimize your route.



Hands-On 3: Emergency Service in a Smart City (2/5)

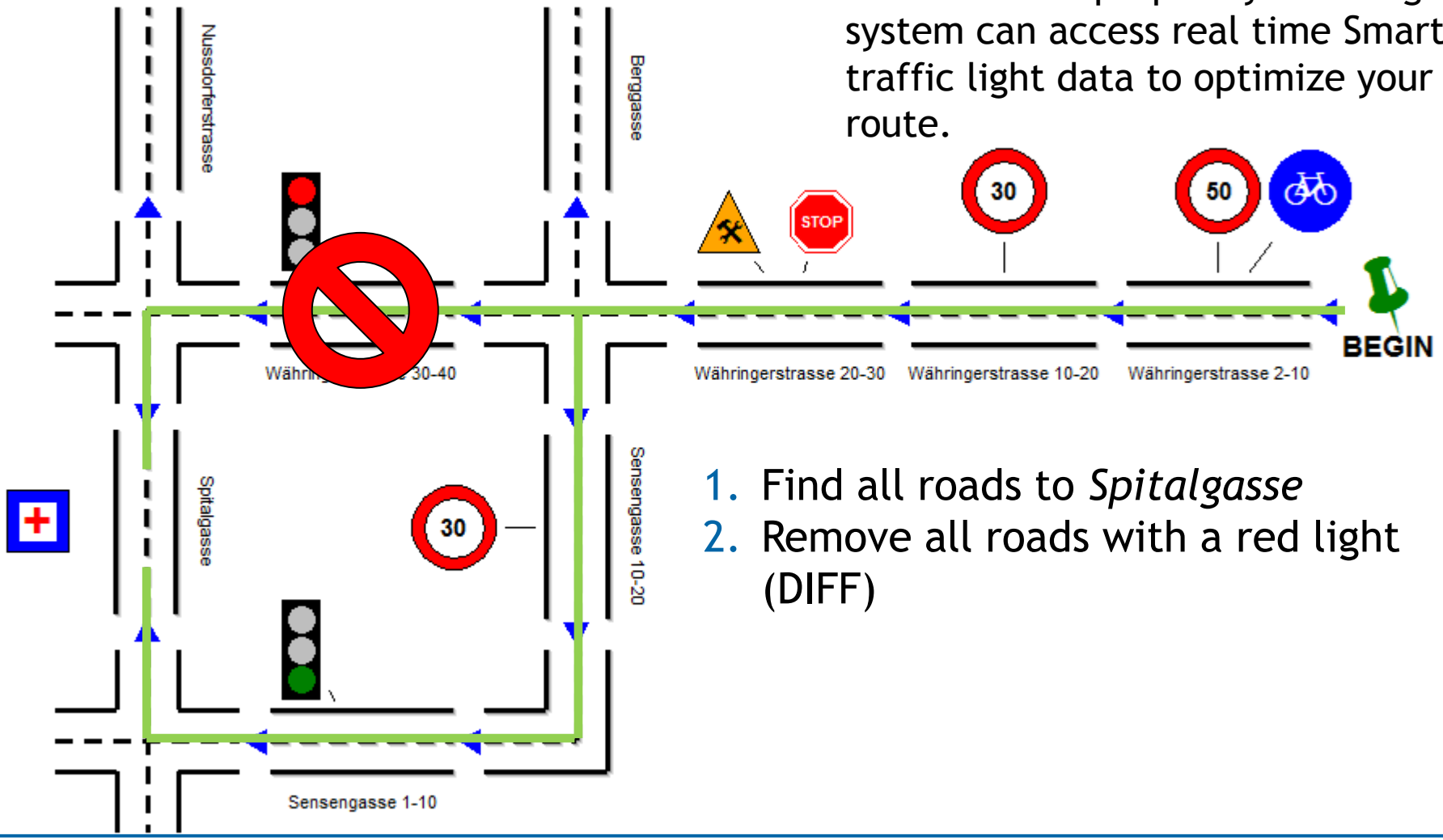
You transport an injured person in an ambulance. The condition is not critical, but you want to reduce waiting time. For this purpose your navigation system can access real time Smart City traffic light data to optimize your route.



1. Find all roads to *Spitalgasse*

Hands-On 3: Emergency Service in a Smart City (3/5)

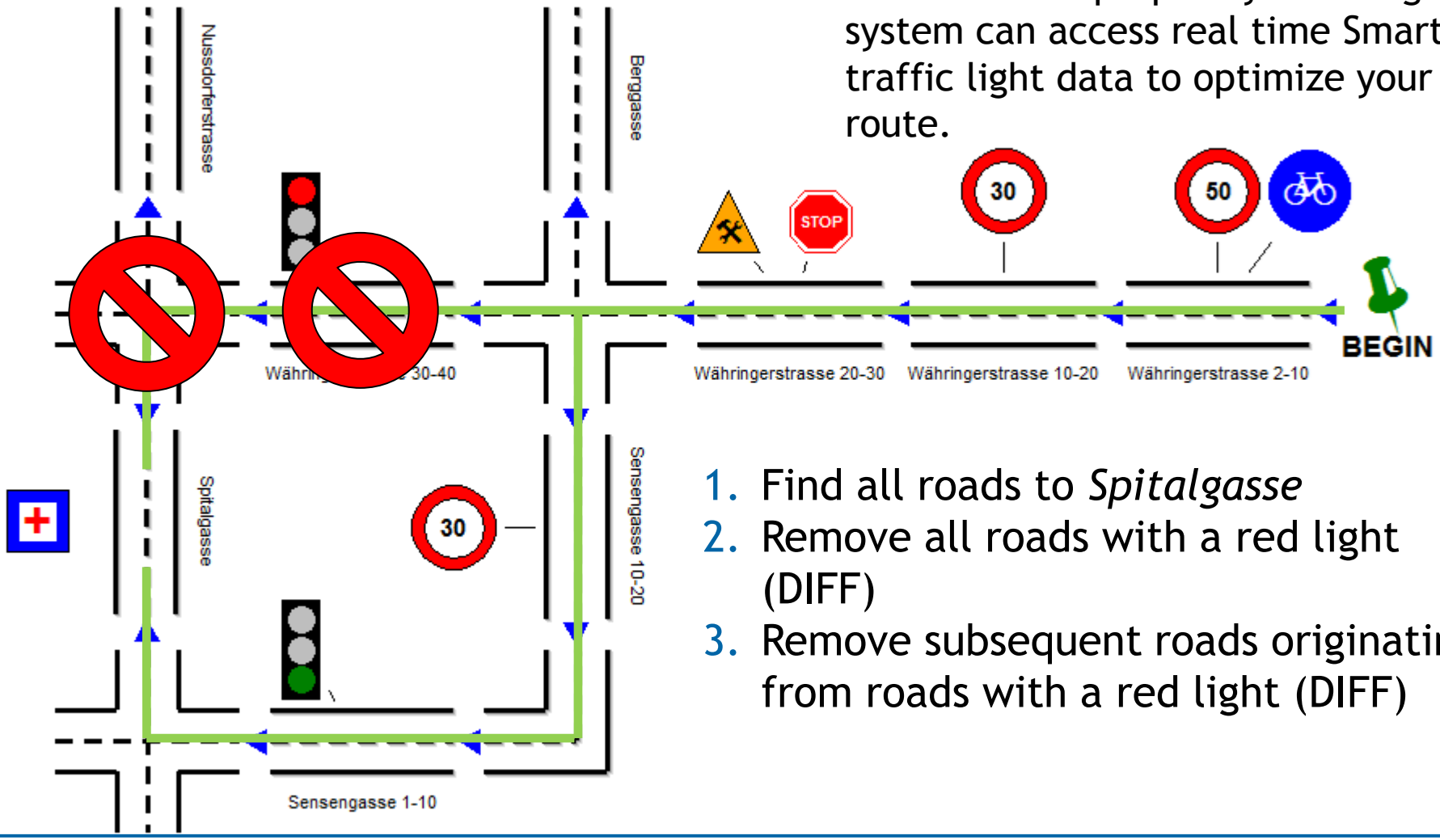
You transport an injured person in an ambulance. The condition is not critical, but you want to reduce waiting time. For this purpose your navigation system can access real time Smart City traffic light data to optimize your route.



1. Find all roads to *Spitalgasse*
2. Remove all roads with a red light (DIFF)

Hands-On 3: Emergency Service in a Smart City (4/5)

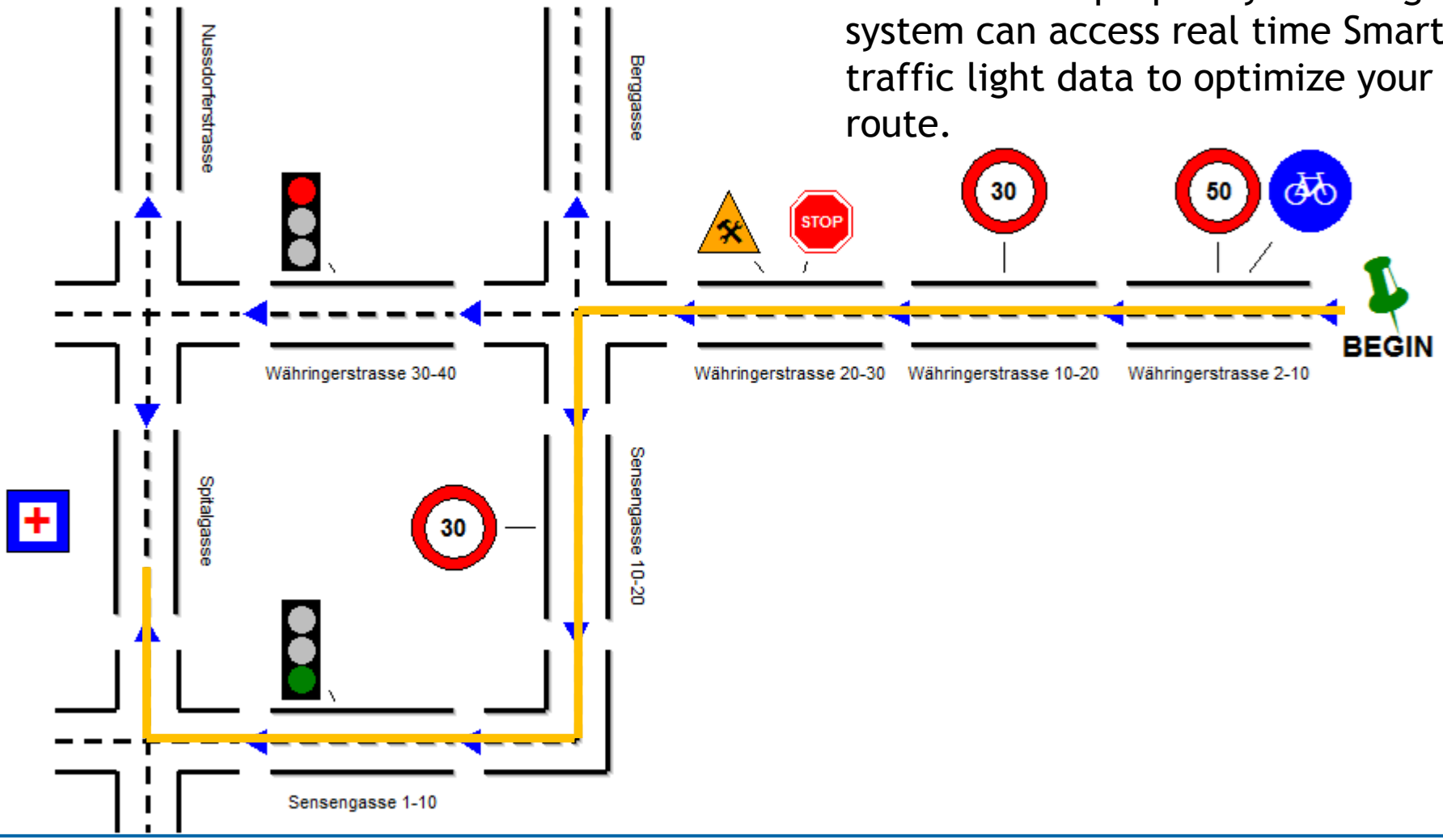
You transport an injured person in an ambulance. The condition is not critical, but you want to reduce waiting time. For this purpose your navigation system can access real time Smart City traffic light data to optimize your route.



1. Find all roads to *Spitalgasse*
2. Remove all roads with a red light (DIFF)
3. Remove subsequent roads originating from roads with a red light (DIFF)

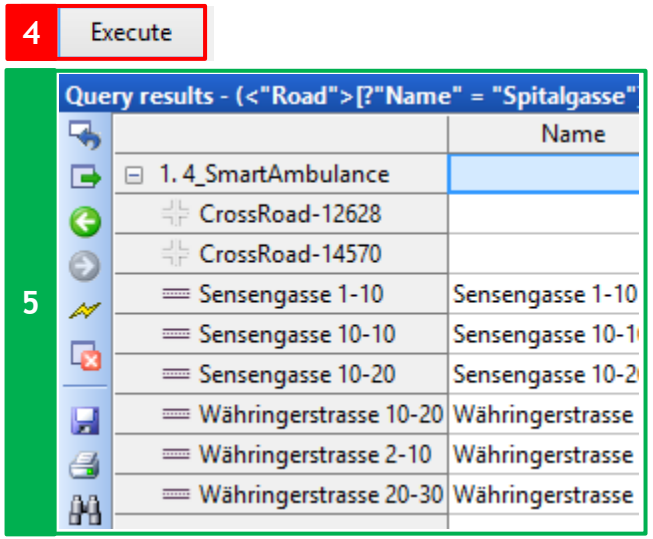
Hands-On 3: Emergency Service in a Smart City (5/5)

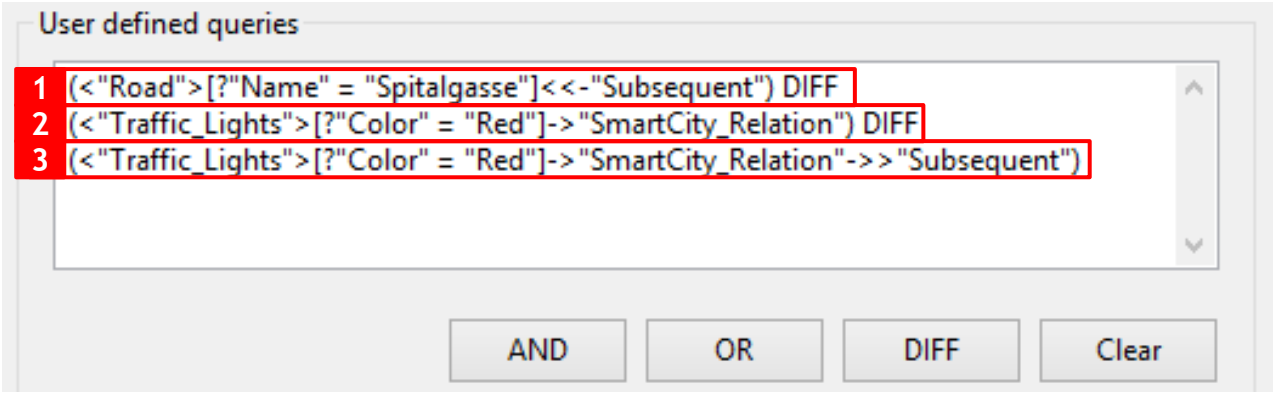
You transport an injured person in an ambulance. The condition is not critical, but you want to reduce waiting time. For this purpose your navigation system can access real time Smart City traffic light data to optimize your route.



Hands-On 3: Solution

- Find a street to the hospital that is not blocked by a red light
  1. Find all roads to *Spitalgasse*
  2. Remove all roads with a red light (DIFF)
  3. Remove subsequent roads originating from roads with a red light (DIFF)



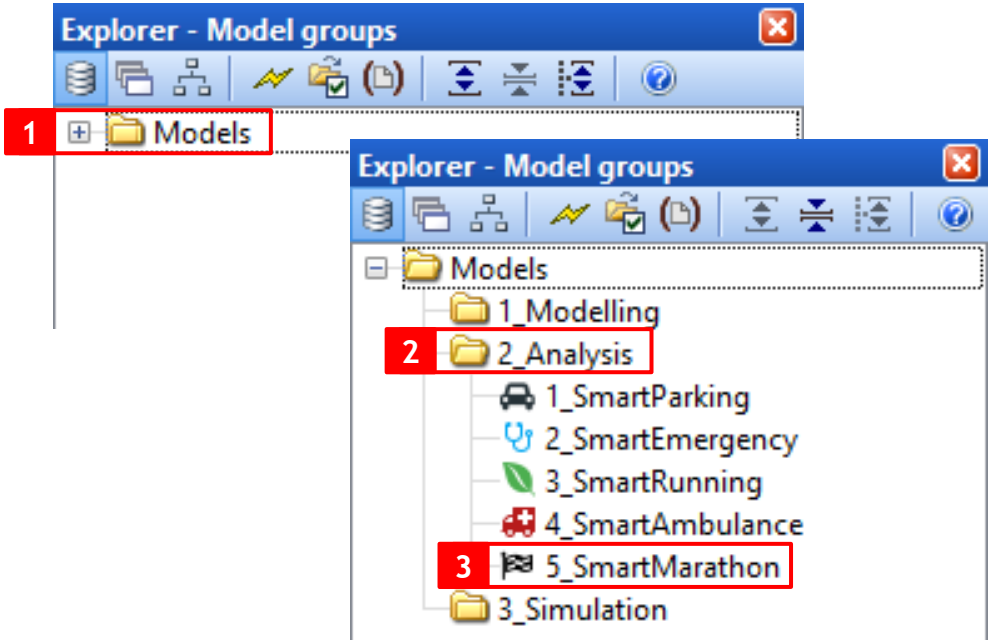


Exercise Structure

1. Standardized queries
   - Basics: The query dialog and the query result window | Smart Traffic Signs
   - Demo: Querying classes and attributes | Smart Parking
   - Hands-on: Operators, wildcards, and changing values | Smart Parking
2. Combination of standardized queries using set operators
   - Basics: Construction of combined queries
   - Demo: Intersection of query result sets | Smart Emergency Response
   - Hands-on: Union of query result sets | Smart Running Track
3. User-defined queries and advanced query combination
   - Basics: Relations and relationship operators, AQL
   - Demo: AQL, transitive relations | Smart Traffic Planning
   - Hands-on: AQL, transitive relations | Smart Ambulance Navigation
4. **Pre-defined analysis queries**
   - **Basics: Extending Mechanisms and Algorithms of Modeling Methods**
   - **Demo: Pre-defined queries | Smart Relay Marathon Runner**
   - **Hands-on: Pre-defined queries | Smart Relay Marathon Organization**

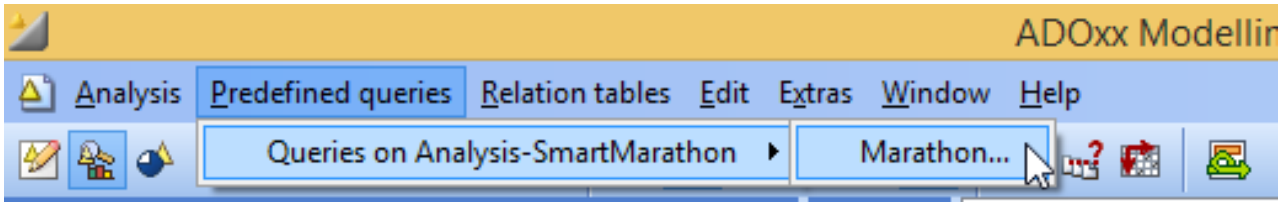
Predefined analysis queries

- Open the **5_SmartMarathon** model within the model group **2_Analysis** in the **Modeling Toolkit**.



Basics: Extending Procedures and Functions of Modeling Methods

- The query functionality in ADOxx can be configured to support e.g., pre-defined queries.
- This pre-configuration supports the implementation of modelling method dependent functionality.
- Properties
  - Session-independent
  - Defined by the method engineer and therefore easy to use
  - The modeler executes the query only by filling place holders
  - Specific for the used model and their modelling language



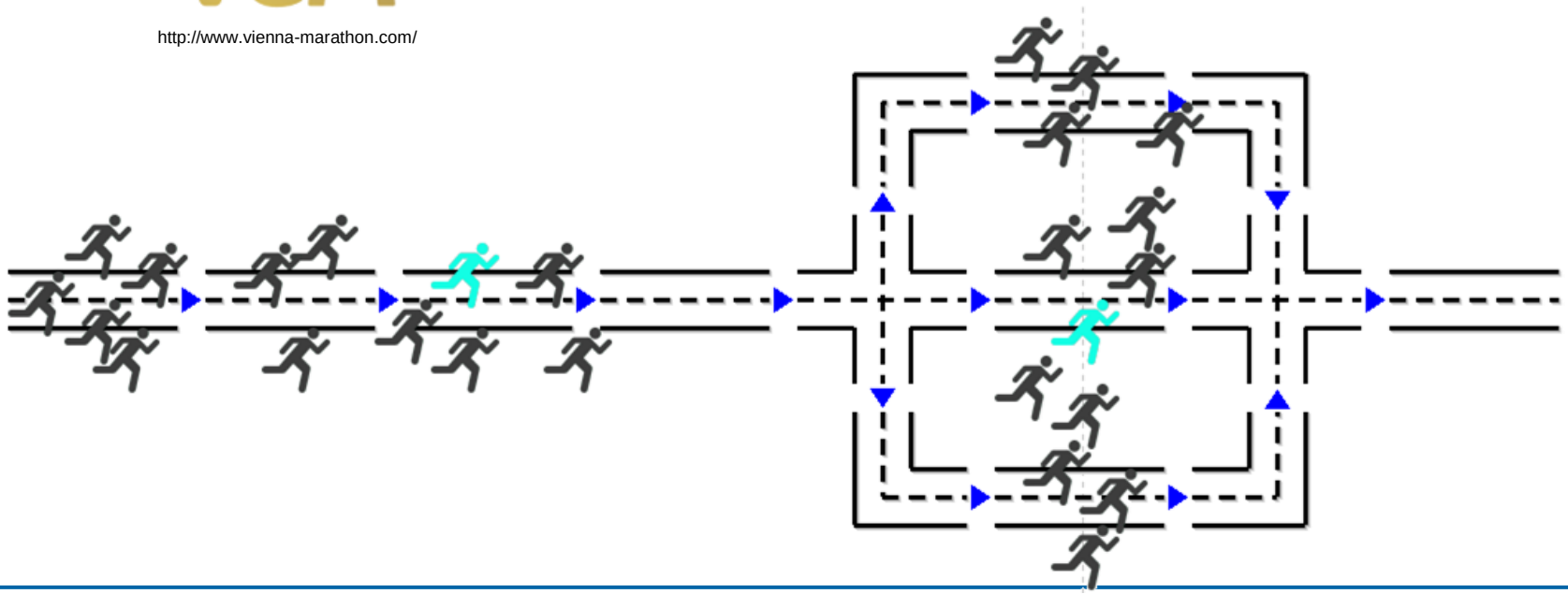
Demo 4: Relay Marathon 1/2

- Relay marathon hand overs are an uncomfortable situation
  - Lots of people waiting
  - No fixed time
  - Runners are exhausted

- Goal: A tool for the waiting person that can track your currently running teammate
  - Once you found your partner, you can pass along the tool (e.g., on a smartphone or smartwatch) and the rest of your stuff (e.g., clothes) to your partner and start running.

Demo 4: Marathon Relay 2/2

http://www.vienna-marathon.com/

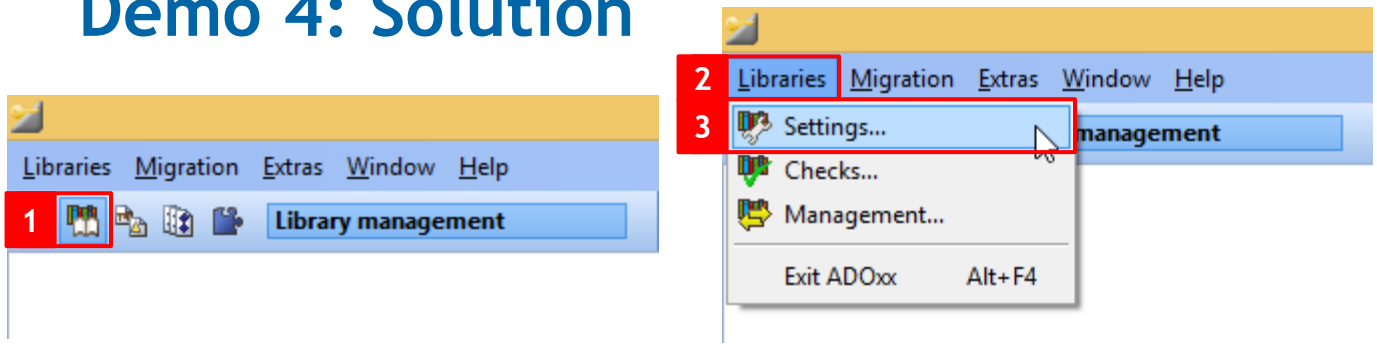


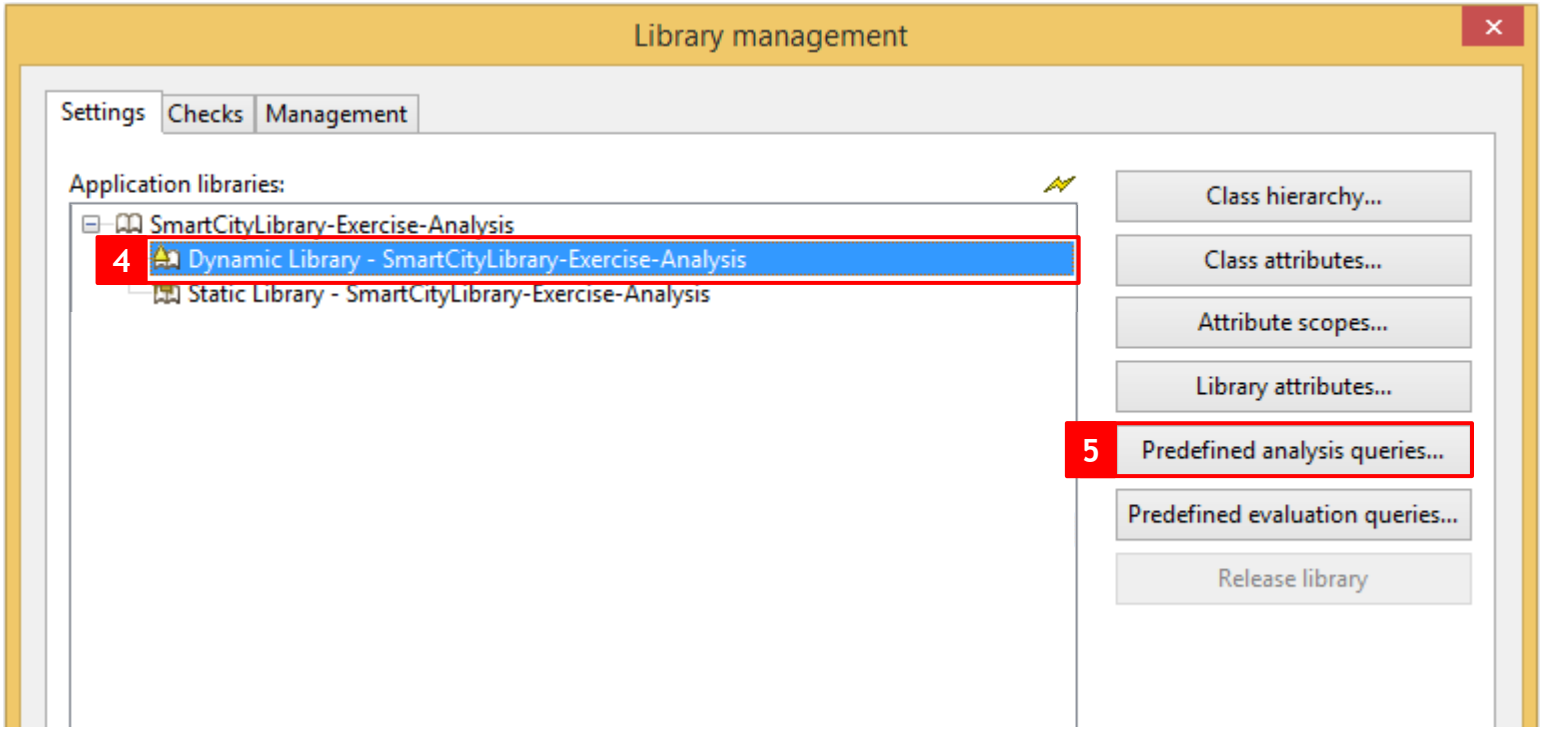
Demo 4: Solution

- **Design** the query in the **Development Toolkit**
  - Create new menu entry
  - Create input fields
  - Customize AQL expressions
  - Paramerterize query with input fields
  - Configure result attributes

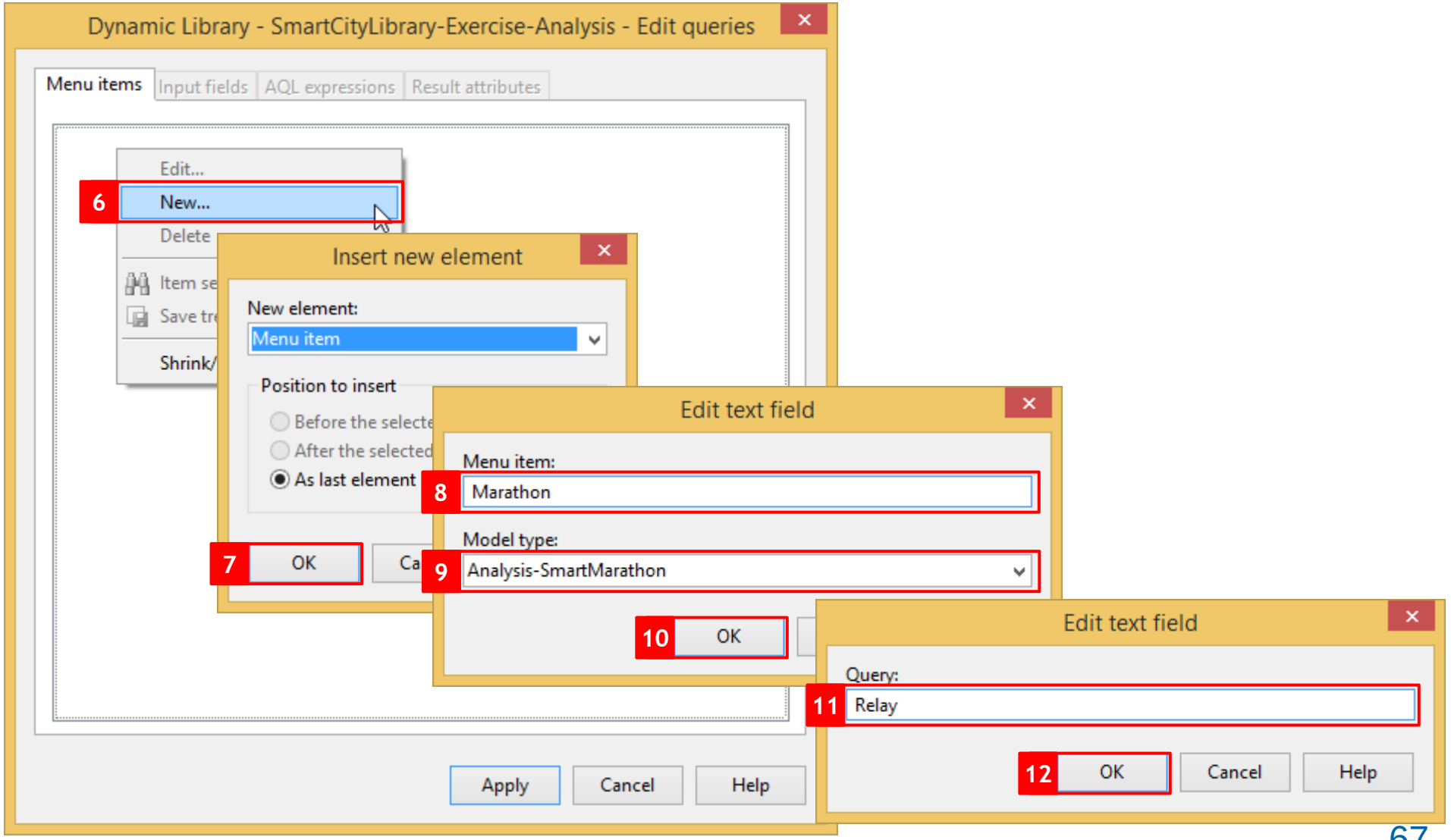
- **Execute** the query in the **Modelling Toolkit**
  - Select query and model
  - Enter input fields
  - Execute query and receive results

Demo 4: Solution

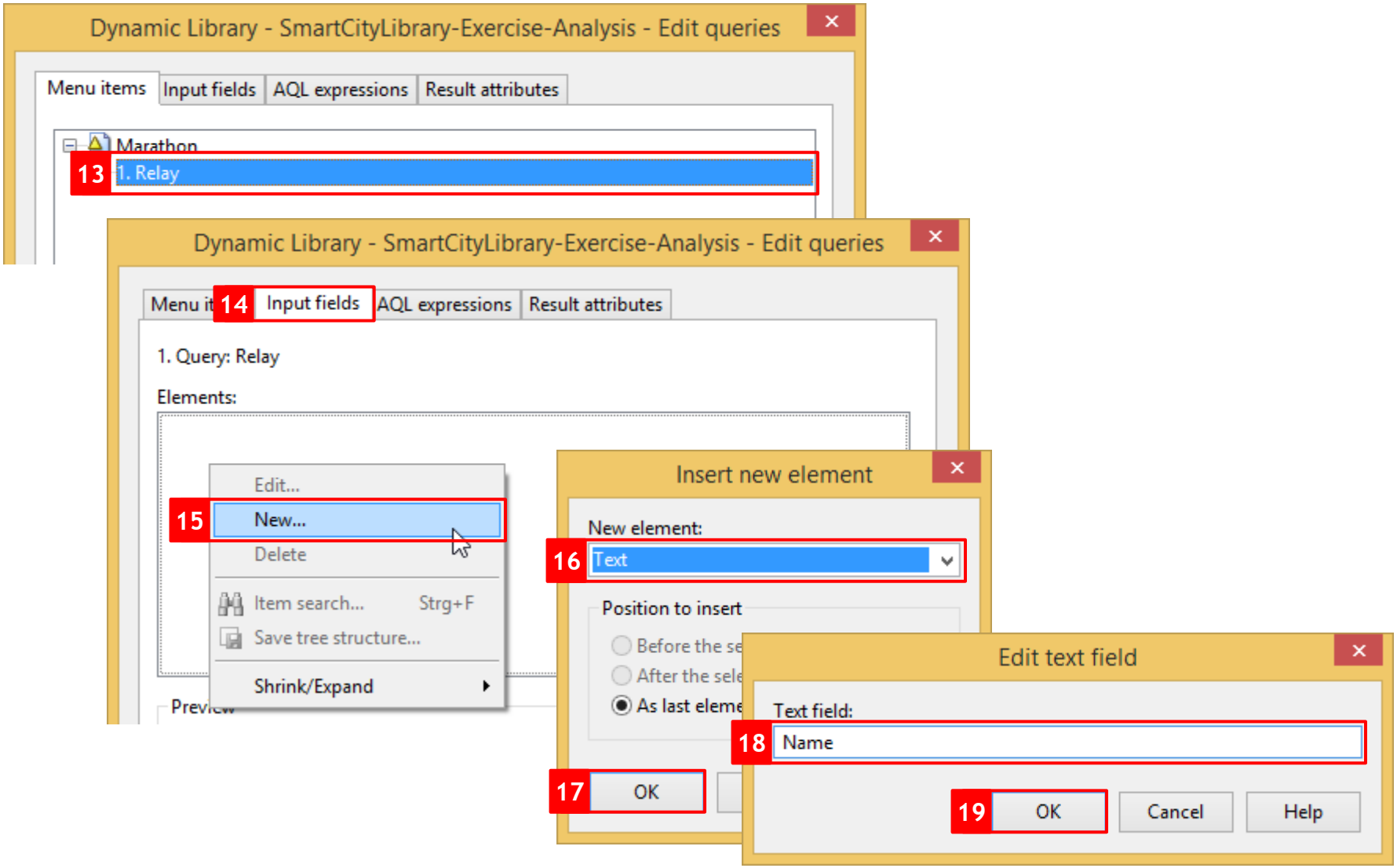




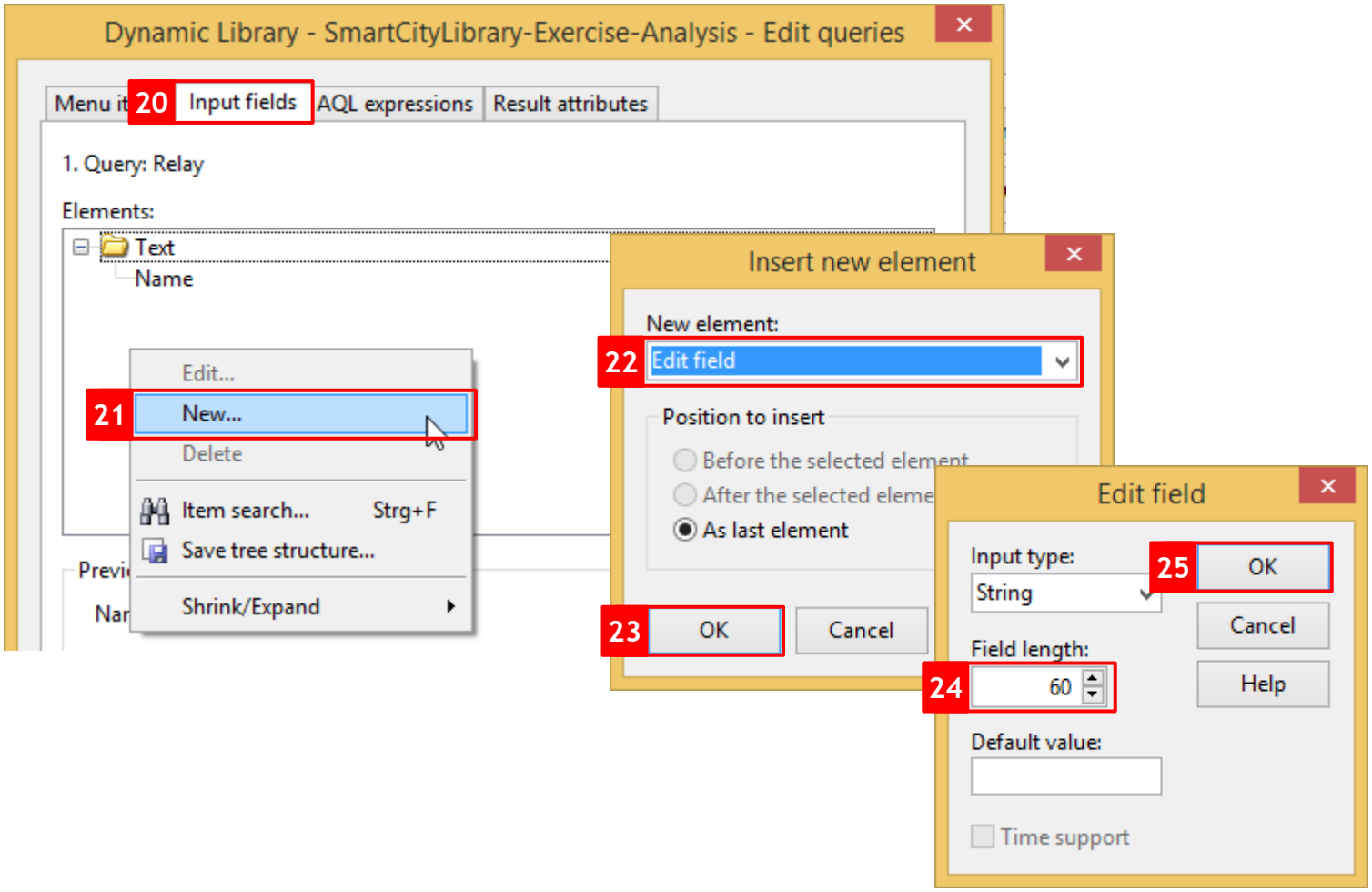
Demo 4: Solution



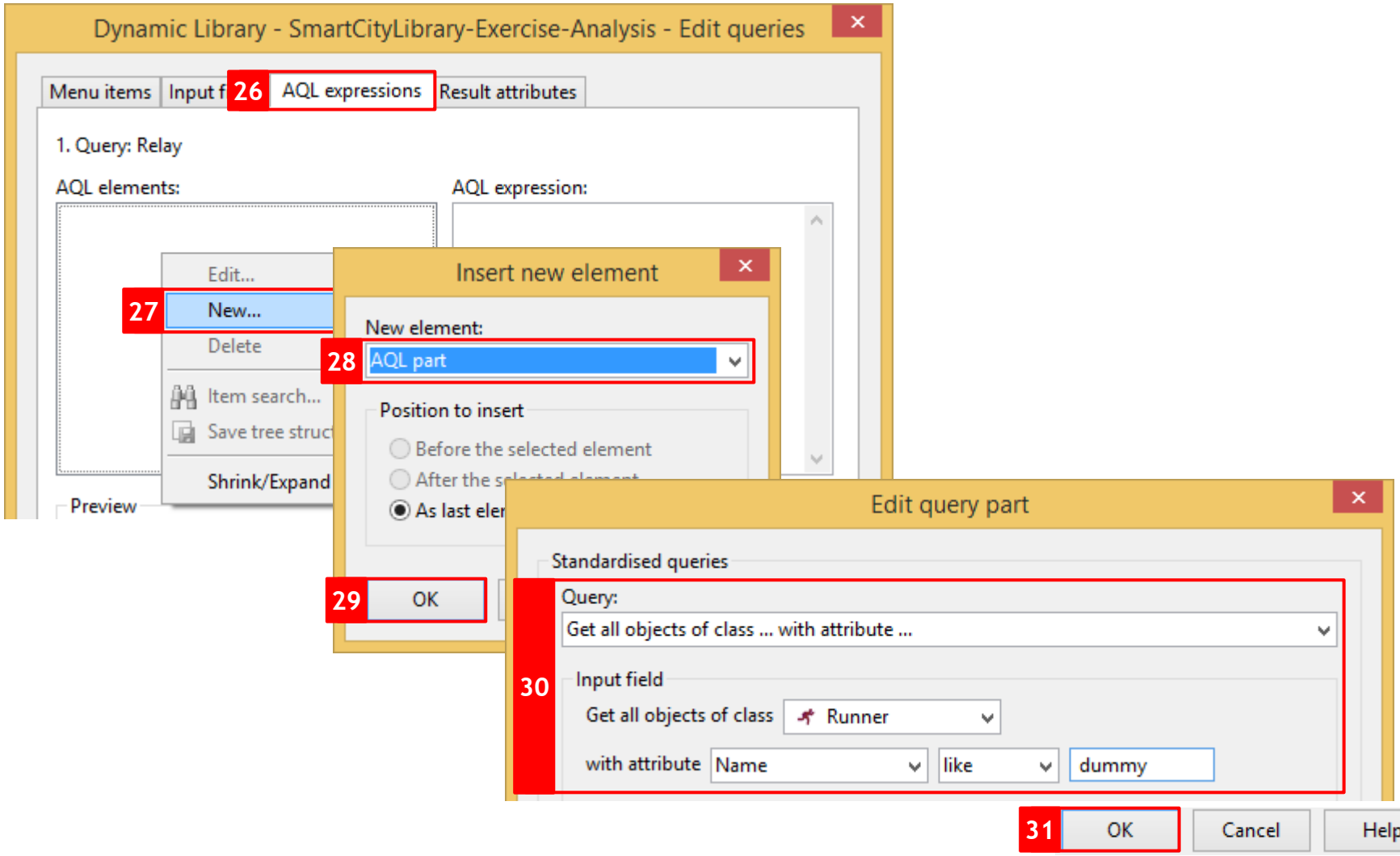
Demo 4: Solution



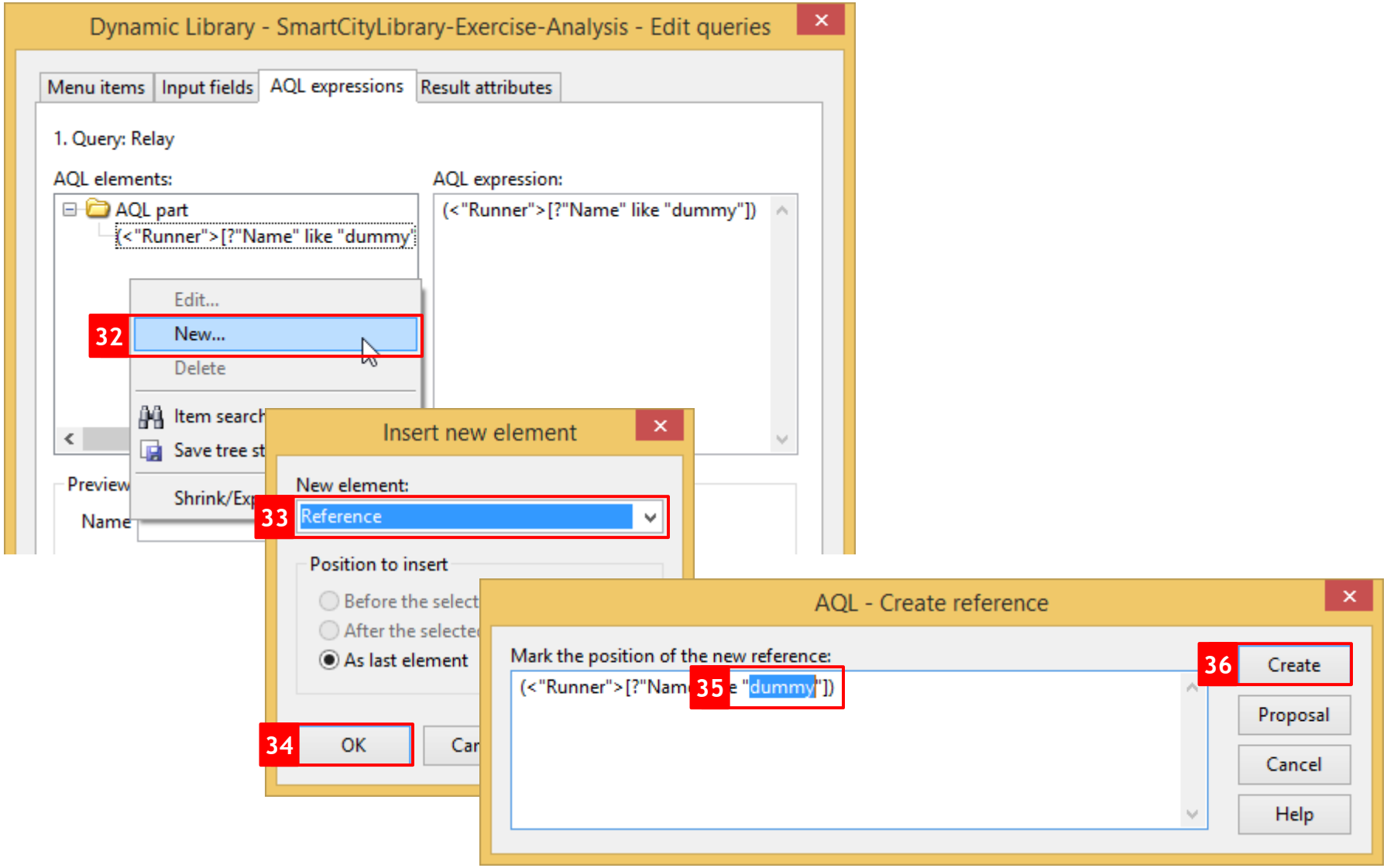
Demo 4: Solution



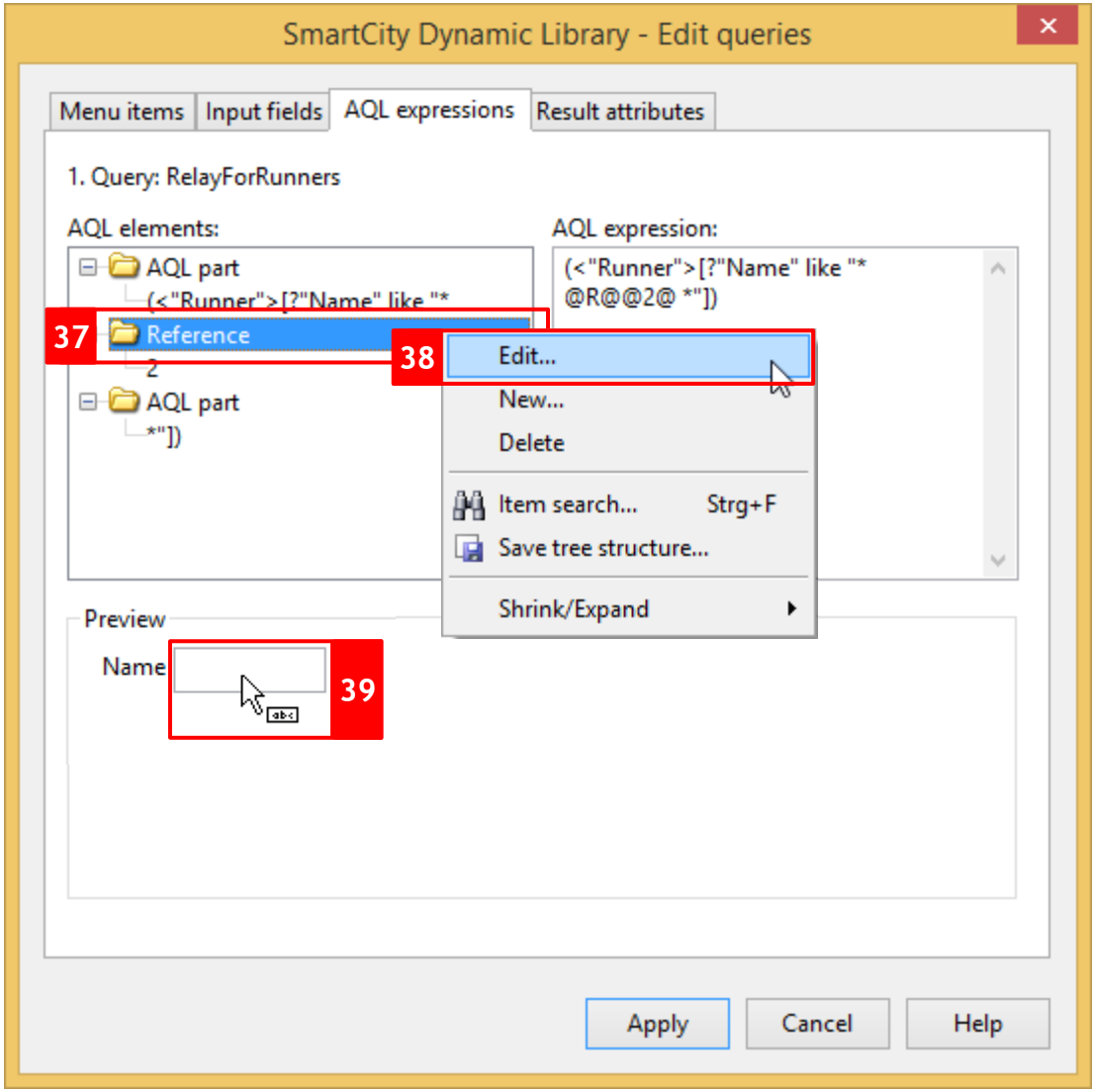
Demo 4: Solution



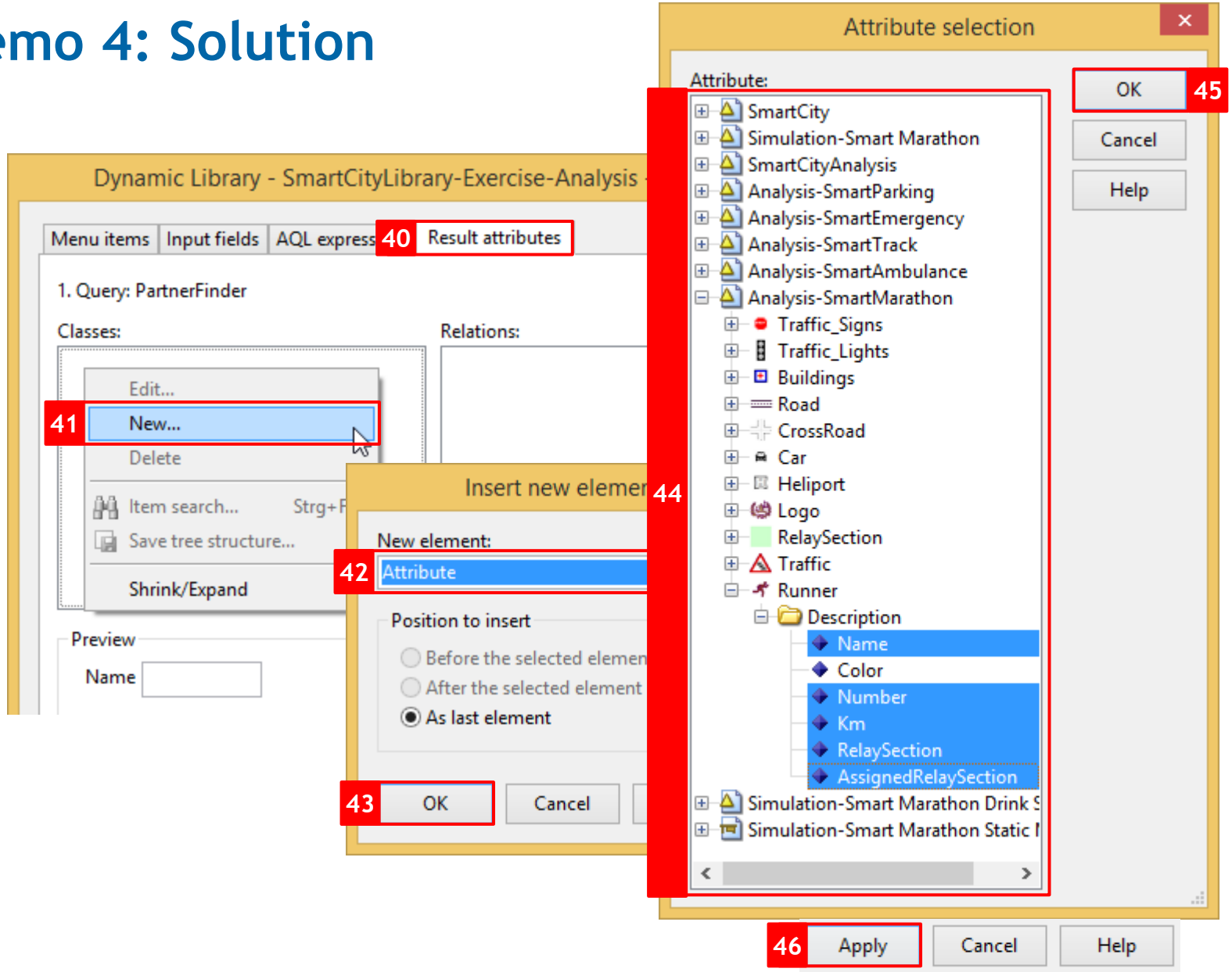
Demo 4: Solution



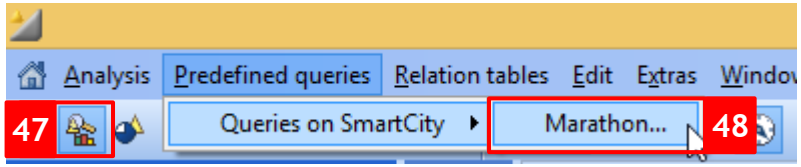
Demo 4: Solution

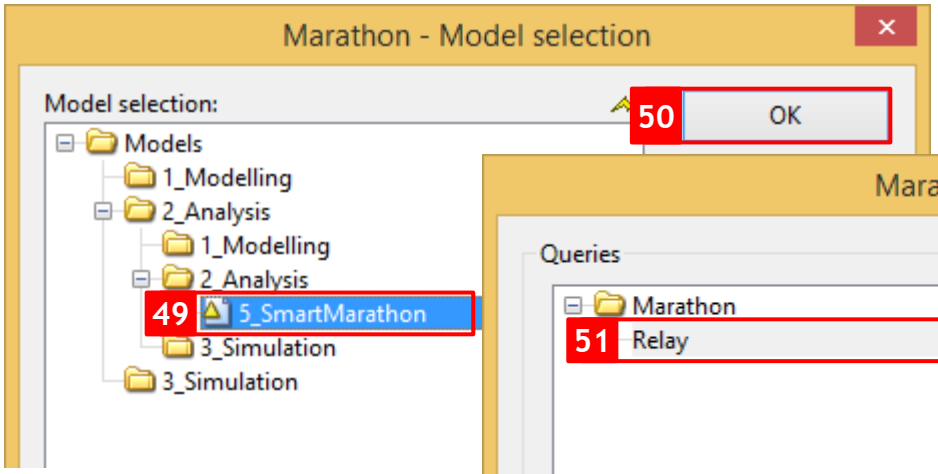


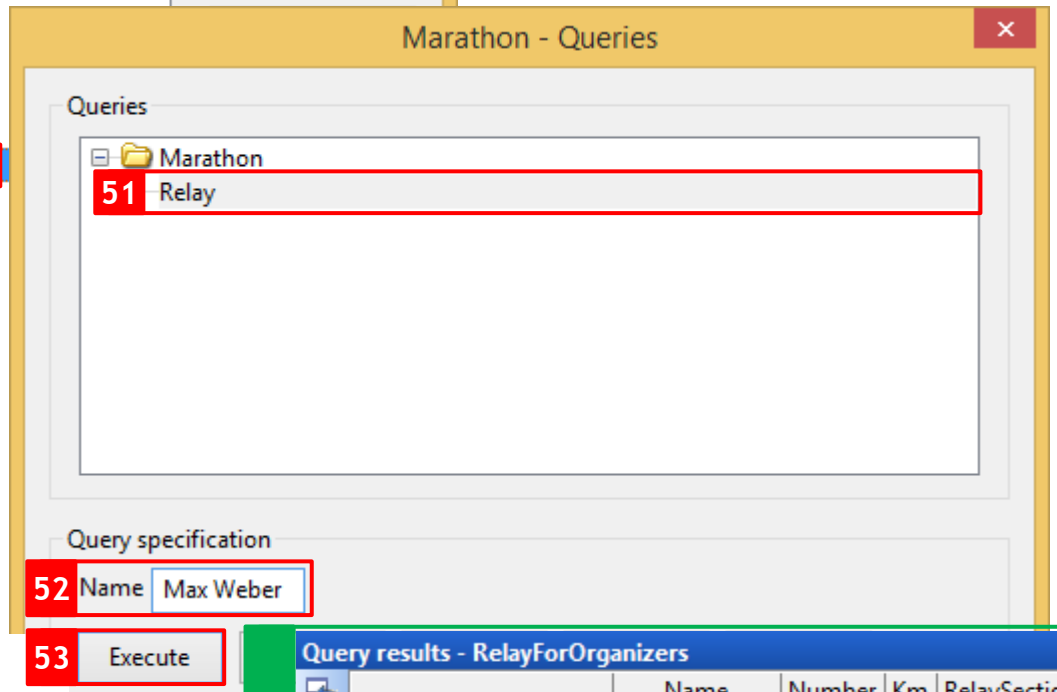
Demo 4: Solution

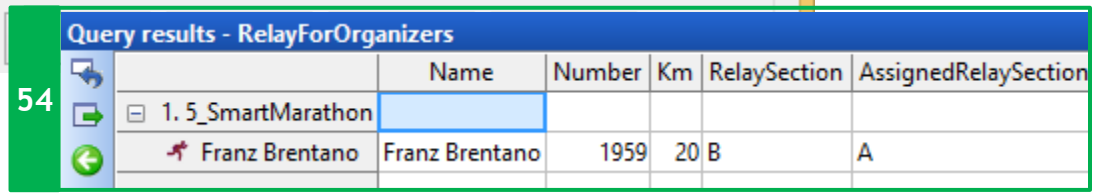


Demo 4: Solution





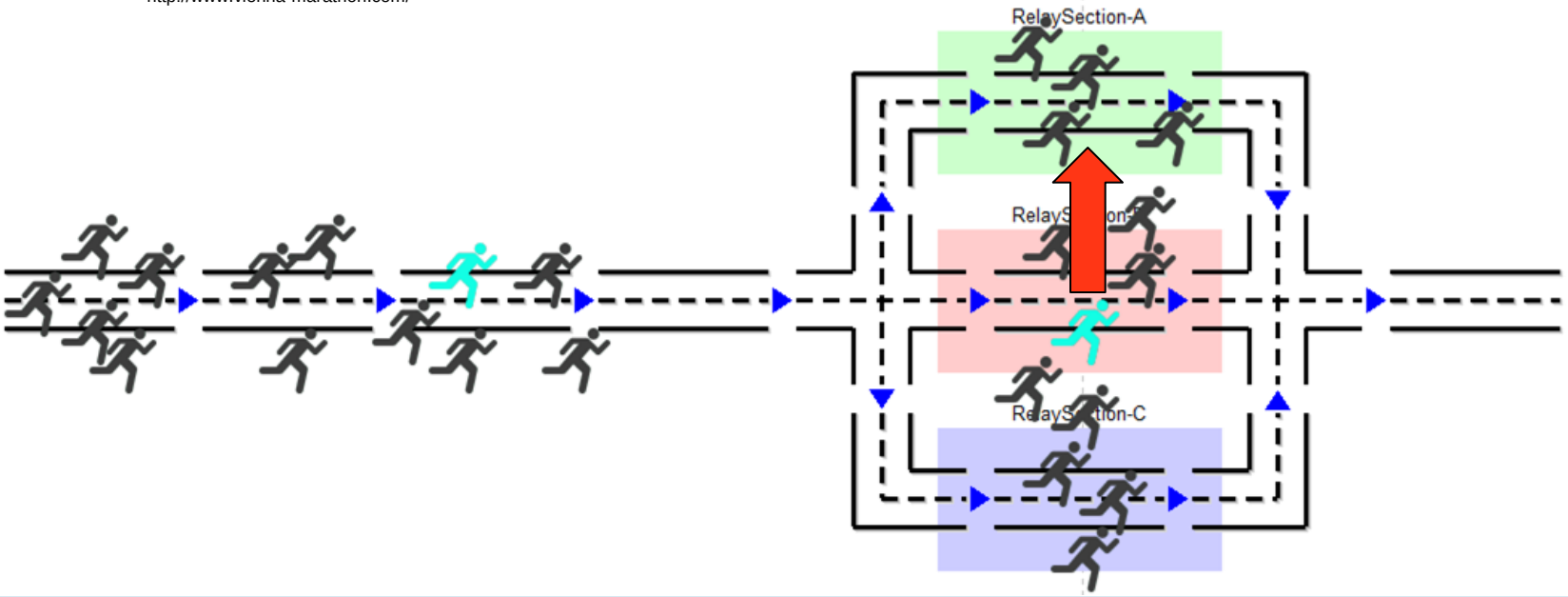




Hands-On 4:

http://www.vienna-marathon.com/

- Support the marathon organization by finding and alerting people that arrive or wait in the wrong relay section



Hands-On 4: Solution (1/2)

Proceed like in the demo example

1. Query for runners inside the relay station A
2. Query for runners whose assigned relay section is not A
3. Create the intersection of queries in 1 and 2
4. Repeat 1-3 for B
5. Repeat 1-3 for C
6. Create the union of 3-5

Hands-On 4: Solution (2/2)

Dynamic Library - SmartCityLibrary-Exercise-Analysis - Edit queries

Menu items | Input fields | AQL expressions | Result attributes

2. Query: RelayForMarathonOrganizer

AQL elements:

AQL part

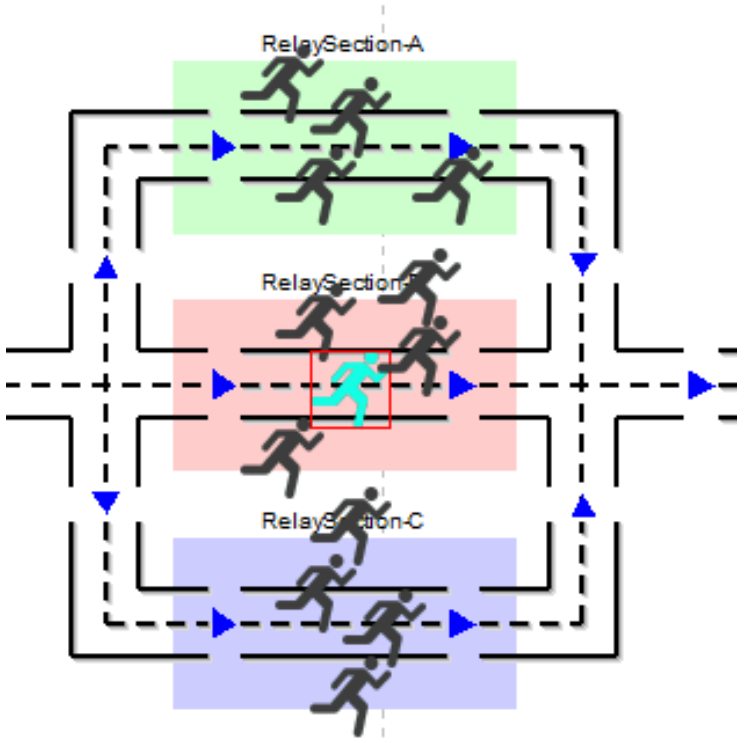
((({"RelaySection-A":"RelaySection"}<

AQL expression:

((({"RelaySection-A":"RelaySection"} <-"Is inside") OR ({"RelaySection-A":"RelaySection"}->"Is inside")) AND (<"Runner">[?"AssignedRelaySection" != "A"])) OR ((({"RelaySection-B":"RelaySection"}<-"Is inside") OR ({"RelaySection-B":"RelaySection"}->"Is inside")) AND (<"Runner">[?"AssignedRelaySection" != "B"])) OR ((({"RelaySection-C":"RelaySection"}

Preview

Apply | Cancel | Help



Query results - RelayForOrganizers

	Name	Number	Km	RelaySection	AssignedRelaySection
1. 5_SmartMarathon					
Franz Brentano	Franz Brentano	1959	20	B	A

NEXT GENERATION ENTERPRISE MODELLING IN THE AGE OF INTERNET OF THINGS
- NEMO 2016 -


Dominik Bork


Nikolaos Tantouris


Niksa Visic


Michael Walch


Simon Doppler


Franz Staffel